Taylor & Francis
Taylor & Francis Group

International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: http://www.tandfonline.com/loi/tprs20

Solving cell formation and task scheduling in cellular manufacturing system by discrete bacteria foraging algorithm

Chunfeng Liu, Jufeng Wang, Joseph Y.-T. Leung & Kai Li

To cite this article: Chunfeng Liu, Jufeng Wang, Joseph Y.-T. Leung & Kai Li (2015): Solving cell formation and task scheduling in cellular manufacturing system by discrete bacteria foraging algorithm, International Journal of Production Research, DOI: 10.1080/00207543.2015.1113328

To link to this article: http://dx.doi.org/10.1080/00207543.2015.1113328

Published online: 13 Nov 2015.

International Journal of Production Research, 2015
http://dx.doi.org/10.1080/00207543.2015.1113328

Solving cell formation and task scheduling in cellular manufacturing system by discrete bacteria foraging algorithm

Chunfeng Liu[a], Jufeng Wang[b], Joseph Y.-T. Leung[cd]* and Kai Li[d]

[a]*School of Management, Hangzhou Dianzi University, Hangzhou, P.R. China;* [b]*Department of Mathematics, China Jiliang University, Hangzhou, P.R. China;* [c]*Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA;* [d]*School of Management, Hefei University of Technology, Hefei, P.R. China*

(*Received 23 June 2015; accepted 21 October 2015*)

We consider a joint decision model of cell formation and task scheduling in cellular manufacturing system under dual-resource constrained (DRC) setting. On one hand, machines and workers are multi-functional and/or multi-skilled, and they are grouped into workstations and cells. On the other hand, there is a processing sequence among operations of the parts which needs to be dispatched to the desirable workstations for processing. Inter-cell movements of parts can reduce the processing times and the makespan but will increase the inter-cell material handling costs. The objective of the problem is to minimise the material handling costs as well as the fixed and operating costs of machines and workers. Due to the NP-hardness of the problem, we propose an efficient discrete bacteria foraging algorithm (DBFA) with elaborately designed solution representation and bacteria evolution operators to solve the proposed problem. We tested our algorithm using randomly generated instances with different sizes and settings by comparing with the original bacteria foraging algorithm and a genetic algorithm. Our results show that the proposed DBFA has better performance than the two compared algorithms with the same running time.

Keywords: cellular manufacturing system; cell formation; group scheduling; bacteria foraging algorithm; operation sequence

1. Introduction

With increasingly short product life cycles and diverse customers in labor-intensive industries, there has been a shift in demands for mid-volume and mid-variety product mixes. Cellular Manufacturing System (CMS) has emerged to cope with such production environments. It is a hybrid system that links the advantages of job shops (flexibility in producing a wide variety of products) and flow lines (efficient flow and high production rate). The CMS has been implemented with favourable results, including better utilisation of resources, material handling and production efficiency. All of these benefits give rise to a reduction in operational costs.

There are two important issues for CMS in labor-intensive industries. One issue is the cell formation which includes the machine (worker) flexibility and assignment, and the other issue is the task scheduling which involves decisions on task dispatching rules as well as task timetabling methods.

For the cell formation issue, there are two constraining resources that need to be considered. The system will have a dual-resource constrained (DRC) requirement such that a part can be processed only if both machine and worker resources are available. Machines and workers need to be multi-functional and/or multi-skilled to perform more than one function. This is due to the lack of purchased machines as well as the shortage of trained workers. From the machine (or worker) flexibility point of view, machines (or workers) are different from each other in terms of capability width and depth. The most desirable capability width is that each machine (worker) is able to process all the tasks. The most common capability depth is that machines (workers) have a different level of proficiency in performing their assigned tasks. As we know, the classical CMS considers grouping machines that can produce families of similar parts in cells, but ignore many manufacturing factors such as human resource and flexibility. Consequently, under the DRC environment, it is more complicated to select appropriate machines and workers to group workstations and form cells than the classical CMS.

For the task scheduling issue, shop floor managers can dispatch different operations of a part to desirable workstation for processing since each workstation may have different efficiency. The part movement is likely to reduce its processing time and the makespan of all the parts, which will result in a reduction of the fixed and operating costs of machines and workers. This movement, however, may occur in different cells which in turn leads to an increase in the inter-cell material handling cost. In many situations, there are processing precedence constraints among the operations of the parts. Therefore, managers often find it difficult to decide whether a part should be moved to another cell for processing the successive operation.

*Corresponding author. Email: leung@njit.edu

Due to the high complexity of CMS with DRC setting, the above two issues are normally studied independently or sequentially, in spite of the inter-relationship between them . The optimisation domain will be restrained when making decision on task scheduling after the completion of cell formation. Consequently, the optimal benefits of the CMS may not be fully realized. In addition, to the best of our knowledge, there are few studies considering multi-skilled resources and operation sequence in this kind of problem. Therefore, simultaneous optimisation of cell formation and task scheduling become a very important area of research in minimising the operational costs.

Ever since Passino (2002) invented the bacteria foraging algorithm (BFA), it has shown a high level optimisation capability in dealing with very complicated NP-hard problems without significantly increasing the computational time. These problems involve portfolio asset selection in financial field (Mishra, Panda, and Majhi 2014), bidding strategy of a supplier (Jain et al. 2015), margin of loading in multimachine power system (Tripathy and Mishra 2015), design strategy of stacked patch resonator (Jain 2015), workspace volume of a three-revolute manipulator (Panda et al. 2014), etc. One of the main challenges for the BFA is to broaden its application to diverse optimisation areas, especially for discrete problems.

The major purpose of this paper is to build an integrated model which can simultaneously group machine and worker resources to cells and schedule parts and operations to minimise the material handling costs as well as the fixed and operating costs of machines and workers. This paper also attempts to develop a discrete bacteria foraging algorithm (DBFA) combining Priority Rule Based Parallel Schedule Generation Scheme (PRBPSGS) for solving this intractable problem.

The remainder of this paper is organized as follows. The literature review related to the cell formation and group scheduling is presented in Section 2. The mathematical model integrating cell formation and task scheduling is formulated in Section 3. In Section 4, the discrete bacteria foraging algorithm (DBFA) is proposed, and its validity is illustrated by a typical case. A conventional genetic algorithm for solving this problem is described in Section 5. The performance of the proposed DBFA, the original algorithm and the conventional genetic algorithm are compared through numerical experiments in Section 6. Finally, the paper closes with a general discussion of the proposed approach as well as a few remarks on future research directions in Section 7.

2. Literature review

In this section, we present related literature review of studies about cell formation and group scheduling in designing the CMS.

2.1 *Cell formation problem in CMS*

Many researchers have studied cell formation problem which determines which part families and machine groups are assigned to which cells. Meta-heuristics have become very important methods to solve this kind of problem. Kia et al. (2012) proposed a model which is able to make decisions of cell formation with group layout in a dynamic environment. This model utilized multi-rows layout to locate machines in the cells configured with flexible shapes, and computed by simulated annealing. Paydar et al. (2010) applied simulated annealing to solve the part-family and machine-cell formation problem, considering intra-cell movement of parts with processing sequence simultaneously. Safaei, Saidi-Mehrabad, and Jabal-Ameli (2008) designed a dynamic cellular manufacturing system considering the batch inter/intra-cell material handling, alternative process plans for part types and machine replication. They developed a hybrid meta-heuristic based on mean field annealing and simulated annealing to solve the problem. Karthikeyan, Saravanan, and Ganesh (2012) used Rank Order Clustering Method to identify products and group cells, and then developed simulated annealing and tabu search methods for processing jobs in order to minimise the total penalty cost. Hamedi et al. (2012) developed a model to group parts, machines and workers and assign them to the generated virtual cells. The model is solved through a multi-objective tabu search algorithm to find optimum or near-optimum solutions. Defersha and Chen (2008) considered cell configuration and lot sizing problem in a dynamic manufacturing environment. They developed a linear programming method, embedded in a genetic algorithm to minimise both the production and quality related costs. Nouri and Tang (2013), Nouri and Hong (2012), Nouri et al. (2010) applied bacteria foraging algorithms to solve cell formation problems in cellular manufacturing systems, considering the cell load variations, operation time, and sequence data in the three articles, respectively.

Some other methods have emerged for cell formation problems, such as two-phase *p*-median approach (Won and Logendran 2015), Mahalanobis distance method (Gupta, Devika, and Panpaliya 2014), fuzzy clustering (Kao and Chen 2013), fuzzy linear programming (Rabbani et al., 2012), branch and bound algorithm (Arkat, Abdollahzadeh, and Ghahve 2012), self-organizing map (Chattopadhyay, Chattopadhyay, and Dan 2011), etc. For instance, Boutsinas (2013) used bi-clustering method to study cell formation to minimise the sum of the intra-cell voids and inter-cell moves. Egilmez, Erenay, and Suer (2014) proposed a hierarchical method consisting of four phases to solve the cell loading with products and manpower allocation problem, considering worker-based probabilistic processing times and probabilistic product demand.

Some optimisation softwares have been used for intractable cell formation problems. Mahdavi et al. (2010) designed an integer programming model for the cell formation of dynamic cellular manufacturing system, considering production planning and worker assignment. They employed LINGO package to minimise the holding and back-order costs, inter-cell material handling cost, machine and reconfiguration costs and human resource costs. Renna and Ambrico (2015) proposed a mathematical model developed in LINGO package to support the design and reconfiguration activities in a CMS. kioon, Bulgak, and Bektas (2009) designed an integrated cellular manufacturing system model with production planning and dynamic system reconfiguration. They applied CPLEX to solve small and medium-sized problems. Sakhaii et al. (forthcoming) presented an integrated dynamic CMS and production planning with unreliable machines. They applied CPLEX to validate the performance of the proposed robust optimisation approach. Rafiei et al. (2015) addressed the cell formation problem considering two conflicting objective functions: (1) minimising the sum of machine purchasing, operating, inter-cell movements, machine relocation, and machine transferring costs, and (2) minimising the work-in-process in terms of inter-cell batch sizes. The problem is solved by Baron Solver.

2.2 *Group scheduling problem in CMS*

In comparison with the cell formation problem, there are only a few papers addressing group scheduling, which determines in which sequence the parts are processed. Sometimes, group scheduling is called parts scheduling in some articles. Taghavifard (2012) studied cellular manufacturing scheduling problem with sequence-dependent setup times, and solved it using ant colony optimisation and GA. Halat and Bashirzadeh (2015) suggested a GA-based heuristic for scheduling operations of manufacturing cells considering sequence-dependent family setup times and intercellular transportation times. Solimanpur and Elmi (2011) proposed a tabu search method for group scheduling in buffer-constrained flow shop cells to minimise the makespan. Elmi et al. (2011) suggested a simulated annealing algorithm for the job shop cell scheduling problem with inter-cell movement and reentrant parts so as to minimise the makespan. Tang et al. (2010) used scatter search for parts scheduling problem with inter-cell movement to minimise the total weighted tardiness. Saravanan and Haq (2008) used scatter search to solve a similar problem that minimises the makespan. Tavakkoli-Moghaddam et al. (2010) designed a model for a multi-criteria group scheduling problem in CMS, and used scatter search to minimise the makespan and the costs of intra-cell movement, tardiness, and sequence-dependent setup, simultaneously. Venkataramanaiah (2008) addressed the parts scheduling in flow-line-based CMS with missing operations, i.e. certain parts do not require processing on some of the machines in a cell, and proposed a simulated annealing algorithm to find a sequence and schedule that minimise the weighted sum of makespan, flow time and idle time.

2.3 *Integrated decision of cell formation and group scheduling*

Recently, some researchers started to exploit the integrated decision of cell formation and group scheduling problem. Tang et al. (2014) used Lagrangian relaxation decomposition with heuristic to solve the problem of minimising the total tardiness cost. Eguia et al. (2013) suggested a tabu search algorithm for cell formation and parts scheduling to minimise the reconfiguration and under-utilisation costs. Arkat, Farahani, and Ahmadizar (2012) presented a model to concurrently identify cell formation with cellular layout and group scheduling with the objective of minimising the total transportation cost of parts as well as minimising the makespan. They developed a multi-objective genetic algorithm to solve the problem. Arkat, Farahani, and Hosseini (2012) found that the genetic algorithm that simultaneously solves the above problem is better than another genetic algorithm that sequentially solves the problem. Wang, Tang, and Yung (2010) addressed a joint decision of cell formation and parts scheduling in batches, and proposed a scatter search approach with dispatching rules to minimise the total tardiness cost. Wu et al. (2007) exploited a model to simultaneously make the decision of cell formation with cellular layout and group scheduling, and proposed a hierarchical genetic algorithm to minimise the makespan.

