A BACTERIA FORAGING ALGORITHM FOR SOLVING INTEGRATED MULTI-PERIOD CELL FORMATION AND SUBCONTRACTING PRODUCTION PLANNING IN A DYNAMIC CELLULAR MANUFACTURING SYSTEM

S.H. Tang[1], H. Nouri[2*] & O. Motlagh[3]

[1,3] Department of Mechanical and Manufacturing Engineering
University Putra, Malaysia
[1]Saihong@eng.upm.edu.my, [3]omid_motlaq@yahoo.com

[2]Department of Mechanical Engineering
Science and Research Branch, Islamic Azad University, Tehran, Iran
ro_eagle@yahoo.com

ABSTRACT

The bacteria foraging algorithm (BFA) is a new computation technique inspired by the social foraging behaviour of *Escherichia coli* (*E. coli)* bacteria. Since the introduction of the BFA by Kevin M. Passino, there have been many challenges in employing this algorithm to problems other than those for which the algorithm was proposed. This research aims to apply this emerging optimisation algorithm to develop a mixed-integer programming model for designing cellular manufacturing systems (CMSs), and production planning in dynamic environments. In dynamic environments, product mix and part demand vary under multi-period planning horizons. Thus the best-designed cells for one period may not be adequate for subsequent periods, requiring their reconstruction. The advantages of the proposed model are as follows: consideration of batch inter-cell and intra-cell material handling by assuming the sequence of operations, allowing for alternative process plans for part types, and consideration of machine copying, with an emphasis on the effect of trade-offs between production and outsourcing costs. The goal is to minimise the sum of the machines' constant and variable costs, inter-cell and intra-cell material handling costs, reconstruction costs, partial subcontracting costs, and inventory carrying costs. In addition, a newly-developed BFA-based optimisation algorithm has been compared with the branch and bound algorithm. The results suggest that the proposed algorithm performs better than related works.

OPSOMMING

Die 'bacteria foraging algorithm' (BFA) is 'n berekeningstegniek gebaseeer op die sosiale soekgedrag van *Escherichia coli (E. coli)* bakterieë. Sedert die bekendstelling van BFA was daar talle uitdagings oor toepassings van die algoritme op ander probleme as dié waarvoor dit ontwikkel is. Dié navorsing poog om deur toepassing van die algoritme 'n gemengde heelgetalprogrammeringmodel te ontwikkel vir die ontwerp van sellulêre vervaardiging-stelsels sowel as die produksiebeplanning in dinamiese omgewings. Die doel is om die som van die masjienkoste, inter- en intraselmateriaalhanteringkoste, rekonstruksiekoste, gedeeltelike subkontrakteringkoste sowel as voorraaddrakoste te minimiseer. 'n Nuut ontwikkelde BFA-optimiseringalgoritme is ook met die vertakkings-en-begrensingsalgoritme vergelyk. Die resultate toon dat die voorgestelde algoritme gunstig presteer in vergelyking met soortgelyke algoritmes.

[2] The author is enrolled for a PhD (Mechanical Engineering) degree in the Department of Mechanical and Manufacturing Engineering, University Putra, Malaysia.

1. INTRODUCTION

Cellular manufacturing is described as a manufacturing method that produces part families within a single line or cell of machines serviced by operators and or robots that work only within the line or cell.

Cell formation (CF), as the first and most important step in designing cellular manufacturing systems, includes two fundamental tasks: part-family formation, and machine-cell formation. Part-family formation groups parts with similar geometric characteristics or processing necessities to determine the benefits of their similarities for design or manufacturing purposes. Because of the growing variety of consumer goods and shorter product life cycles, manufacturing organisations often face variations in product demand and product mix, leading to a dynamic or unstable production environment [19]. In dynamic production environments, a multi-period planning horizon is considered in which the product mix or the part demand rate in each period might be different – e.g., cyclical products demand. As commonly discussed in the literature, traditional CMS has many operational advantages over other manufacturing systems, such as job shop and flow shop; yet there are also some drawbacks, such as the limitation in shop flexibility and machine utilisation [19]. Traditional CMS discounts any changes in demand over time from product redesign and other factors. It assumes that product mix and part demand are constant for the entire planning horizon. Product mix passes on a set of part types to be produced in each period. In a dynamic environment, a planning horizon can be separated into smaller periods, where each has different product mix and demand requirements. Accordingly, the formed cells in a particular period may not be optimal or efficient for the next period.