Many of these researches have considered the impact of intercell transfer of parts on cell formation and group scheduling. Moreover, their optimisation objectives are related with time, such as transfer time, makespan or tardiness. In this kind of integrated models few studies, however, have involved multi-skilled human resources, operation sequence and optimisation objectives related with transfer cost, resource cost and makespan.

3. Problem statement and formulation

In this section, we formulate the cell formation and task scheduling problem as a non-linear 0–1 integer programming. The objective is to minimise the sum of the inter-cell material handling cost, the fixed and operating costs of machines and workers. The main constraints are the operational precedence constraints and the availability of machines and workers. The problem is formulated according to the following assumptions:

(1) The number of parts is known in advance. All parts should be finished in the planning horizon.

(2) Each part has a number of operations among which there is a chain precedence relation. However, there is no precedence constraint among different parts.

(3) Each operation of each part is processed at one workstation. All machine types are multi-functional and all worker types are multi-skilled. Each operation of each part is able to be processed on any machine type with any worker type, but the processing times may be different. The processing time is known and depends on the assigned machine and worker type. In addition, each operation of each part only needs one machine with one worker to process.

(4) The number of machine types and the number of machines of each type are known in advance. The number of machines, the number of workers and the number of workstations are equal. One machine and one worker are grouped as a pair and assigned to a workstation.

(5) The processing of each operation of each part is not interrupted. The setup time of each part on the machine is ignored.

(6) The base salary of each worker type is known and considered, regardless of whether the worker is active or not.

(7) The maintenance and overhead costs of each machine type are known and considered, regardless of whether the machine is active or not.

(8) The operating costs of each machine type and worker type are known. These costs depend on the workload assigned to the machine and worker.

(9) The number of cells to be formed is given and is constant through the production period. The maximum and minimum of the cell size in terms of the number of grouped workstations are specified in advance. Too many workstations in a cell may generate cluttered flows in a cell due to many routes, whereas too few workstations in a cell may increase the movements of parts to other cells for processing.

(10) The material handling devices moving the parts between cells are assumed to carry only one part at a time. Inter-cell material handing cost of each part is known. The cost is constant for one move between cells regardless of the distance. Intra-cell material handing cost can be ignored. The time of moving part is trivial in comparison with its processing time and can be ignored.

Input parameters

M Number of machine types.
ϖ_m Number of machines of type m, m is denoted as index for the machine types ($m = 1, 2, \ldots, M$).
W Number of worker types.
ω_w Number of workers of type w, w is denoted as index for the worker types ($w = 1, 2, \ldots, W$).
P Number of parts, p is denoted as index for the parts ($p = 1, 2, \ldots, P$).
K_p Number of operations in part p, k is denoted as index for the operations in part p ($k = 1, 2, \ldots, K_p$).
L Number of workstations, l is denoted as index for the workstations ($l = 1, 2, \ldots, L$).
C Number of cells, c is denoted as index for the cells ($c = 1, 2, \ldots, C$).
δ_{pkmw} Processing time (measured in hours) of operation k in part p on machine type m with worker type w.
α_m Maintenance and overhead costs (i.e. fixed costs) per unit of machine type m per hour.
$\widetilde{\alpha_m}$ Operating cost per unit of machine type m per hour.
S_w Base salary cost per unit of worker type w per hour.
$\widetilde{S_w}$ Operating salary cost per unit of worker type w per hour.
θ_p Inter-cell material handling cost of part p.
T Upper bound of finish time of all parts $\left(T = \sum_{p=1}^{P} \sum_{k=1}^{K_p} \max\left\{\delta_{pkmw} | m = 1, 2, \ldots, M; w = 1, 2, \ldots, W\right\}\right)$.
B_u Upper bound of each cell size (measured in the number of workstations).
B_d Lower bound of each cell size (measured in the number of workstations).

Decision variables

X_{ml} 1 if one unit of machine type m is assigned to workstation l, and 0 otherwise.
Y_{wl} 1 if one unit of worker type w is assigned to workstation l, and 0 otherwise.
Z_{cl} 1 if workstation l is assigned to cell c, and 0 otherwise.
Q_{pkl} 1 if operation k of part p is processed in workstation l, and 0 otherwise.
S_{pkt} 1 if operation k of part p starts to be processed at time instant t, and 0 otherwise ($t = 0, 1, \ldots, T$).

According to the input parameters and decision variables, an intermediate variable is defined as:

J_{pk} The processing time of operation k of part p,

$$J_{pk} = \sum_{l=1}^{L} \sum_{m=1}^{M} \sum_{w=1}^{W} Q_{pkl} \cdot X_{ml} \cdot Y_{wl} \cdot \delta_{pkmw}.$$

The problem can be formulated as a non-linear 0-1 integer programming as follows:

$$\begin{aligned}
\text{Min} \quad & \sum_{p=1}^{P} \left[\sum_{k=1}^{K_p-1} \frac{1}{2} \cdot \sum_{c=1}^{C} \left| \sum_{l=1}^{L} Q_{pkl} \cdot Z_{cl} - \sum_{l=1}^{L} Q_{p,k+1,l} \cdot Z_{cl} \right| \right] \cdot \theta_p \\
& + \left[\sum_{m=1}^{M} \left(\sum_{l=1}^{L} X_{ml} \right) \cdot \alpha_m + \sum_{w=1}^{W} \left(\sum_{l=1}^{L} Y_{wl} \right) \cdot S_w \right] \cdot \max_{p=1,\ldots,P} \left\{ \sum_{t=0}^{T} t \cdot S_{p,K_p,t} + J_{p,K_p} \right\} \\
& + \sum_{p=1}^{P} \sum_{k=1}^{K_p} \sum_{l=1}^{L} \sum_{m=1}^{M} \sum_{w=1}^{W} Q_{pkl} \cdot X_{ml} \cdot Y_{wl} \cdot \delta_{pkmw} \cdot \left(\widetilde{\alpha_m} + \widetilde{S_w} \right) \qquad (1)
\end{aligned}$$

$$\text{s.t.} \quad \sum_{m=1}^{M} X_{ml} = 1, \forall l \qquad (2)$$

$$\sum_{w=1}^{W} Y_{wl} = 1, \forall l \qquad (3)$$

$$\sum_{l=1}^{L} X_{ml} = \varpi_m, \forall m \qquad (4)$$

$$\sum_{l=1}^{L} Y_{wl} = \omega_w, \forall w \qquad (5)$$

$$\sum_{c=1}^{C} Z_{cl} = 1, \forall l \qquad (6)$$

$$\sum_{l=1}^{L} Q_{pkl} = 1, \forall p, k \qquad (7)$$

$$\sum_{l=1}^{L} Z_{cl} \le B_u, \forall c \qquad (8)$$

$$\sum_{l=1}^{L} Z_{cl} \ge B_d, \forall c \qquad (9)$$

$$\sum_{t=0}^{T} S_{pkt} = 1, \forall p, k \qquad (10)$$

$$\sum_{t=0}^{T} t \cdot S_{p,k+1,t} - \sum_{t=0}^{T} t \cdot S_{pkt} \ge J_{pk}, \forall p, \forall k = 1, 2, \ldots, K_p - 1 \qquad (11)$$

$$\sum_{p=1}^{P} \sum_{k=1}^{K_p} \sum_{\tau=\text{Max}\{0, t-J_{pk}\}}^{t} S_{pk\tau} \cdot Q_{pkl} \le 1, \forall l, t \qquad (12)$$

$$J_{pk} = \sum_{l=1}^{L} \sum_{m=1}^{M} \sum_{w=1}^{W} Q_{pkl} \cdot X_{ml} \cdot Y_{wl} \cdot \delta_{pkmw}, \forall p, k \qquad (13)$$

$$X_{ml}, Y_{wl}, Z_{cl}, Q_{pkl}, S_{pkt} \in \{0, 1\}, \forall m, w, l, c, p, k, t \qquad (14)$$

The objective function (1) consists of three terms defined as follows:

(The first term) Inter-cell material handling cost: The cost of part movement is incurred when the part in certain cell is transferred to a more efficient workstation in another cell for processing. The component $\left|\sum_{l}^{L} Q_{pkl} \cdot Z_{cl} - \sum_{l}^{L} Q_{p,k+1,l} \cdot Z_{cl}\right|$ checks whether the consecutive operations k and $k+1$ of part p are processed in the same cell c. For any operation k of part p, the component $\frac{1}{2} \cdot \sum_{c=1}^{C} \left|\sum_{l}^{L} Q_{pkl} \cdot Z_{cl} - \sum_{l}^{L} Q_{p,k+1,l} \cdot Z_{cl}\right|$ equaling 1 means that part p is transferred to another cell for operation $k+1$ after completing operation k. Inter-cell part movements complicate production control and decrease the efficiency of CMS, so it is better to locate two workstations with a small quantity of material flow in different cells, and to locate two workstations with a large quantity of material flow in the same cell.

(The second term) The fixed costs of machines and workers: The costs refer to the maintenance and overhead costs of machines as well as the base salary cost paid for workers. The component $\sum_{l=1}^{L} X_{ml}$ represents the number of machine type m, and $\sum_{l=1}^{L} Y_{wl}$ represents the number of worker type w. The component $\max\limits_{p=1,\ldots,P} \left\{\sum_{t=0}^{T} t \cdot S_{p,K_p,t} + J_{p,K_p}\right\}$ denotes the completion time of all parts (makespan) in the planning horizon.

(The third term) The operating costs of machines and workers: The costs are imposed only when the machines and workers are active. For instance, if machine type m and worker type w are assigned to workstation l at which operation k of part p is processed, the operating costs of the machine and worker will be incurred.

Constraints (2) and (3) ensure that each workstation can only receive one machine and one worker. Constraints (4) and (5) specify the number of each machine type and each worker type, respectively. Constraint (6) ensures that each workstation only belongs to one cell. Constraint (7) guarantees that any operation k of part p must be assigned to only one workstation to process. Constraints (8) and (9) specify the upper and lower bounds for the number of workstations allocated to each cell. Constraint (10) indicates that every operation k of part p must start once. Constraint (11) shows the precedence relationship between consecutive operations of part p. Constraint (12) ensures that each workstation l is to be occupied by at most one operation of the parts at a time. Constraint (13) defines an intermediate variable. Constraint (14) ensures that the decision variables are binary variables.

It is widely recognized in the literature that the whole problem of designing a CMS, taking into account the numerous phases and criteria involved, belongs to the class of NP-hard problems (Ballakur 1985; King and Nakornchai 1982). The proposed model has three simultaneous phases. The first phase is to determine the number and types of machines and workers assigned to workstations and cells. The second phase is to determine the processing routes from possibly multiple ones (machines and/or workers can perform more than one operation and come in multiple copies). Logendran, Ramakrishna, and Sriskandarajah (1994) have shown that the problem involved in this phase is NP-hard. The third phase is to determine the timetable of all parts. So the proposed model is NP-hard since it integrates the problem of cell formation, task scheduling along with the consideration of resources capability and operation sequence. This rules out the possibility of employing exhaustive exact algorithms such as branch-and-bound technique, backtracking or dynamic programming. Because these algorithms would be too time consuming even for a problem with a moderate number of parts, machines, workers and cells.

Since the proposed model has three simultaneous phases, the solution representation might be very complicated and should be designed as a multi-dimensional structure. Although many metaheuristic algorithms such as genetic algorithm and particle swarm optimisation can solve the model, the DBFA is more suitable for the multi-dimensional structure to fulfill a deeper exploration. The reason is that the DBFA possesses chemotactic capability, i.e. the bacterium (corresponding to solution representation) can tumble in an altogether different dimension and swim in a predefined dimension. Consequently, a high quality solution may be found during the exploration. In the next section, we will present a novel DBFA algorithm embedding Priority Rule Based Parallel Schedule Generation Scheme to solve this discrete problem.

4. Discrete bacteria foraging algorithm

The classical bacteria foraging algorithm was first invented based on the foraging strategy of *Escherichia coli* bacteria in human intestines, which can be explained by three processes, namely, chemotaxis (including swarming), reproduction, and elimination-dispersal (Passino 2002).

In BFA, through chemotaxis the bacteria try to search for nutrient alternating between ‘tumbling’ and ‘swimming’. If the bacterium happens to encounter a nutrient gradient (e.g. serine), the change in the concentration of the nutrient triggers a reaction such that the bacterium will spend more time swimming and less time tumbling. As long as it travels toward increasing nutrient concentrations, it will tend to swim farther, up to a point. The tumbles can change the direction via flagella rotating and basically determine the direction of the swim. The bacterium does not change its direction on a swim due to changes in the gradient. Through reproduction process the unhealthy bacteria die and each fitter bacterium splits into two bacteria. When the local environment where a population of bacteria live changes either gradually (e.g. via consumption of nutrients) or suddenly (e.g. due to medicine influence), the group of bacteria is dispersed to a new region or all bacteria in the area are eliminated.