To overcome the drawbacks of traditional CMS, the concept of the dynamic cellular manufacturing system (DCMS) has been introduced [19]. DCMS involves the reconstruction of manufacturing cells, including part families and machine groups in each period. Reconstruction entails the substitution of existing machines between cells, called 'machine replacement'; adding new machines to cells, or 'machine copying'; and removing existing machines from cells. Recently, the advancing computational practice of swarm intelligence has promised solutions to many engineering and financial problems. It has been found to be a powerful method in domains where analytical solutions have not been very effective.

The Bacterial Foraging Optimisation (BFO), invented by Passino [18], is one such evolutionary computational approach. It is inspired by the foraging behaviour of *Escherichia coli* bacteria in human intestines. According to this approach, foraging is seen as an optimisation process: the bacterium strives to maximise the energy gained per unit of foraging time. The BFO has been successfully applied to various real-world problems, such as harmonic estimation by Mishra [15], transmission loss reduction by Tripathy et al. [26], active power filter for load compensation by Mishra & Bhende [16], power network by Tripathy & Mishra [25], load forecasting by Ulagammai et al. [27], independent component analysis by Acharya et al. [1], identification of nonlinear dynamic systems by Majhi & Panda [11,12] and Panda et al. [17], stock market prediction by Majhi et al. [14], and adaptive channel equalisation by Majhi et al. [13].

In this paper, the bacteria foraging algorithm (BFA), which is a kind of metaheuristic algorithm, is proposed for minimising the sum of the machine constant and variable costs, inter-cell and intra-cell material handling, reconstruction costs, inventory costs, back order costs, and subcontracting costs in dynamic machine cell formation problems.

This paper is organised as follows: Section 2 introduces the literature review. Section 3 presents the proposed dynamic cell formation model. Section 4 overviews the optimisation of bacterial foraging and presents the proposed bacterial foraging algorithm. Computational experiences, with a number of test problems drawn from the literature, are shown in Section 5. Section 6 summarises the findings and draws conclusions.

2. LITERATURE REVIEW

The idea of the dynamic cellular manufacturing system (DCMS) was introduced by Rheault [19]. Seifoddin & Djassemi [23] introduced a simple dynamic part assignment (DPA) procedure and evaluated its effects on the performance of cellular manufacturing systems. In this procedure, rerouting of parts among machine cells is permitted for better use of machines. Baykasogylu & Gindy [4] developed a simulated annealing-based procedure for dynamic layout. Tavakkoli-Moghaddam et al. [24] discussed solving a cell formation (CF) problem in dynamic conditions by applying some traditional meta-heuristic methods such as the genetic algorithm, SA, and tabu search.

Jeon & Leep [9] developed a methodology that can be used to form manufacturing cells, applying both a new similarity coefficient based on the number of alternative routes during machine failure, and demand changes for multiple periods. The methodology is divided into two phases. A new similarity coefficient, which takes into account the number of available alternative routes during machine failure, is suggested in Phase I. A new methodology for cell formation, which considers the scheduling and operational aspects in cell design under demand changes, is begun in Phase II.

Defersha & Chen [6] proposed a comprehensive mathematical model for the design of CMS, based on the tooling requirements of the parts and the tooling available on the machines. The model integrates dynamic cell configuration, alternative routings, lot splitting, sequence of operations, multiple units of identical machines, machine capacity, workload balancing among cells, operation cost, cost of subcontracting part processing, tool consumption cost, setup cost, cell size limits, and machine adjacency constraints.

Defersha & Chen [7] developed a mathematical programming model following an integrated approach to cell configuration and lot sizing in a dynamic manufacturing environment. The model development also takes into account the impact of lot sizes on product quality. The solution of the mathematical model is to minimise both production and quality-related costs.

Boulif & Atif [5] studied manufacturing cell formation, considering the dynamic behaviour of the production system. First, they discussed the importance of considering the dynamic aspect of the problem, which has been insufficiently studied in the related literature. They further argued that, by considering a multi-periodic planning horizon, the problem can be resolved according to two strategies: passive and active.

Safaei et al. [20] developed a mixed-integer programming model to design the cellular manufacturing systems (CMSs) under dynamic environments. The advantages of the proposed model are that it takes into account the batch inter-cell and intra-cell material handling by assuming the sequence of operations; and it considers alternative process plans for part types, as well as machine copying. The major constraints are maximum cell size and machine time-capacity. The aim is to minimise the sum of the machine constant and variable costs, inter-cell and intra-cell material handling, and reconstruction costs. An efficient hybrid meta-heuristic method, based on mean field annealing and simulated annealing (MFA–SA), is applied to solve the proposed model. In this case, the MFA technique is used to generate a good initial solution for SA.

Safaei et al. [21] presented an integration of explicit uncertainty for a cell formation problem (CFP) with a dynamic condition in cellular manufacturing systems (CMS). A fuzzy approach was developed to solve an extended mixed-integer programming model of the dynamic CFP, in which piecewise fuzzy numbers are coefficients in the objective function and the technological matrix. The method is proposed to find out the optimal cell configuration in each period, with the optimal achievement of fuzzy objectives under the given constraints.

Safaei et al. [22] proposed an integrated mathematical model of multi-period cell formation and production planning in a dynamic cellular manufacturing system (DCMS) with the aim of minimising machines' inter-cell and intra-cell travelling costs, reconstruction, partial subcontracting, and inventory carrying costs. Emphasis is placed on the effect of the trade-off between production and outsourcing costs on the re-configuration of the cells in cellular manufacturing systems (CMSs) in a dynamic environment.

Bajestani et al. [3] presented a multi-objective dynamic cell formation problem, where the total cell load variation and sum of the miscellaneous costs – i.e., machine cost, inter-cell material handling cost, and machine replacement cost – are minimised concurrently, and a new multi-objective scatter search (MOSS) is designed for finding the local Pareto-optimal frontier.

Ah Kioon [2] presented and analysed a comprehensive model for the design of cellular manufacturing systems. A recurring theme in the research is a gradual approach when formulating CMS models. The proposed model is comprehensive, with a more integrated approach to CMS design, where production planning and system reconstruction decisions are integrated.

Deljoo et al. [8] discussed solving a cell formation problem in dynamic conditions using the genetic algorithm (GA). Some errors related to model formulation were discussed, while a new improved formulation for dynamic cell formation (DCF) problem was presented.

Mahdavi et al. [10] presented an integer mathematical programming model for the design of cellular manufacturing systems in a dynamic environment. The benefits of the proposed model are as follows: consideration of multi-period production planning, dynamic system reconstruction, duplicate machines, machine capacity, available time of workers, and worker assignment. The intention of the proposed model is to minimise holding and backorder costs, inter-cell material handling costs, machine and reconstruction costs, and hiring, firing, and salary costs.

As a general rule, incorporation of the theories of CMS design and PP is meant to be basic to the modelling and simulation of the real production environments. In fact, variations in product mix, volume, and introduction of new products are the key aspects that validate the incorporation of the CMS and PP. The research in this paper refers to the model of Safaei [22]. The purpose of a typical PP problem is to minimise total production-related costs such as variable production, inventory, and shortage costs, over a fixed planning horizon [22]. The main constraints of PP problem areas go beyond: 1) the inventory balance equation for balancing the inventory and/or shortages with those from the previous period, production quantity, and demand quantity; 2) capacity constraints that ensure that the total workload for each supply (labour, machines, etc.) does not exceed the capacity in each period [22]. Because of the dynamic environment of PP problems, the incorporation of CMS and PP makes problems very complex and difficult to compute. The reason is that the cell rearrangement is the vital operational feature of the CMS design in the dynamic environment, all of which must be considered in a real incorporated model [22].

Meta-heuristics methods, including Genetic Algorithm (GA), Simulated Annealing (SA), Tabu Search (TS), Ants Colony Systems (ACS), Particle Swarm Optimisation (PSO), and Bee Swarm Optimisation (BSO), are popular algorithms for solving cell formation problems using MPIM (Machine Part Incidence Matrix). Therefore it is believed that the opportunity exists for a wide range of research into the application of meta-heuristic techniques on cell formation, by considering real-life production factors and the effects of meta-heuristic tools' parameters that are set up to provide guidelines for users. As a consequence, it is advisable to examine the performance of meta-heuristic models using real-value matrices.