The algorithm is suitable for continuous optimisation problem. In this paper a DBFA is suggested to solve the above discrete optimisation problem. In this DBFA, the solution representation and neighborhood generation operators are elaborately designed.

4.1 *Solution representation*

Since our problem is a discrete problem, we should move the bacteria discrete position to find a better solution in terms of the relationships of machine, worker, workstation, cell and task schedule. Thus, we suggest a schema which consists of five ingredients as follows:

The first ingredient, related to the assignment of duplicates of machine types to workstations, is named matrix Ma_Lo, and is shown in Equation (15). The components of this $M \times L$ matrix are variables X_{ml} which are either 0 or 1. For example, the term $X_{34} = 1$ means that one duplicate of machine type 3 is assigned to workstation 4. While completing the matrix, constraints (2) and (4) should be satisfied.

$$Ma\_Lo = \begin{bmatrix} X_{11} & X_{12} & \ldots & X_{1L} \\ X_{21} & X_{22} & \ldots & X_{2L} \\ \vdots & \vdots & & \vdots \\ X_{M1} & X_{M2} & \ldots & X_{ML} \end{bmatrix} \tag{15}$$

The second ingredient, related to the assignment of duplicates of worker types to workstations, is named matrix Wo_Lo, and is shown in Equation (16). The components of this $W \times L$ matrix are variables Y_{wl} which are either 0 or 1. For example, the term $Y_{23} = 1$ means that one duplicate of worker type 2 is assigned to workstation 3. While completing the matrix, constraints (3) and (5) should be satisfied.

$$Wo\_Lo = \begin{bmatrix} Y_{11} & Y_{12} & \ldots & Y_{1L} \\ Y_{21} & Y_{22} & \ldots & Y_{2L} \\ \vdots & \vdots & & \vdots \\ Y_{W1} & Y_{W2} & \ldots & Y_{WL} \end{bmatrix} \tag{16}$$

The third ingredient, related to the assignment of workstations to cells, is named matrix Ce_Lo, and is shown in Equation (17). The components of this $C \times L$ matrix are variables Z_{cl} which are either 0 or 1. For example, the term $Z_{24} = 1$ means that workstation 4 is assigned to cell 2. While completing the matrix, constraints (6), (8) and (9) should be satisfied.

$$Ce\_Lo = \begin{bmatrix} Z_{11} & Z_{12} & \ldots & Z_{1L} \\ Z_{21} & Z_{22} & \ldots & Z_{2L} \\ \vdots & \vdots & & \vdots \\ Z_{C1} & Z_{C2} & \ldots & Z_{CL} \end{bmatrix} \tag{17}$$

The fourth ingredient, related to the assignment of part operations to workstations, is named matrix Pa_Op_Lo, and is shown in Equation (18). The components of this $P \times K \times L$ matrix are variables Q_{pkl} which are either 0 or 1. For example, the term $Q_{345} = 1$ means that the operation 4 of part 3 is assigned to workstation 5 for processing. While completing the matrix, constraints (7) should be satisfied.

$$Pa\_Op\_Lo = \begin{bmatrix} Q_{1K1} & \ldots & Q_{1KL} & & & \\ \vdots & \ddots & & \ddots & & \\ Q_{PK1} & & & Q_{111} & \ldots & Q_{11L} \\ & \ddots & & \vdots & & \vdots \\ & & & Q_{P11} & \ldots & Q_{P1L} \end{bmatrix} \tag{18}$$

The fifth ingredient is named matrix ST_Op and/or FT_Op, and are shown in Equation (19). The components of the matrices are variables ST_{pk} and/or FT_{pk} which are non-negative integers representing the start and/or the finish time of the part operations. These variables can be computed by *Algorithm* 1 (Priority Rule Based Parallel Schedule Generation Scheme). While implementing this algorithm, constraints (10)–(12) should be satisfied.

$$ST\_Op = \begin{bmatrix} ST_{11} & ST_{12} & \dots & ST_{1K} \\ ST_{21} & ST_{22} & \dots & ST_{2K} \\ \vdots & \vdots & & \vdots \\ ST_{P1} & ST_{P2} & \dots & ST_{PK} \end{bmatrix}, \qquad FT\_Op = \begin{bmatrix} FT_{11} & FT_{12} & \dots & FT_{1K} \\ FT_{21} & FT_{22} & \dots & FT_{2K} \\ \vdots & \vdots & & \vdots \\ FT_{P1} & FT_{P2} & \dots & FT_{PK} \end{bmatrix} \tag{19}$$

Combining the five ingredients described above, the solution representation is as shown in (20). The ingredients can be referred as dimensions or directions, so it is very suitable for the bacterium of DBFA to tumble in a randomly selected dimension and to swim in the same previous dimension.

$$(Ma\_Lo \mid Wo\_Lo \mid Ce\_Lo \mid Pa\_Op\_Lo \mid ST\_Op) \tag{20}$$

The main variables used for *Algorithm* 1 are summarized as follows:

N Sum of the operations of all parts, $N = \sum_{p=1}^{P} K_p$
u Iteration counter.
t Schedule time point.
I_l Start idle time of workstation l.
$\hbar$ Completion operation-set up to the schedule time.
D Decision operation-set for the associated workstation, i.e. the operations in the set are available for scheduling at the associated workstation w.r.t. precedence constraints but have yet unscheduled.
H Scheduled operation-set at each iteration.
V 0 if all decision operation-sets for the associated workstations are empty at the schedule time point, otherwise 1.

Algorithm 1: Priority Rule Based Parallel Schedule Generation Scheme

```
1. Initialize: $u = 0$; $I_l = 0, \forall l$; $\hbar = H = \emptyset$
2. While (1) Do
        2.1. $T = \{I_l | l = 1, 2, \dots, L\}$
        2.2. $V = 0$
        2.3. While (1) Do
                2.3.1. $t = \min\{\tau | \tau \in T\}$
                2.3.2. $\{l^*\} : I_{l^*} \leqslant t$
                2.3.3. $\hbar = \{k_p | FT_{pk} \leq t, k_p \in H\}$
                2.3.4. For each $l^*$
                        2.3.4.1. $D = \{k_p | Q_{pkl^*} = 1, k_p \notin H, (k-1)_p \in \hbar\}$
                        2.3.4.2. If $(D \neq \emptyset)$
                                        $V = 1$
                                        Randomly select a $k^*_{p^*}$ from $\{k^\wedge_{p^\wedge}\} : J_{p^\wedge k^\wedge} = \min\{J_{pk} | k_p \in D\}$
                                        $ST_{p^*k^*} = t$, $FT_{p^*k^*} = t + J_{p^*k^*}$
                                        $I_{l^*} = FT_{p^*k^*}$
                                        $H := H \cup \{k^*_{p^*}\}$
                                        $u := u + 1$
                                        If $(u = N)$ then stop the procedure
                2.3.5. If $(V = 0)$
                                $T := T \backslash \{t\}$
                        Else
                                Break
```

Algorithm 1 consists of N iterations. At each iteration, an eligible operation is selected according to its priority and inserted inside a partial schedule on the earliest eligible workstation (respecting the precedence constraints), while keeping the start time of the already scheduled operations unchanged. An operation is eligible if its predecessor has already been scheduled. A workstation is eligible if it is idle from the start of the schedule time point t.

Step 1 initializes some variables u, I_l, $\hbar$, H. Step 2.1 computes the set (T) of start idle times for all workstations. The binary variable V is initialized in Step 2.2. The scheduled time point t is calculated in Step 2.3.1. All idle workstations at schedule time t are shown in Step 2.3.2. The completion operation-set $\hbar$ up to the scheduled time t are calculated in Step 2.3.3. In Step 2.3.4, for each idle workstation, the procedure randomly schedules an operation which is the minimum in terms of the processing time in the associated decision operation-set. Then the start and finish times of the operation are calculated. Moreover, the idle time of the workstation, the scheduled operation-set and the iteration counter are updated.

$$Ma\_Lo=\begin{bmatrix}1\,0\,0\,1\\0\,1\,1\,0\end{bmatrix},\qquad Wo\_Lo=\begin{bmatrix}1\,0\,0\,0\\0\,1\,1\,1\end{bmatrix}$$

$$Ce\_Lo=\begin{bmatrix}0\,0\,0\,1\\1\,1\,1\,0\end{bmatrix}$$

$$Pa\_Op\_Lo=\begin{array}{|c|c|c|}\hline p & k & l \\ & & 1\,2\,3\,4 \\ \hline 1 & 1 & 0\,0\,1\,0 \\ & 2 & 0\,1\,0\,0 \\ & 3 & 0\,0\,1\,0 \\ \hline 2 & 1 & 0\,0\,0\,1 \\ \hline 3 & 1 & 0\,0\,0\,1 \\ & 2 & 0\,0\,0\,1 \\ \hline 4 & 1 & 1\,0\,0\,0 \\ \hline 5 & 1 & 0\,0\,0\,1 \\ & 2 & 1\,0\,0\,0 \\ & 3 & 0\,0\,1\,0 \\ \hline\end{array}\left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\end{array}}\right\}Q_{pkl}$$

J_{11}	J_{12}	J_{13}	J_{21}	J_{31}	J_{32}	J_{41}	J_{51}	J_{52}	J_{53}
21	24	30	29	23	22	30	22	24	20

Figure 1. Parameters of the instance using PRBPSGS.

$$ST\_Op=\begin{bmatrix}0 & 21 & 45\\67 & - & -\\22 & 45 & -\\0 & - & -\\0 & 30 & 75\end{bmatrix}\qquad FT\_Op=\begin{bmatrix}21 & 45 & 75\\96 & - & -\\45 & 67 & -\\30 & - & -\\22 & 54 & 95\end{bmatrix}$$

Figure 2. Solution of the instance using PRBPSGS.

Step 2.3.5 implies that the scheduled time point will be postponed to the next minimum start idle time for all workstations if all decision operation-sets for the associated workstations are empty at the scheduled time point. The procedure will stop when all operations are scheduled.

To better illustrate the proposed PRBPSGS, let us consider a simple instance. Because the PRBPSGS is used to compute the fifth ingredient of the solution, the first four ingredients and the processing time of operations should be given before the PRBPSGS starts. Figure 1 shows these given parameters of this instance. The solution is reached after ten iterations using the PRBPSGS. Table 1 shows the computational process of each iteration.

For example, during the third iteration $u = 4$, schedule time point $t = 0$ is determined in Step 2.3.1. For idle workstation $l^* = 4$, decision operation-set $D = \{(2, 1), (3, 1), (5, 1)\}$ is computed in Step 2.3.4.1. In Step 2.3.4.2, operation $k^* = 1$ of part $p^* = 5$ is selected for processing, started at time $ST_{p^*k^*} = 0$, and completed at time $FT_{p^*k^*} = 22$. The other iterations are similar to the third iteration. The final solution is presented by Figure 2.