This research initiates the application of a bacteria foraging algorithm in dynamic cell formation. The model has been tested using a wide variety of problems reported in the literature, and has been found to produce consistently good results. The major purpose of

this work is to develop a simple yet efficient methodology that is capable of producing quick solutions for shop floor managers with minimal computational efforts.

3. PROBLEM FORMULATION

In this section a new mixed-integer programming model of integrated DCMS and production process planning is formulated under the following assumptions [22].

3.1 Assumptions

1) Each part type has a number of operations that should be processed respectively as numbered.
2) The processing time for all operations of a part type on different machine types is known and determined.
3) The demand for each part type in each period is acknowledged and determined.
4) The abilities and time-capacity of each machine type are known and constant over the planning horizon.
5) The constant cost of each machine type is recognised. The constant cost is independent of the workload allocated to the machine, and entails rent or overall service costs. This cost is considered for each machine in each cell and period, whether the machine is active or idle.
6) The variable cost of each machine type is recognised. Variable cost entails an operating cost that is dependent on the workload allocated to the machine.
7) The replacement cost of each machine type from one cell to another between periods is recognised. All machine types can be moved to any cell. Replacement cost is the sum of uninstalling, shifting, and installing costs, where installing and uninstalling costs are supposed to be the same. Note that, if a new machine is added to the system, we have only an installing cost. On the other hand, if a machine is removed from the system, we have only an uninstalling cost. Thus we assume that the unit cost of adding a new machine or removing a current machine is half of the replacement cost [22].
8) Parts are moved between and within cells as batches. As mentioned earlier, inter- and intra-cell batches have different costs and sizes. To reduce complexity, unit inter- and intra-cell travelling costs are constant for all moves, regardless of the distance travelled. In other words, we suppose that the distance between cells is the same, the distance between machines in each cell is the same, and also all machine types have the same dimensions [22].
9) The maximum number of cells formed in each period is specified beforehand.
10) The maximum cell size is known beforehand. A lower limit is not considered for the cell size, because we suppose that smaller cells are preferable.
11) All machine types are supposed to be multipurpose. Thus, each machine type can carry out one or more operations without incurring a modification cost. Likewise, each operation-part can be carried out on different machine types with different processing times.
12) Holding and backorders inventories are allowed between periods with known costs. Thus the demand for a part in a given period can be satisfied in the preceding or succeeding periods.
13) Partial subcontracting is allowed. This means that all or only a part of the demand of the part types can be subcontracted in each period. Also, the time-gap between releasing and receiving orders – the lead time – is known in advance.

3.2 Notation

Indices

c index for manufacturing cells (c = 1, . . . , C)
m index for machine types (m = 1, . . . ,M)
p index for part types (p = 1, . . . , P)
h index for time periods (h = 1, . . . ,H)
j index for operations belong to part p (j = 1, . . . ,Op)

3.3 Input parameters

P number of part types
O_p number of operations for part p
M number of machine types
C maximum number of cells that can be formed
D_{ph} demand for part p in period h
β_p^{inter} batch size for inter-cell travelling of part p
β_p^{intra} batch size for intra-cell travelling of part p
γ^{inter} inter-cell travelling cost per batch
γ^{intra} intra-cell travelling cost per batch. For justification of CMS, it is assumed that $(\gamma^{intra}/\beta_p^{intra}) < (\gamma^{inter}/\beta_p^{inter})$ [22].
α_m constant cost of machine type m in each period
β_m variable cost of machine type m for each unit time
δ_m replacement cost of machine type m
T_m time-capacity of machine type m in each period
UB maximal cell size
t_{jpm} processing time required to perform operation j of part type p on machine type m
$a_{jpm} = 1$ if operation j of part p can be done on machine type m; 0 otherwise
λ_p unit cost of subcontracting part p
η_p inventory carrying cost per unit part p during each period
ρ_p backorder cost per unit part p during each period
l lead time where l≤ $H-1$
M^{∞} large positive number