Table 1. Summary of iterations of the instance using PRBPSGS.

t	l^*	$D = \{(p,k)\}$	(p^*, k^*)	$ST_{p^*k^*}$	$FT_{p^*k^*}$	u
0	1	{(4,1)}	(4, 1)	0	30	1
0	3	{(1,1)}	(1, 1)	0	21	2
0	4	{(2,1),(3,1),(5,1)}	(5, 1)	0	22	3
21	2	{(1,2)}	(1, 2)	21	45	4
22	4	{(2,1),(3,1)}	(3, 1)	22	45	5
30	1	{(5,2)}	(5, 2)	30	54	6
45	3	{(1,3)}	(1, 3)	45	75	7
45	4	{(2,1),(3,2)}	(3, 2)	45	67	8
67	4	{(2,1)}	(2, 1)	67	96	9
75	3	{(5,3)}	(5, 3)	75	95	10

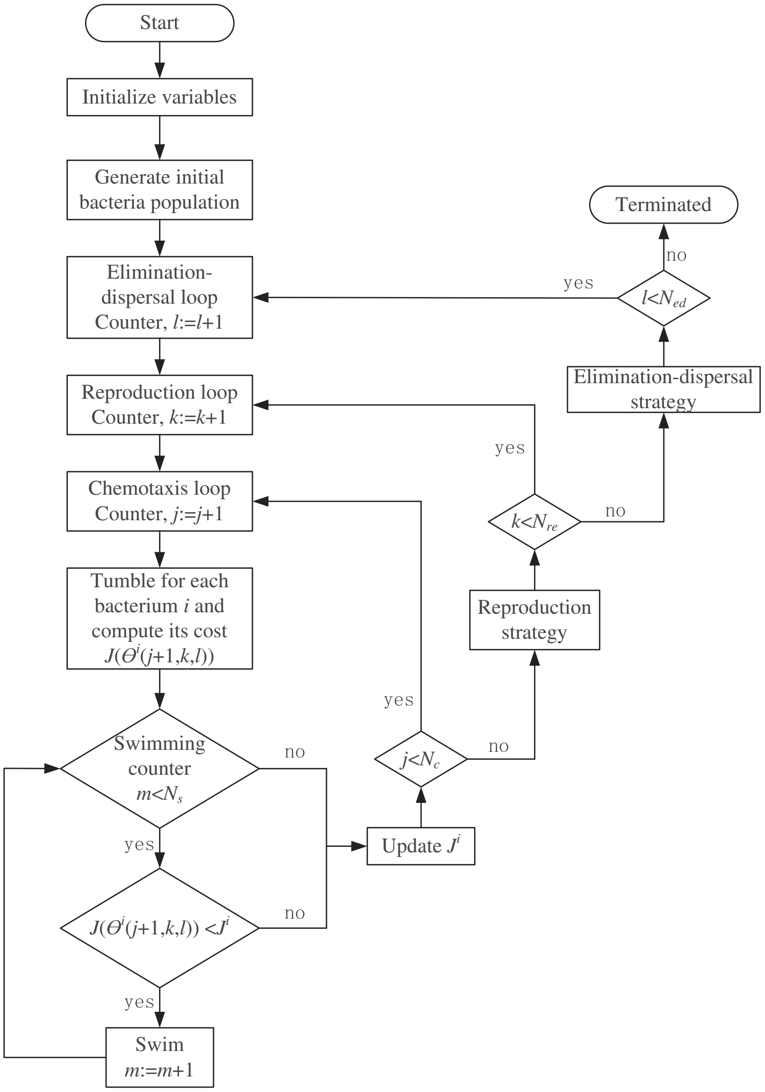
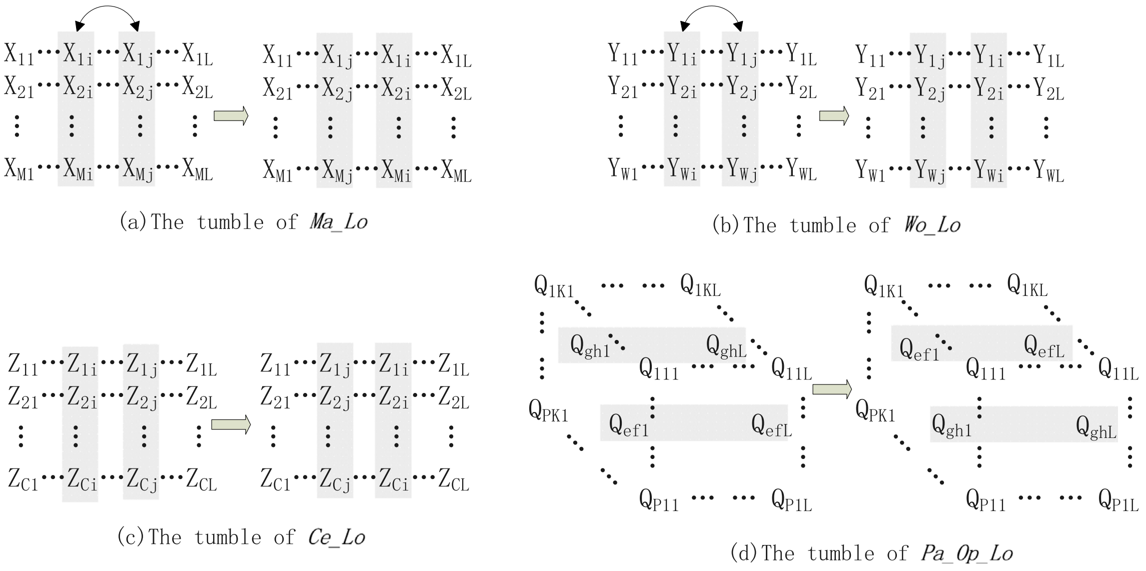

Start
Initialize variables
Generate initial bacteria population
Elimination-dispersal loop Counter, $l{:=}l{+}1$
Reproduction loop Counter, $k{:=}k{+}1$
Chemotaxis loop Counter, $j{:=}j{+}1$
Tumble for each bacterium i and compute its cost $J(\Theta^i(j{+}1,k,l))$
Swimming counter $m<N_s$
yes
no
$J(\Theta^i(j{+}1,k,l)) < J^i$
yes
no
Swim $m{:=}m{+}1$
Update J^i
$j<N_c$
yes
no
Reproduction strategy
$k<N_{re}$
yes
no
Elimination-dispersal strategy
$l<N_{ed}$
yes
no
Terminated

Figure 3. Flowchart of proposed DBFA algorithm.



(a)The tumble of *Ma_Lo*

(b)The tumble of *Wo_Lo*

(c)The tumble of *Ce_Lo*

(d)The tumble of *Pa_Op_Lo*

Figure 4. Tumble of bacterium.

4.2 *Implementation of the proposed DBFA*

Some notations to be used are summarized as follows:

j Index for the chemotactic step.
k Index for the reproduction step.
l Index of the elimination-dispersal event.
S Number of bacteria in a population (the number is assumed to be a positive even integer).
S_r Number of half population of bacteria.
$\theta^i(j,k,l)$ The ith bacterium position at the jth chemotactic step, kth reproduction step, and lth elimination-dispersal event, which corresponds to a feasible solution of the problem, $\theta^i \in R^p$, where p is the number of dimensions of the position (sometimes we drop the indices and refer to the ith bacterium position as θ^i).
$J(\theta^i)$ The 'cost' of being in the position θ^i (using terminology from optimisation theory) or the nutrient surface (in reference to the biological connections), which corresponds to the objective function value of the solution.
N_c Length of the lifetime of the bacteria as measured by the number of chemotactic steps they take during their life.
m Counter for swimming steps.
N_s Maximum number of swimming steps.
N_{re} Number of reproduction steps.
N_{ed} Number of elimination-dispersal events.
p_{ed} Probability of elimination-dispersal event for each bacterium.
J^i_{cur} The best cost of each bacterium i in the current generation.
J^i_{last} The last cost of each bacterium i (a better cost may be found via a run).
J^i_{health} Accumulated cost of all the chemotactic steps of bacterium i in the current generation; $J^i_{health} = \sum_{j=1}^{N_c+1} J(i,j,k,l)$.

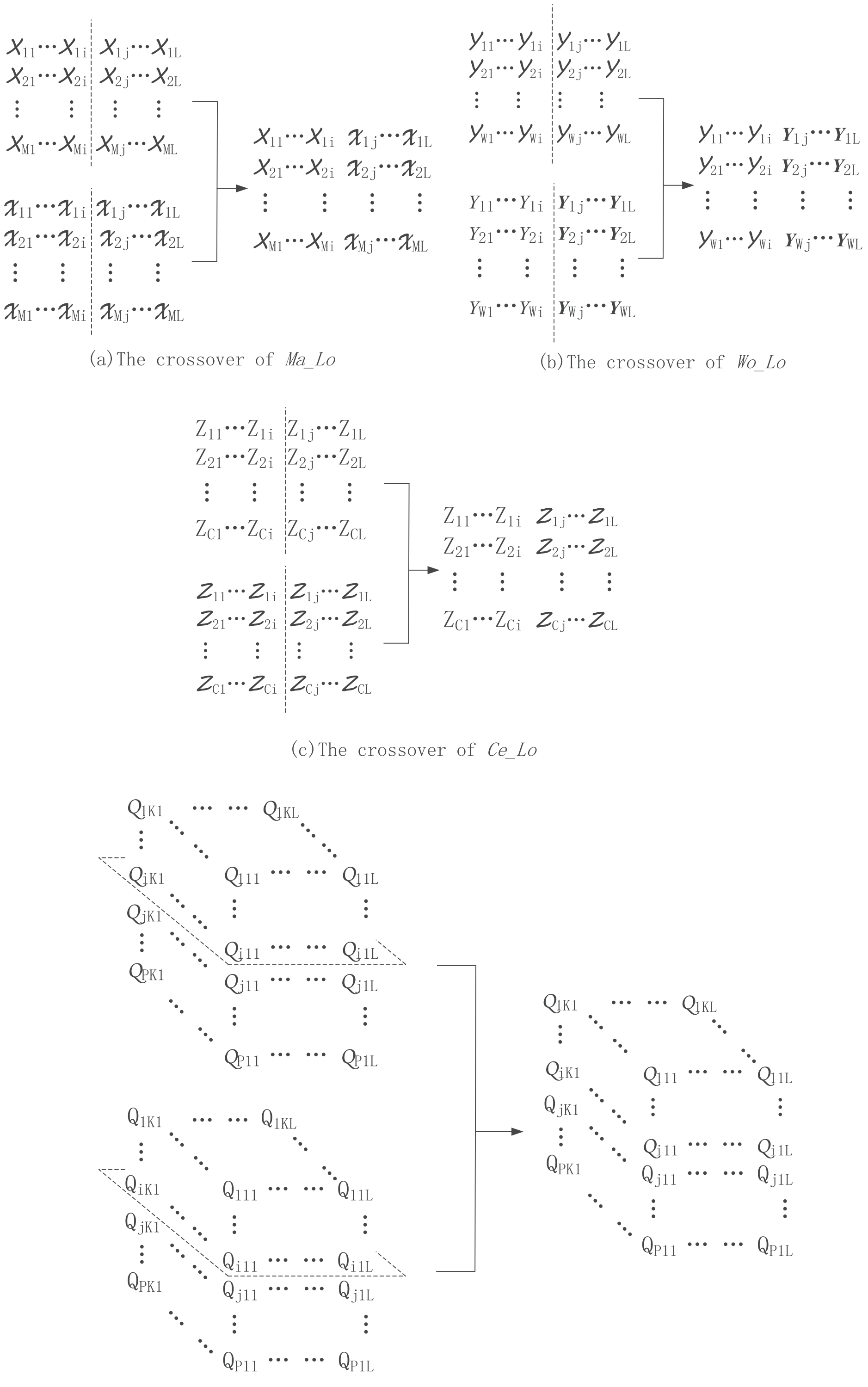
The DBFA simulates the foraging behavior of bacteria which tries to climb up the nutrient concentration (finding lower and lower values of $J(\theta^i)$), avoid the noxious substances, and search for ways out of the neutral media through three nested loops of chemotaxis, reproduction, and elimination-dispersal. For example, $J(\theta^i) < 0$, $J(\theta^i) = 0$, and $J(\theta^i) > 0$ represent that the bacterium i at position θ^i is in nutrient-rich, neutral, and noxious environments, respectively. The bacterium tends to avoid being at positions θ^i, where $J(\theta^i) \geq 0$. The flowchart of the proposed DBFA algorithm is outlined in Figure 3, and its detailed procedure is described in *Algorithm 2*. We refer to the DBFA with Passino's original operators (including swimming strategy, reproduction strategy and elimination-dispersal strategy) as DBFA1, and refer to the DBFA with the modified operators as DBFA2.

Algorithm 2: Discrete Bacteria Foraging Algorithm (including DBFA1 and DBFA2)

```
1. Initialize: $S = 10$, $N_c = 2$, $N_s = 10$, $N_{re} = 10$, $N_{ed} = 6$, $p_{ed} = 0.3$, $j = k = l = 0$;
2. Randomly generate initial values for the $\theta^i$, $i = 1, 2, \ldots S$
3. Elimination-dispersal loop: $l := l + 1$
    3.1. Reproduction loop: $k := k + 1$
        3.1.1. Let $J^i_{cur} = J(\theta^i(j, k, l))$ to save the best cost of each bacterium $i$ in the current
               generation.
        3.1.2. Chemotaxis loop: $j := j + 1$
        3.1.3. For $i = 1, 2, \ldots, S$, take a chemotactic step for bacterium $i$ as follows.
            3.1.3.1. Let $J^i_{last} = J(\theta^i(j, k, l))$ to save this value since we may find a better cost via
                     a run.
            3.1.3.2. Tumble: movement (mutation) of bacterium $i$ in the $d$th dimensional
                     direction ($d$ is randomly selected), which results in new position
                     $\theta^i(j + 1, k, l)$ for bacterium $i$.
            3.1.3.3. Compute $J(\theta^i(j + 1, k, l))$.
            3.1.3.4. Swimming process of bacterium $i$ if needed:
                     Let $m$=0 (counter for swimming steps).
                     While ($m < N_s$)
                         If ($J(\theta^i(j + 1, k, l)) < J^i$) ✻
                             Let $J^i = J(\theta^i(j + 1, k, l))$.
                             Swim in the above $d$th dimensional direction, and update $\theta^i(j + 1, k, l)$.
                             Compute $J(\theta^i(j + 1, k, l))$.
                             Let $m := m + 1$.
                         Else
                             Break
            3.1.3.5. If($J^i_{last} < J^i_{cur}$)
                         $J^i_{cur} = J^i_{last}$
            3.1.3.6. If($J(\theta^i(j + 1, k, l)) < J^i_{cur}$)
                         $J^i_{cur} = J(\theta^i(j + 1, k, l))$
        3.1.4. If $j < N_c$, go to step 3.1.2. In this case, continue chemotaxis, since the life of the
               bacteria is not over.
    3.2. Implement reproduction strategy.★
    3.3. If $k < N_{re}$, go to step 3.1 (i.e. start the next generation in the chemotactic loop).
4. Elimination-dispersal strategy ♣.
5. If $l < N_{ed}$, then go to step 3; otherwise end.
```

A chemotactic step is defined to be a tumble or a tumble followed by at most N_s swimming steps. Tumble refers to the movement (mutation) of a bacterium in a random direction displayed in Figure 4. For example, Figure 4(a) shows a bacterium runs a tumble in the first dimension, i.e. two columns of matrix Ma_Lo are selected randomly and their components are substituted. Similarly, Figure 4(b)–(d) show a bacterium runs a tumble in the second, third and fourth dimension, respectively. Swim indicates that when the 'cost' of the above bacterium is better than its specific 'cost' J (which is referred to its last 'cost' J_{last} in Passino's swimming strategy, or its best historic 'cost' in the current generation J_{cur} in this paper), the bacterium will move (mutate) further in the same direction. If that movement resulted in a better position according to the above comparison rules, another movement is taken. This swim is continued as long as it continues to reduce the cost, but only up to a maximum number of movements, N_s. This represents that the bacterium will tend to keep moving if it is headed in the direction of increasingly favourable environments. The two swimming rules above are employed in the DBFA1 and DBFA2, respectively.

After N_c chemotactic steps, a reproduction step is taken. In Passino's reproduction strategy (Passino 2002), the population is sorted in order of ascending accumulated cost J_{health} of all the chemotactic steps of each bacterium in the current generation. The $S_r = S/2$ bacteria with the highest J_{health} values die and the other $S/2$ bacteria with the lowest values split (each bacteria split into two bacteria). Each copy made is placed at the same position as its parent. In this paper's reproduction strategy, the population is sorted in order of ascending historic 'cost' J_{cur} of each bacterium in the current generation. The S_r bacteria with the highest J_{cur} values die. Each of the other S_r bacteria with the lowest values moves to its best position in the current



(a) The crossover of *Ma_Lo*

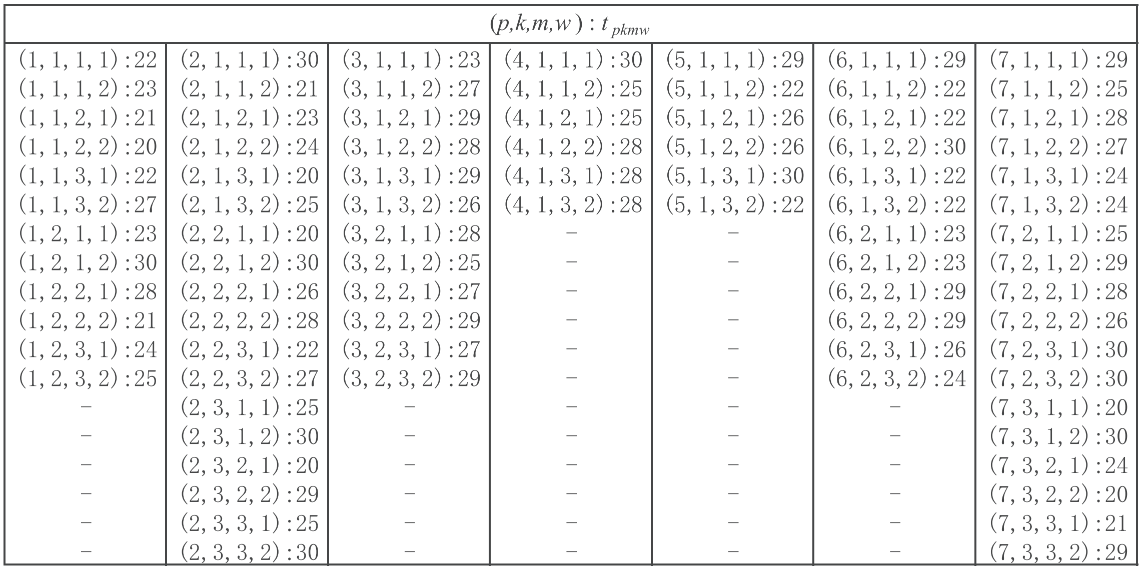
(b) The crossover of *Wo_Lo*

(c) The crossover of *Ce_Lo*

(d) The crossover of *Pa_Op_Lo*

Figure 5. Crossover of bacteria.

generation (associated with the value J_{cur}). Randomly select two of them to cross over (cf. Algorithm 3 and Figure 5) and put the offspring bacterium into the current generation until the population size reaches S. The two reproduction strategies above are employed in the DBFA1 and DBFA2, respectively.



$(p,k,m,w) : t_{pkmw}$						
(1, 1, 1, 1) :22	(2, 1, 1, 1) :30	(3, 1, 1, 1) :23	(4, 1, 1, 1) :30	(5, 1, 1, 1) :29	(6, 1, 1, 1) :29	(7, 1, 1, 1) :29
(1, 1, 1, 2) :23	(2, 1, 1, 2) :21	(3, 1, 1, 2) :27	(4, 1, 1, 2) :25	(5, 1, 1, 2) :22	(6, 1, 1, 2) :22	(7, 1, 1, 2) :25
(1, 1, 2, 1) :21	(2, 1, 2, 1) :23	(3, 1, 2, 1) :29	(4, 1, 2, 1) :25	(5, 1, 2, 1) :26	(6, 1, 2, 1) :22	(7, 1, 2, 1) :28
(1, 1, 2, 2) :20	(2, 1, 2, 2) :24	(3, 1, 2, 2) :28	(4, 1, 2, 2) :28	(5, 1, 2, 2) :26	(6, 1, 2, 2) :30	(7, 1, 2, 2) :27
(1, 1, 3, 1) :22	(2, 1, 3, 1) :20	(3, 1, 3, 1) :29	(4, 1, 3, 1) :28	(5, 1, 3, 1) :30	(6, 1, 3, 1) :22	(7, 1, 3, 1) :24
(1, 1, 3, 2) :27	(2, 1, 3, 2) :25	(3, 1, 3, 2) :26	(4, 1, 3, 2) :28	(5, 1, 3, 2) :22	(6, 1, 3, 2) :22	(7, 1, 3, 2) :24
(1, 2, 1, 1) :23	(2, 2, 1, 1) :20	(3, 2, 1, 1) :28	–	–	(6, 2, 1, 1) :23	(7, 2, 1, 1) :25
(1, 2, 1, 2) :30	(2, 2, 1, 2) :30	(3, 2, 1, 2) :25	–	–	(6, 2, 1, 2) :23	(7, 2, 1, 2) :29
(1, 2, 2, 1) :28	(2, 2, 2, 1) :26	(3, 2, 2, 1) :27	–	–	(6, 2, 2, 1) :29	(7, 2, 2, 1) :28
(1, 2, 2, 2) :21	(2, 2, 2, 2) :28	(3, 2, 2, 2) :29	–	–	(6, 2, 2, 2) :29	(7, 2, 2, 2) :26
(1, 2, 3, 1) :24	(2, 2, 3, 1) :22	(3, 2, 3, 1) :27	–	–	(6, 2, 3, 1) :26	(7, 2, 3, 1) :30
(1, 2, 3, 2) :25	(2, 2, 3, 2) :27	(3, 2, 3, 2) :29	–	–	(6, 2, 3, 2) :24	(7, 2, 3, 2) :30
–	(2, 3, 1, 1) :25	–	–	–	–	(7, 3, 1, 1) :20
–	(2, 3, 1, 2) :30	–	–	–	–	(7, 3, 1, 2) :30
–	(2, 3, 2, 1) :20	–	–	–	–	(7, 3, 2, 1) :24
–	(2, 3, 2, 2) :29	–	–	–	–	(7, 3, 2, 2) :20
–	(2, 3, 3, 1) :25	–	–	–	–	(7, 3, 3, 1) :21
–	(2, 3, 3, 2) :30	–	–	–	–	(7, 3, 3, 2) :29

Figure 6. Processing time of operations in the case using DBFA2.

After N_{re} reproduction steps, an elimination-dispersal event takes place. Each bacterium in the population is subjected to elimination-dispersal with probability p_{ed}. The elimination-dispersal events allow the bacteria to look into more regions to find good nutrient concentrations. Obviously, if p_{ed} is chosen appropriately, the elimination-dispersal events can help the DBFA jump out of the local optima and into a global optimum. However, if p_{ed} is large, the DBFA will degrade to random exhaustive search. In Passino's elimination-dispersal strategy (shown in DBFA1), the best bacterium may be subject to elimination-dispersal. In this paper, the DBFA2 suggests that each bacterium, except the best one, is to be dispersed to a random position on the optimisation domain with probability p_{ed}. This keeps the best bacterium remaining in the next generation, which may speed up the convergence.

Algorithm 3: Crossover Operator

Step 1. (cf. Figure 5(a)): Randomly select the same position (dash line) of Ma_Lo in two bacteria. The left part of the dash line in the first Ma_Lo and the right part of the dash line in the second Ma_Lo are combined into a new Ma_Lo of offspring bacterium. If the sum of each row of offspring Ma_Lo is equal to one of the corresponding row of parent Ma_Lo, complete the combination to generate a feasible offspring Ma_Lo; otherwise, randomly select the other
position (dash line). If there is no feasible offspring Ma_Lo for all positions, select either of parent Ma_Lo as offspring Ma_Lo.
Step 2. (cf. Figure 5(b)): Generate a feasible offspring Wo_Lo as in Step 1.
Step 3. (cf. Figure 5(c)): Randomly select the same position (dash line) of Ce_Lo in two bacteria. The left part of the dash line in the first Ce_Lo and the right part of the dash line in the second Ce_Lo are combined into a new Ce_Lo of offspring bacterium. If the sum of each row of offspring Ce_Lo is between the lower and upper bounds of cell size, complete the combination to generate a feasible offspring Ce_Lo; otherwise, randomly select the other position (dash line). If there is no feasible offspring Ce_Lo for all positions, select either of parent Ce_Lo as offspring Ce_Lo.
Step 4. (cf. Figure 5(d)): Randomly select the same position (dash plane) of Pa_Op_Lo in two bacteria. The upper part of the dash line in the first Pa_Op_Lo and the lower part of the dash line in the second Pa_Op_Lo are combined to generate a feasible offspring Pa_Op_Lo.
Step 5. Apply Algorithm 1 to compute the fifth ingredient of the offspring bacterium.