3.4 Decision variables

N_{mch} number of machines type m allocated to cell c in period h
K_{mch}^{+} number of machines type m added in cell c in period h
K_{mch}^{-} number of machines type m removed from cell c in period h
$X_{jpmch} = 1$ if operation j of part type p is done on machine type m in cell c in period h.
Q_{ph} number of part p that produced during period h
y_{ph} =1 if $Q_{ph} > 0$; 0 otherwise
S_{ph} number of demand of part p to be subcontracted in period h
I_{ph} inventory/backorder level of part p at the end of period h.
A negative value of I_{ph} means the backorder level or shortage

3.5 Mathematical model

By using the above notations, the proposed model is written as follows:

$$\min \quad Z = \sum_{h=1}^{H} \sum_{m=1}^{\mathrm{M}} \sum_{c=1}^{\mathrm{C}} N_{mch}\, \alpha_m$$

$$+ \sum_{h=1}^{H} \sum_{c=1}^{C} \sum_{p=1}^{P} \sum_{j=1}^{Op} \sum_{m=1}^{M} \beta_m\, Q_{ph} t_{jpm} x_{jpmch}$$

$$+ \frac{1}{2} \sum_{h=1}^{H} \sum_{p=1}^{P} \sum_{j=1}^{Op-1} \sum_{c=1}^{C} \left\lceil \frac{Q_{ph}}{\beta_p^{inter}} \right\rceil \gamma^{inter} \left| \sum_{m=1}^{M} x_{(j+1)pmch} - \sum_{m=1}^{M} x_{jpmch} \right|$$

$$+\frac{1}{2}\sum_{h=1}^{H}\sum_{p=1}^{P}\sum_{j=1}^{Op-1}\sum_{c=1}^{C}\left\lceil\frac{Q_{ph}}{\beta_p^{intra}}\right\rceil\gamma^{intra}\left(\sum_{m=1}^{M}\left|x_{(j+1)pmch}-x_{jpmch}\right|-\left|\sum_{m=1}^{M}x_{(j+1)pmch}-\sum_{m=1}^{M}x_{jpmch}\right|\right)$$

$$+\frac{1}{2}\sum_{h=1}^{H}\sum_{c=1}^{C}\sum_{m=1}^{M}\delta_m\left(K_{mch}^{+}+K_{mch}^{-}\right)$$

$$+\sum_{h=1}^{H}\sum_{p=1}^{P}\eta_p I_{ph}^{+}+\sum_{h=1}^{H}\sum_{p=1}^{P}\rho_p I_{ph}^{-}+\sum_{h=1}^{H}\sum_{p=1}^{P}\lambda_p\, S_{ph} \quad (1)$$

s.t.

$$\sum_{c=1}^{C}\sum_{m=1}^{M}a_{jpm}\,x_{jpmch}=y_{ph} \qquad \forall\, j,p,h, \quad (2)$$

$$\sum_{p=1}^{P}\sum_{j=1}^{Op}Q_{ph}t_{jpm}x_{jpmch}\le T_m\,N_{mch} \qquad \forall\, m,c,h, \quad (3)$$

$$\sum_{m=1}^{M}N_{mch}\le UB \quad \forall\, c,h, \quad (4)$$

$$N_{mc(h-1)}+K_{mch}^{+}-K_{mch}^{-}=N_{mch} \qquad \forall m,c,h \quad (5)$$

$$I_{ph}=I_{p(h-1)}+Q_{ph}+S_{p(h-1)}-D_{ph} \qquad \forall p,h \quad (6)$$

$$I_{ph}^{+}\le I_{ph}\,,\ I_{ph}^{-}\ge -I_{ph} \qquad \forall p,h;\ \ I_{pH}=0 \quad \forall p \quad (7)$$

$$Q_{ph}\le M^{\infty}y_{ph} \qquad \forall p,h \quad (8)$$

$$x_{jpmch}\,,y_{ph}\in\{0,1\},N_{mch},K_{mch}^{+}\,,K_{mch}^{-},Q_{ph},I_{ph}^{+},I_{ph}^{-},S_{ph}\ge 0 \ \text{ and integer }, -\infty<I_{ph}<\infty \text{ and integer} \quad (9)$$

The objective function given in Eq. (1) is to minimise the total sum of the machine constant and variable costs, inter-cell and intra-cell travelling cost, reconstruction cost, inventory cost, backorder cost, and subcontracting costs over the planning period. The first term represents the constant cost of all machines needed in all cells over the planning period. This cost is attained by the product of the number of machine type m allocated to cell c in period h and their associated costs. This term does not allow for extra machine copying, and forces the model to achieve maximum machine exploitation. The second term is the variable cost of all machines needed in all cells over the planning period. It is the sum of the product of the time-workload assigned to each machine type in each cell and their associated cost. This term balances the workload assigned to machines at each cell [20]. It should be remembered that each machine type with low (or high) constant cost does not necessarily have a low (or high) variable cost, and vice versa. Thus there is a deal between the first and second terms of the objective function, with the remaining terms of the objective function for choosing the machines with high capability and low relatively cost.

In general, the proposed model picks a machine type based on four criteria: constant cost, variable cost, processing capabilities, and capacity [20]. The third term computes the total inter-cell material handling cost. It is the sum of the product of the number of inter-cell transfers (i.e., $Q_{ph/}\beta_p^{inter}$) resulting from both the consecutive operation of each part type and the cost of moving an inter-cell batch of each part type (γ^{inter}). Likewise, the fourth term of the objective function calculates the total intra-cell material handling cost. It is

the sum of the number of the product of intra-cell transfers (i.e., $Q_{ph/}\beta_p^{intra}$) resulting from both the consecutive operation of each part type and the cost of moving an intra-cell batch of each part type (γ^{intra}). The sequence of operations directly affects the inter- and intra-cell moving costs [20]. If two consecutive operations of a part type that must be processed are assigned to different cells, then the system will acquire a unit inter-cell moving cost (γ^{inter}). Conversely, if two consecutive operations of a part type are allocated to the same cell but to different machines, then the system will acquire a unit intra-cell moving cost (γ^{intra}). From the modelling approach, if two consecutive operations j and j+1 of part p cause inter- or intra-cell moving in period h, then, $\omega_{jpch} = \sum_{m=1}^{M}\left|x_{(j+1)pmch} - x_{jpmch}\right| = 1$. On the other hand, if two consecutive operations j and j+1 of part p cause only inter-cell moving in period h, then, $\mu_{jpch} = \left|\sum_{m=1}^{M} x_{(j+1)pmch} - \sum_{m=1}^{M} x_{jpmch}\right|$. Consequently, if $\omega_{jpch} - \mu_{jpch} = 1$, then intra-cell moving will be encountered [20].

Inter-cell and intra-cell moving costs are computed with respect to the third and fourth terms of the objective function. The coefficient 1/2 in the third and fourth terms is implanted because each inter- and intra-cell movement has been calculated twice, since moving from a cell or a machine corresponds to moving towards another cell or machine.

The fifth term of the objective function computes the reconstruction cost in a dynamic environment. This term is the sum of the number of products of relocated, added, and/or eliminated machines, and their associated costs. The coefficient 1/2 in this term is implanted because each replacement is considered twice in computations [20].

The last three terms are related to the total sum of the PP costs, including inventory carrying, backorder incurring, and subcontracting costs. The first term is the sum of the product of the inventory level for each part type at the end of the given period, and related cost. Similarly, the second term is the sum of the product of the backorder shortage level for each part type at the end of the given period, and related cost. The third term is the sum of the product of the number of the subcontracted parts and related cost [22].

Eq. (2) ensures that each part operation is assigned to one machine and one cell in each period. Eq. (3) guarantees that machine capacities are not exceeded, and thus satisfy the demand. Also, this equation determines the required number of each machine type in each cell, to avoid machine duplication. Inequality (4) makes certain that the maximum cell size is not violated. Eq. (5) is named 'balance constraint', and ensures that the number of machines in the current period is equal to the number of machines in the previous period, plus the number of machines being moved in, minus the number of machines being moved out. On the other hand, it ensures the conserving of machines over the time period, and remembers the available machine types in each period.