A typical case is given to better demonstrate the optimisation problem and the result using the DBFA2. The parameters of the case are listed in Table 2. The processing time of operation k of part p on machine type m with worker type w is listed in Figure 6. The solution of the case is shown in Figure 7. It can also be displayed visually by the Gantt chart in Figure 8.

$$Ma\_Lo = \begin{bmatrix} 1\,1\,1\,0\,0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1\,0\,1\,0\,0 \\ 0\,0\,0\,1\,0\,0\,0\,0\,1 \end{bmatrix}, \quad Wo\_Lo = \begin{bmatrix} 0\,1\,1\,0\,0\,0\,1\,1\,0 \\ 1\,0\,0\,1\,1\,1\,0\,0\,1 \end{bmatrix}$$

$$Ce\_Lo = \begin{bmatrix} 0\,1\,1\,1\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,1\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1\,1\,0\,1 \end{bmatrix}$$

Pa_Op_Lo =

p	*k*	*l*
		1 2 3 4 5 6 7 8 9
1	1	0 0 0 0 1 0 0 0 0
	2	0 0 0 0 1 0 0 0 0
2	1	0 0 0 0 0 1 0 0 0
	2	0 0 1 0 0 0 0 0 0
	3	0 0 0 0 0 0 1 0 0
3	1	0 0 1 0 0 0 0 0 0
	2	0 0 0 0 0 1 0 0 0
4	1	0 0 0 0 0 0 0 0 1
5	1	0 0 0 1 0 0 0 0 0
6	1	0 0 0 1 0 0 0 0 0
	2	0 0 0 1 0 0 0 0 0
7	1	0 0 0 0 0 0 0 0 1
	2	0 0 0 0 0 0 0 1 0
	3	0 0 0 0 1 0 0 0 0

Q_{pkl}

$$ST\_Op = \begin{bmatrix} 0 & 20 & - \\ 0 & 23 & 43 \\ 0 & 23 & - \\ 24 & - & - \\ 0 & - & - \\ 22 & 44 & - \\ 0 & 24 & 49 \end{bmatrix}, \qquad FT\_Op = \begin{bmatrix} 20 & 41 & - \\ 21 & 43 & 63 \\ 23 & 48 & - \\ 52 & - & - \\ 22 & - & - \\ 44 & 68 & - \\ 24 & 49 & 69 \end{bmatrix}$$

Figure 7. Solution of the case using DBFA2.

Table 2. Parameters of the case using DBFA2.

Number of each machine type: $\varpi_1 = 5$, $\varpi_2 = 2$, $\varpi_3 = 2$
Number of each worker type: $\omega_1 = 4$, $\omega_2 = 5$
Number of parts: $P = 7$
Number of operations of each part: $K_1 = 2$, $K_2 = 3$, $K_3 = 2$, $K_4 = 1$, $K_5 = 1$, $K_6 = 2$, $K_7 = 3$
Lower and upper bound of each cell size: $B_d = 2$, $B_u = 4$
Fixed costs per unit of machine type: $\alpha_1 = 15$, $\alpha_2 = 14$, $\alpha_3 = 17$
Operating cost per unit of machine type: $\widetilde{\alpha_1} = 19$, $\widetilde{\alpha_2} = 13$, $\widetilde{\alpha_3} = 17$
Base salary cost per unit of worker type: $S_1 = 13$, $S_2 = 15$
Operating salary cost per unit of worker type: $\widetilde{S_1} = 22$, $\widetilde{S_2} = 10$
Inter-cell material handling cost of each part: $\theta_1 = 6$, $\theta_2 = 5$, $\theta_3 = 5$, $\theta_4 = 9$, $\theta_5 = 8$, $\theta_6 = 8$, $\theta_7 = 10$

The legend represents that operation k of part p is processed on workstation l which is grouped by machine type m and w. Workstation l is assigned to cell c. The operation k of part p is started at ST_{pk} and completed at FT_{pk}.

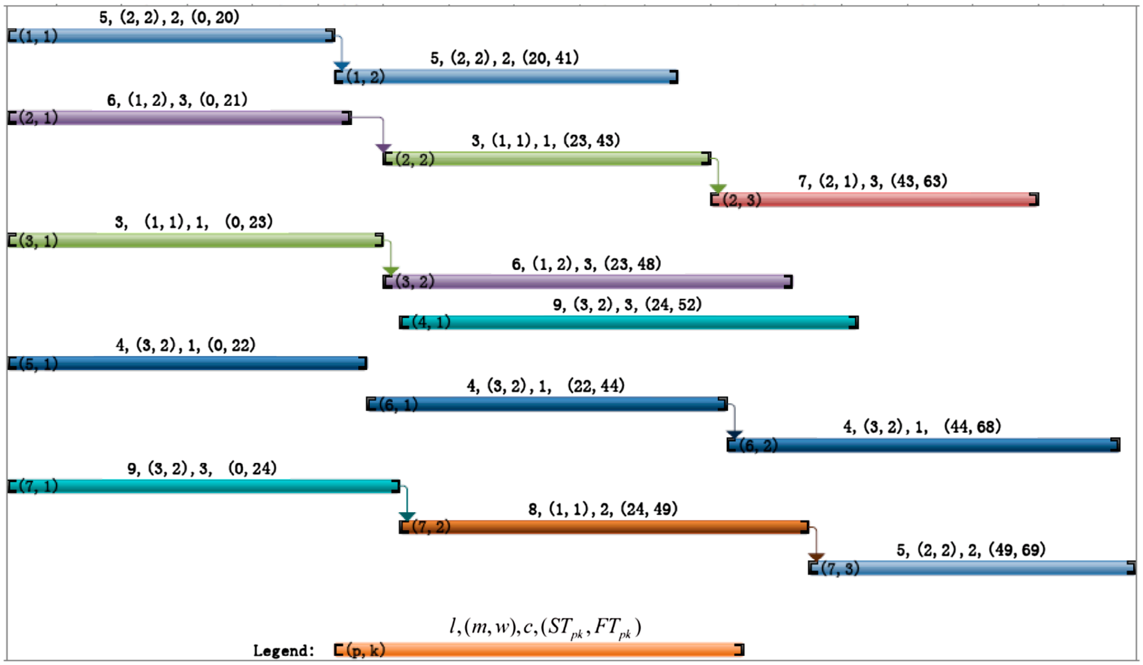


Figure 8. Gantt chart of the solution of the case using DBFA2.

5. Genetic algorithm

Genetic algorithm (GA) is a powerful and broadly applicable stochastic search and optimisation technique based on the principles of evolution theory. Here, a conventional GA is suggested for solving the problem, so that it can be compared with the proposed DBFA in the following experiment.

- Chromosome representation: The chromosome representation employs the same solution structure of the DBFA. This helps to exclude the influence of different solution structures when comparing the DBFA with the GA.
- Fitness function: Let F_g^k denote the fitness value of the kth chromosome in generation g before selection. It is computed as $F_g^k = \xi + \max_{i \in \{1,\ldots,E\}} \psi_g^i - \psi_g^k$, where ψ_g^i is the objective function value of the ith chromosome in generation g, E is the number of chromosomes in generation g before selection, and ξ is a small constant (say 3). Obviously, the smaller the objective function value of chromosome, the greater its fitness value.
- Initial population: Randomly generate $E = 10$ chromosomes to form the initial population.
- Crossover: All chromosomes in the parent generation are mutually crossed over. The crossover mechanism is similar to Algorithm 3. If the fitness value of offspring is greater than the average fitness value of its parent generation, the offspring will be accepted for the new generation; otherwise, it will be thrown out.
- Mutation: Each chromosome will mutate according to a given mutation probability $P_m = 0.1$. The mutation mechanism is to make the first four ingredients to tumble similar to Figure 4. The last ingredient is then computed using Algorithm 1. All offsprings from the mutation are accepted for the new generation.
- Selection: The most common method 'roulette wheel' sampling is applied in the selection. Each chromosome is assigned a slice of the circular roulette wheel and the size of the slice is proportional to the selection value (i.e. fitness value) of the chromosome. The wheel is spun E times. On each spin, the chromosome under the wheel's marker is selected to be in the pool of parents for the next generation.
- Stopping rule: The GA is stopped when its runtime reaches the DBFA2's.

6. Computational experiments

We compare the performance of DBFA1, DBFA2 and GA through the following numerical experiments. To compare the solution quality of the DBFA1 with the DBFA2 within the same CPU time, we modify the stopping rule of the DBFA1. Let N_{ed} of DBFA1 be a sufficient large integer. The DBFA1 is terminated if its runtime reaches the DBFA2's runtime after reproduction.

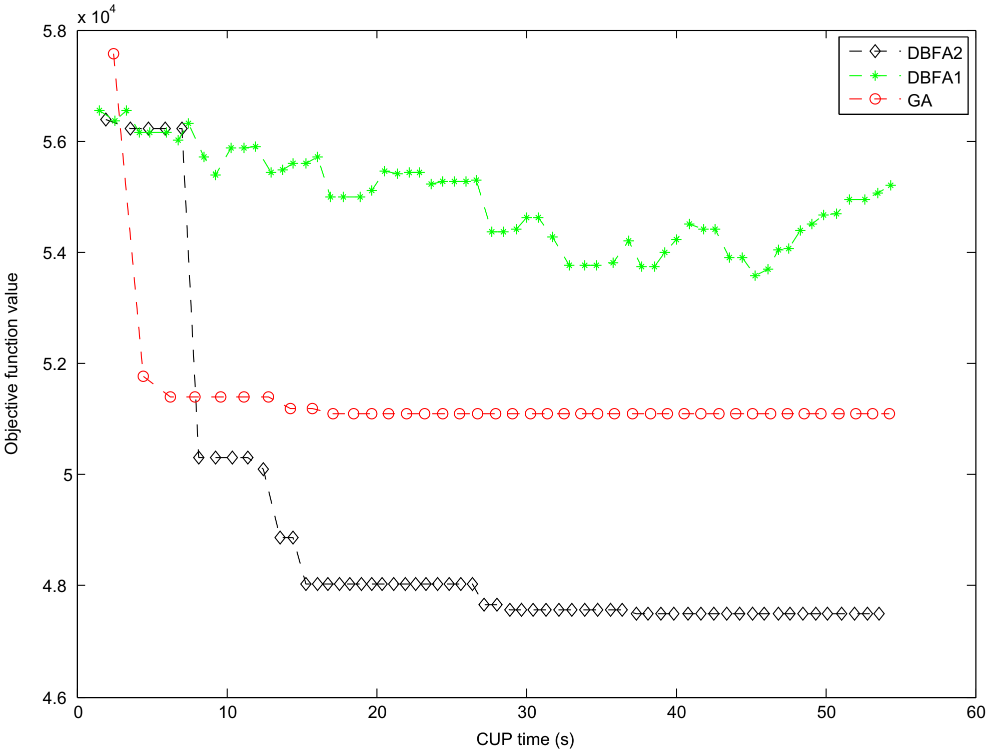


Figure 9. Typical convergence of DBFA1, DBFA2 and GA within the same runtime.

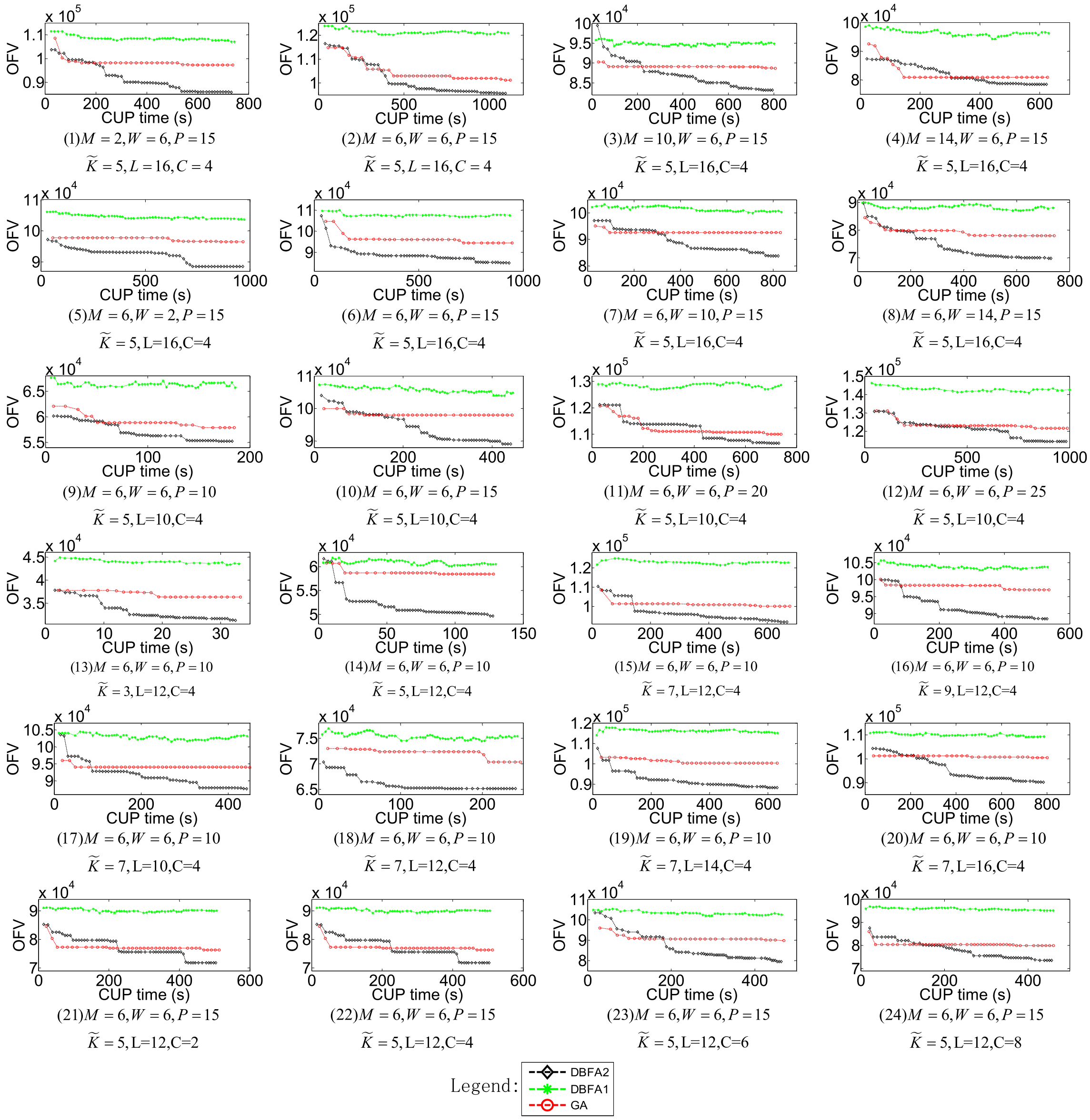
The experiments are performed on a Pentium-based Dell-compatible personal computer with 2.30 GHz clock-pulse and 4.00 GB RAM. The DBFA1, DBFA2 and GA algorithms are coded in C++, compiled with the Microsoft Visual C++ 6 compiler, and tested under Microsoft Windows 7 operating system.

The performance of the three algorithms is to be evaluated by the use of six impact factors, including the number of machine types (M), the number of worker type (W), the number of parts (P), the upper bound of operations for each part ($\widetilde{K}$), the number of workstations (L), and the number of cells (C). Six sets of sub-experiments are conducted. The instances (including small, medium and large size) have been randomly regenerated to verify the proposed algorithm. In the first set (displayed in Table 4), M is allowed to vary to test its impact effect, given $W = 6, P = 15, \widetilde{K} = 5, L = 16$ and $C = 4$. The other five sets (displayed in Tables 5–9) test the effects of varying W, P, $\widetilde{K}$, L and C, respectively. The other parameters for the randomly generated instances are listed in Table 3. B_u and B_d are given definite values, and the other parameters are given corresponding random integers between the minimum and the maximum. Given a typical instance with $M = 2, W = 3, P = 10, \widetilde{K} = 4, L = 9$, and $C = 3$, Figure 9 shows the respective convergence of DBFA1, DBFA2 and GA within the same runtime.

Each entry of Tables 4–9 represents the average of its associated 10 randomly generated instances. Let OFV_{BF1}, OFV_{BF2} and OFV_{GA} denote the average objective function values (OFV) using the DBFA1, DBFA2 and GA, respectively. Let $\triangle OFV_{BF1}^{BF2}$ denote the declining percentage of OFV_{BF2} over OFV_{BF1}, and let $\triangle OFV_{GA}^{BF2}$ denote the declining percentage of OFV_{BF2} over OFV_{GA}. Figure 10(1)–(24) show the convergence for an instance of each entry of Tables 4–9, respectively.

As can be seen from Tables 4–9, $\triangle OFV_{BF1}^{BF2}$ reaches 14.73–21.86% regardless of the variation of the six impact factors M, W, P, $\widetilde{K}$, L and C. There exists the following condition: Through one or more chemotactic steps, a bacterium may find a worse position than its best position in the current generation. So in the swimming strategy of DBFA1, a bacterium probably wastes many swimming steps but still results in a worse position than its best position in the current generation. In the swimming strategy of DBFA2, J_{cur}^{i}, instead of J_{last}^{i}, is used as a criterion for further swimming. Consequently, many invalid swimming steps can be avoided.

In addition, for the accumulated cost scheme of reproduction of DBFA1, it may not retain the fittest bacterium for subsequent generation. In the reproduction strategy of DBFA2, the bacterium with the best historic position in the current generation will be moved to the above position, and produce its offspring replacing inferior bacterium. So the best bacterium



(1)$M=2, W=6, P=15$
$\widetilde{K}=5, L=16, C=4$

(2)$M=6, W=6, P=15$
$\widetilde{K}=5, L=16, C=4$

(3)$M=10, W=6, P=15$
$\widetilde{K}=5$, L=16,C=4

(4)$M=14, W=6, P=15$
$\widetilde{K}=5$, L=16,C=4

(5)$M=6, W=2, P=15$
$\widetilde{K}=5$, L=16,C=4

(6)$M=6, W=6, P=15$
$\widetilde{K}=5$, L=16,C=4

(7)$M=6, W=10, P=15$
$\widetilde{K}=5$, L=16,C=4

(8)$M=6, W=14, P=15$
$\widetilde{K}=5$, L=16,C=4

(9)$M=6, W=6, P=10$
$\widetilde{K}=5$, L=10,C=4

(10)$M=6, W=6, P=15$
$\widetilde{K}=5$, L=10,C=4

(11)$M=6, W=6, P=20$
$\widetilde{K}=5$, L=10,C=4

(12)$M=6, W=6, P=25$
$\widetilde{K}=5$, L=10,C=4

(13)$M=6, W=6, P=10$
$\widetilde{K}=3$, L=12,C=4

(14)$M=6, W=6, P=10$
$\widetilde{K}=5$, L=12,C=4

(15)$M=6, W=6, P=10$
$\widetilde{K}=7$, L=12,C=4

(16)$M=6, W=6, P=10$
$\widetilde{K}=9$, L=12,C=4

(17)$M=6, W=6, P=10$
$\widetilde{K}=7$, L=10,C=4

(18)$M=6, W=6, P=10$
$\widetilde{K}=7$, L=12,C=4

(19)$M=6, W=6, P=10$
$\widetilde{K}=7$, L=14,C=4

(20)$M=6, W=6, P=10$
$\widetilde{K}=7$, L=16,C=4

(21)$M=6, W=6, P=15$
$\widetilde{K}=5$, L=12,C=2

(22)$M=6, W=6, P=15$
$\widetilde{K}=5$, L=12,C=4

(23)$M=6, W=6, P=15$
$\widetilde{K}=5$, L=12,C=6

(24)$M=6, W=6, P=15$
$\widetilde{K}=5$, L=12,C=8

Figure 10. Convergence for an instance of each entry of Tables 5–9.

Table 3. Parameters for randomly generated instances.

Parameter	Value	Min	Max
B_u: Upper bound of each cell size	$\llcorner (L/C)/1.2 \lrcorner$		
B_d: Lower bound of each cell size	$\llcorner (L/C) \times 1.2 \lrcorner + 1$		
K_p: Number of operations of part p		1	$\widetilde{K}$
δ_{pkmw}: Processing time of task		20	30
α_m: Fixed costs per unit of machine type m		10	20
$\widetilde{\alpha_m}$: Operating cost per unit of machine type m		20	40
S_w: Base salary cost per unit of worker type w		5	15
$\widetilde{S_w}$: Operating salary cost per unit of worker type w		10	30
θ_p: Inter-cell material handling cost of part p		4	10

Table 4. Performance comparison between the DBFA1, DBFA2 and GA for different number of machine types (M).

$W = 6$, $P = 15$, $\widetilde{K} = 5$ $L = 16, C = 4$		OFV_{BF1}	OFV_{BF2}	OFV_{GA}	$\triangle OFV_{BF1}^{BF2}$ (%)	$\triangle OFV_{GA}^{BF2}$ (%)	CPU (s)
M	2	104425.6	87681.6	93151.4	16.10	5.98	852.91
	6	108538.0	89358.0	95675.9	17.58	6.68	959.44
	10	104173.4	84426.6	91230.5	18.88	7.27	908.34
	14	102004.4	81432.5	89377.7	20.19	8.59	777.75

Table 5. Performance comparison between the DBFA1, DBFA2 and GA for different number of worker types (W).

$M = 6$, $P = 15$, $\widetilde{K} = 5$ $L = 16, C = 4$		OFV_{BF1}	OFV_{BF2}	OFV_{GA}	$\triangle OFV_{BF1}^{BF2}$ (%)	$\triangle OFV_{GA}^{BF2}$ (%)	CPU (s)
W	2	106015.0	87922.7	94707.0	16.91	6.81	907.27
	6	107739.1	87421.8	93899.5	19.01	7.25	880.97
	10	96585.9	79234.7	86207.3	17.87	8.01	718.22
	14	102149.8	82629.1	92741.9	19.12	10.81	853.37

Table 6. Performance comparison between the DBFA1, DBFA2 and GA for different number of parts (P).

$M = 6$, $W = 6$, $\widetilde{K} = 5$ $L = 10, C = 4$		OFV_{BF1}	OFV_{BF2}	OFV_{GA}	$\triangle OFV_{BF1}^{BF2}$ (%)	$\triangle OFV_{GA}^{BF2}$ (%)	CPU (s)
P	10	69690.8	56183.3	60748.5	19.39	7.47	152.72
	15	100037.2	83521.3	87793.1	16.69	4.83	385.47
	20	114262.0	95851.7	101387.4	15.97	5.51	604.32
	25	144406.3	123115.0	131115.5	14.73	6.04	1120.99

Table 7. Performance comparison between the DBFA1, DBFA2 and GA for different number of parts $\widetilde{K}$.

$M = 6$, $W = 6$, $P = 10$ $L = 12, C = 4$		OFV_{BF1}	OFV_{BF2}	OFV_{GA}	$\triangle OFV_{BF1}^{BF2}$ (%)	$\triangle OFV_{GA}^{BF2}$ (%)	CPU (s)
$\widetilde{K}$	3	48570.8	38007.1	41082.1	21.86	7.60	60.41
	5	72035.8	58681.1	63873.6	18.47	8.27	207.16
	7	92483.8	75555.4	81817.1	17.89	7.47	387.09
	9	112524.1	94235.8	100523.1	16.51	6.53	715.46

Table 8. Performance comparison between the DBFA1, DBFA2 and GA for different number of workstations (L).

$M = 6$, $W = 6$, $P = 10$ $\widetilde{K} = 7, C = 4$		OFV_{BF1}	OFV_{BF2}	OFV_{GA}	$\triangle OFV_{BF1}^{BF2}$ (%)	$\triangle OFV_{GA}^{BF2}$ (%)	CPU (s)
L	10	90901.5	73505.3	81558.1	19.26	9.96	300.03
	12	96628.0	80328.4	87421.6	16.79	8.14	468.56
	14	102253.5	84053.2	90937.7	17.52	7.20	604.41
	16	107753.3	90154.9	97534.9	16.27	7.60	828.15

can be passed on to the next generation. This speeds up the convergence. Similarly, in the elimination-dispersal strategy of DBFA1, the best bacterium may be dispersed to an inferior position, whereas in the elimination-dispersal strategy of DBFA2, the best bacterium is kept unchanged and transferred to the subsequent stage. This also speeds up the convergence and will not trap the solution into the local optima, because the tumble of chemotactic can modify the position of each bacterium in a random dimension and helps to jump out of the local optima.

Table 9. Performance comparison between the DBFA1, DBFA2 and GA for different number of cells (C).

$M = 6, W = 6, P = 15$ $\widetilde{K} = 5, L = 12$		OFV_{BF1}	OFV_{BF2}	OFV_{GA}	$\triangle OFV_{BF1}^{BF2}$ (%)	$\triangle OFV_{GA}^{BF2}$ (%)	CPU (s)
C	2	90995.7	74790.2	81742.0	17.76	8.44	405.14
	4	99162.4	80933.0	88035.2	18.19	7.89	511.82
	6	95202.3	75990.8	81672.0	19.94	6.73	438.19
	8	94611.0	78953.0	84577.9	16.57	6.62	493.81

We can also observe from Tables 4–9 that $\triangle OFV_{GA}^{BF2}$ reaches 4.83–10.81% in despite of the variation of the six impact factors. The reason can be demonstrated as follows: There are some similarities and differences between the DBFA2 and GA. The reproduction strategy of DBFA2 is similar to the selection plus crossover of GA, and the elimination-dispersal strategy of DBFA2 is similar to the mutation of GA. The DBFA2, however, has its unique chemotactic strategy. It is the tumble and the following swimming steps that lead to a deeper exploration of DBFA2 for the solution than the GA.

7. Conclusions

In this paper a new optimisation model of cellular manufacturing system (CMS) under DRC setting is introduced along with a DBFA embedding Priority Rule Based Parallel Schedule Generation Scheme (PRBPSGS). The advantage of the proposed model is simultaneously considering cell formation and task scheduling by assuming multi-skilled resources and operation sequence. Main constraints are the machine and worker time-capacity as well as the minimal and maximal cell sizes. The objective is to minimise the sum of inter-cell material handling cost, fixed costs of machines and workers, and operating costs of machines and workers. The PRBPSGS helps to generate a high quality initial solution through calculating task timetable.

The main difference between the DBFA1 and DBFA2 is that the former applies Passino's swimming strategy, reproduction strategy and elimination-dispersal strategy, and the latter uses these modified strategies. The performance of DBFA2 is evaluated and compared with the performance of DBFA1 and conventional genetic algorithm (GA) in terms of objective function values within the same runtime. It is observed that the quality of results obtained by DBFA2 is better than DBFA1 and GA regardless of the variation of some important parameters. The superiority of DBFA2 over DBFA1 lies in that the former may avoid many invalid swimming steps and transfer the fittest bacterium to subsequent generation. So the DBFA2 pays more attention to the efficiency of exploitation and convergence than the DBFA1. The advantage of DBFA2 over GA can be explained in that the DBFA2 has unique chemotactic strategy besides the characteristics of crossover, mutation and selection which the GA possesses. So the DBFA2 does well in balancing the depth of exploitation and the width of exploration, and pays more attention to the depth of exploitation than the conventional GA and even most efficient GA. An important research direction that may be pursued in the future is to extend chain precedence constraints of operations to arbitrary precedence constraints of parts and operations. The other potential interest would consider multi-level flexibility of resources, i.e. each machine or worker has a different number of functions or skills. In addition, this paper proposes the non-linear binary integer programming (NLBIP) which is a special case of mixed integer programming. From a formulation point of view, the NLBIP has the flexibility of being converted to constraint programming (CP) that may seem to be succinct apparently. For example, the binary variable in the NLBIP can be easily converted to the global *all-different* constraint which is often used in the CP. It is worthwhile to compare the characteristics and properties of NLBIP and CP (e.g. using IBM ILOG CPLEX software) according to optimisation success performance and computational processing time.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This research was supported by the Humanities and Social Sciences Youth Foundation of the Ministry of Education [grant number 14YJC630089]; Zhejiang Provincial Natural Science Foundation of China [grant number LY14G020014]; China Scholarship Council, Zhejiang Provincial Key Research Base of Humanities and Social Sciences in Hangzhou Dianzi University [grant number ZD03-201501], the Research Center of Information Technology & Economic and Social Development, and the National Natural Science Foundation of China [grant number 71471052]. The authors are grateful for the financial supports.

References

Arkat, J., H. Abdollahzadeh, and H. Ghahve. 2012. "A New Branch and Bound Algorithm for Cell Formation Problem." *Applied Mathematical Modelling* 36 (10): 5091–5100.

Arkat, J., M. H. Farahani, and F. Ahmadizar. 2012. "Multi-objective Genetic Algorithm for Cell Formation Problem Considering Cellular Layout and Operations Scheduling." *International Journal of Computer Integrated Manufacturing* 25 (7): 625–635.

Arkat, J., M. H. Farahani, and L. Hosseini. 2012. "Integrating Cell Formation with Cellular Layout and Operations Scheduling." *The International Journal of Advanced Manufacturing Technology* 61 (5–8): 637–647.

Ballakur, A. 1985. "An Investigation of Part Family/Machine Group Formation in Designing a Cellular Manufacturing System." PhD thesis, University of Wisconsin, Madison, WI.

Boutsinas, B. 2013. "Machine-part Cell Formation Using Biclustering." *European Journal of Operational Research* 230 (3): 563–572.

Chattopadhyay, M., S. Chattopadhyay, and P. K. Dan. 2011. "Machine-part Cell Formation through Visual Decipherable Clustering of Self-organizing Map." *The International Journal of Advanced Manufacturing Technology* 52 (9–12): 1019–1030.

Defersha, F. M., and M. Chen. 2008. "A Linear Programming Embedded Genetic Algorithm for an Integrated Cell Formation and Lot Sizing Considering Product Quality." *European Journal of Operational Research* 187 (1): 46–69.

Egilmez, G., B. Erenay, and G. A. Suer. 2014. "Stochastic Skill-based Manpower Allocation in a Cellular Manufacturing System." *Journal of Manufacturing Systems* 33 (4): 578–588.

Eguia, I., J. Racero, F. Guerrero, and S. Lozano. 2013. "Cell Formation and Scheduling of Part Families for Reconfigurable Cellular Manufacturing Systems Using Tabu Search." *Simulation* 89 (9): 1056–1072.

Elmi, A., M. Solimanpur, S. Topaloglu, and A. Elmi. 2011. "A Simulated Annealing Algorithm for the Job Shop Cell Scheduling Problem with Intercellular Moves and Reentrant Parts." *Computers & Industrial Engineering* 61 (1): 171–178.

Gupta, N. S., D. Devika, and V. Panpaliya. 2014. "Some Clarifications on the Use of Mahalanobis Distance for the Machine-part Cell Formation Problem." *The International Journal of Advanced Manufacturing Technology* 73 (5–8): 783–794.

Halat, K., and R. Bashirzadeh. 2015. "Concurrent Scheduling of Manufacturing Cells Considering Sequence-dependent Family Setup Times and Intercellular Transportation Times." *The International Journal of Advanced Manufacturing Technology* 77 (9–12): 1907–1915.

Hamedi, M., G. R. Esmaeilian, N. Ismail, and M. K. A. Ariffin. 2012. "Capability-based Virtual Cellular Manufacturing Systems Formation in Dual-resource Constrained Settings Using Tabu Search." *Computers & Industrial Engineering* 62 (4): 953–971.

Jain, A. K., S. C. Srivastava, S. N. Singh, and L. Srivastava. 2015. "Bacteria Foraging Optimization Based Bidding Strategy under Transmission Congestion." *IEEE Systems Journal* 9 (1): 141–151.

Jain, S. K. 2015. "Optimization of Dual Resonance Stacked Patch Resonator by Neural Hybridized Bacteria Foraging Algorithm." *Microwave and Optical Technology Letters* 57 (5): 1191–1199.

Kao, Y., and C. C. Chen. 2013. "A Differential Evolution Fuzzy Clustering Approach to Machine Cell Formation." *The International Journal of Advanced Manufacturing Technology* 65 (9–12): 1247–1259.

Karthikeyan, S., M. Saravanan, and K. Ganesh. 2012. "GT Machine Cell Formation Problem in Scheduling for Cellular Manufacturing System Using Meta-Heuristic Method." *Procedia Engineering* 38: 2537–2547.

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." *Computers & Operations Research* 39 (11): 2642–2658.

King, J. R., and V. Nakornchai. 1982. "Machine-component Group Formation in Group Technology: Review and Extension." *International Journal of Production Research* 20 (2): 117–133.

kioon, S. A., 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 (2): 414–428.

Logendran, R., P. Ramakrishna, and C. Sriskandarajah. 1994. "Tabu Search-based Heuristics for Cellular Manufacturing Systems in the Presence of Alternative Process Plans." *International Journal of Production Research* 32 (2): 273–297.

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." *Computers and Mathematics with Applications* 60 (4): 1014–1025.

Mishra, S. K., G. Panda, and R. Majhi. 2014. "Constrained Portfolio Asset Selection Using Multiobjective Bacteria Foraging Optimization." *Operational Research* 14 (1): 113–145.

Nouri, H., and T. S. Hong. 2012. "A Bacteria Foraging Algorithm Based Cell Formation Considering Operation Time." *Journal of Manufacturing Systems* 31 (3): 326–336.

Nouri, H., and S. H. Tang. 2013. "Development of Bacteria Foraging Optimization Algorithm for Cell Formation in Cellular Manufacturing System Considering Cell Load Variations." *Journal of Manufacturing Systems* 32 (1): 20–31.

Nouri, H., S. H. Tang, B. T. Hang Tuah, and M. K. Anuar. 2010. "BASE: A Bacteria Foraging Algorithm for Cell Formation with Sequence Data." *Journal of Manufacturing Systems* 29 (2–3): 102–110.

Panda, S., D. Mishra, B. Biswal, and M. Tripathy. 2014. "Revolute Manipulator Workspace Optimization Using a Modified Bacteria Foraging Algorithm: A Comparative Study." *Engineering Optimization* 46 (2): 181–199.

Passino, K. M. 2002. "Biomimicry of Bacterial Foraging for Distributed Optimization and Control." *IEEE Control Systems Magazine* 22 (3): 52–67.

Paydar, M. M., I. Mahdavi, I. Sharafuddin, and M. Solimanpur. 2010. "Applying Simulated Annealing for Designing Cellular Manufacturing Systems Using MDmTSP." *Computers & Industrial Engineering* 59 (4): 929–936.

Rabbani, M., F. Jolai, N. Manavizadeh, F. Radmehr, and B. Javadi. 2012. "Solving a Bi-objective Cell Formation Problem with Stochastic Production Quantities by a Two-phase Fuzzy Linear Programming Approach." *The International Journal of Advanced Manufacturing Technology* 58 (5–8): 709–722.

Rafiei, H., M. Rabbani, B. Nazaridoust, and S. S. Ramiyani. 2015. "Multi-objective Cell Formation Problem Considering Work-in-process Minimization." *The International Journal of Advanced Manufacturing Technology* 76 (9–12): 1947–1955.

Renna, P., and M. Ambrico. 2015. "Design and Reconfiguration Models for Dynamic Cellular Manufacturing to Handle Market Changes." *International Journal of Computer Integrated Manufacturing* 28 (2): 170–186.

Safaei, N., M. Saidi-Mehrabad, and M. S. Jabal-Ameli. 2008. "A Hybrid Simulated Annealing for Solving an Extended Model of Dynamic Cellular Manufacturing System." *European Journal of Operational Research* 185 (2): 563–592.

Sakhaii, M., R. Tavakkoli-Moghaddam, M. Bagheri, and B. Vatani. Forthcoming. "A Robust Optimization Approach for an Integrated Dynamic Cellular Manufacturing System and Production Planning with Unreliable Machines." *Applied Mathematical Modelling*.

Saravanan, M., and A. Noorul Haq. 2008. "A Scatter Search Method to Minimise Makespan of Cell Scheduling Problem." *International Journal of Agile Systems and Management* 3 (12): 18–36.

Solimanpur, M., and A. Elmi. 2011. "A Tabu Search Approach for Group Scheduling in Buffer-constrained Flow Shop Cells." *International Journal of Computer Integrated Manufacturing* 24 (3): 257–268.

Taghavifard, M. T. 2012. "Scheduling Cellular Manufacturing Systems Using ACO and GA." *International Journal of Applied Metaheuristic Computing* 3 (1): 48–64.

Tang, J., X. Wang, I. Kaku, and K. Yung. 2010. "Optimization of Parts Scheduling in Multiple Cells Considering Intercell Move Using Scatter Search Approach." *Journal of Intelligent Manufacturing* 21 (4): 525–537.

Tang, J., C. Yan, X. Wang, and C. Zeng. 2014. "Using Lagrangian Relaxation Decomposition with Heuristic to Integrate the Decisions of Cell Formation and Parts Scheduling Considering Intercell Moves." *IEEE Transactions on Automation Science and Engineering* 11 (4): 1110–1121.

Tavakkoli-Moghaddam, R., N. Javadian, A. Khorrami, and Y. Gholipour-Kanani. 2010. "Design of a Scatter Search Method for a Novel Multi-criteria Group Scheduling Problem in a Cellular Manufacturing System." *Expert Systems with Applications* 37 (3): 2661–2669.

Tripathy, M., and S. Mishra. 2015. "Coordinated Tuning of PSS and TCSC to Improve Hopf Bifurcation Margin in Multimachine Power System by a Modified Bacteria Foraging Algorithm." *Energy Systems* 66: 97–109.

Venkataramanaiah, S. 2008. "Scheduling in Cellular Manufacturing Systems: An Heuristic Approach." *International Journal of Production Research* 46 (2): 429–449.

Wang, X., J. Tang, and K. Yung. 2010. "A Scatter Search Approach with Dispatching Rules for a Joint Decision of Cell Formation and Parts Scheduling in Batches." *International Journal of Production Research* 48 (12): 3513–3534.

Won, Y., and R. Logendran. 2015. "Effective Two-phase p-median Approach for the Balanced Cell Formation in the Design of Cellular Manufacturing System." *International Journal of Production Research* 53 (9): 2730–2750.

Wu, X., C. H. Chu, Y. Wang, and D. Yue. 2007. "Genetic Algorithms for Integrating Cell Formation with Machine Layout and Scheduling." *Computers & Industrial Engineering* 53 (2): 277–289.