The last term of the objective function and combination constraint (6) sets up a connection between periods [20]. Eq. (6) points out the balance inventory constraint between periods for each part type in each period. It denotes that the inventory level of each part at the end of each period is equal to the inventory level of the part at the end of the previous period, plus the quantity of production and quantity of subcontracting, minus the part demand rate in the current period. Eq. (7) finds out the inventory and backorder level of each part type in each period. Clearly, the total demand for all part types over the planning period must be fulfilled during the planning period. As a result, the inventory and backorder level of all part types in the last period must be zero [22].

Eq. (8) balances Eq. (2), guaranteeing that a portion of the part demand can be produced at the given period if its operations are allocated in the first constraint given in Eq. (2). Eq. (5), Eq. (6), and Eq. (7) establish a connection between periods. By removing these equations, the model will be fully decomposed into sub-problems corresponding to different periods [22]. This decomposition approach can be used to obtain a lower boundary for the objective function value. By removing the last three terms of Eq. (1), and relaxing from the demands of Eqs. (6) – (9), the basic DCMS model of Safaei et al. [20] can be obtained

without considering outsourcing. Linearisation of the proposed model is explained fully in [22].

By reconstructing Eq. (3) and (5), N_{mch} ,K_{mch}^{+} and K_{mch}^{-} can be formulated as follows:

$$N_{mch} = \left(\sum_{p=1}^{P}\sum_{j=1}^{Op} Q_{ph} t_{jpm} x_{jpmch}\right)/T_m \qquad \forall\, m, c, h, \tag{10}$$

$$K_{mch}^{+} = \max\{N_{mch} - N_{mc(h-1)}, 0\},\ K_{mch}^{-} = \max\{N_{mc(h-1)} - N_{mch}, 0\} \tag{11}$$

4. BACTERIA FORAGING OPTIMISATION: A BRIEF OVERVIEW AND ALGORITHM

4.1 BFO operators

The survival of a species in any natural evolutionary process depends on its fitness criteria, which in turn rely on its food searching and motile behaviour. The law of evolution supports those species that have a better food searching ability, and either eliminates or reshapes those with a poor search ability. The genes of those species that are stronger propagate in the evolutionary chain, since they possess the ability to reproduce even better species in future generations. In the light of this, a clear understanding and modelling of the foraging behaviour in any evolutionary species is critical to its successful application in any non-linear system optimisation algorithm. The foraging strategy of *Escherichia coli* bacteria in human intestines can be explained by four processes: chemo taxis, swarming, reproduction, and elimination–dispersal [18].

The characteristics of bacteria travelling in search of food are defined in two ways: swimming and tumbling, referred to together as 'chemo taxis'. A bacterium is said to be 'swimming' if it travels in a predefined direction, and 'tumbling' if moving in an altogether different direction. Mathematically, the tumble of any bacterium can be described by a unit length of random direction *φ(j)* multiplied by the step length of that bacterium *C(i).* In the case of swimming, this random length is predefined [18].

For the bacteria to reach the richest food location (i.e., for the algorithm to converge at the solution point), it is desired that the optimum bacterium, until a point of time in the search period, tries to attract other bacteria so that they converge together at the desired location (solution point) more rapidly [18]. Achieving this entails a penalty function based upon the relative distances of each bacterium from the fittest bacterium until that search duration is added to the original cost function. Eventually this penalty function becomes zero when all the bacteria have merged into the solution point. The purpose of swarming is to make the bacteria congregate into groups and travel as concentric units of high density [18]. Since a cell-to-cell attraction calculation is needed for all bacteria in swarming, swarming increases the computational time of algorithm execution, while it only slightly enhances the performance of the BFO algorithm. Thus swarming has not been considered in the proposed algorithm.

After evolving through several chemo tactic stages, the original set of bacteria reaches the reproduction stage. Here the best set of bacteria (selected from all of the chemo tactic stages) is divided into two groups. The healthier group of bacteria replaces the other group, which is eliminated due to its poorer foraging abilities. This effects a constant population of bacteria in the evolutionary process [18]. After *Nc* chemo tactic steps, a reproduction step is taken. Let *Nre* be the number of reproduction steps to be taken. To expedite the convergence of the proposed algorithm, it is assumed that only 1/5 of bacteria reproduce. Let

$$S_r = \frac{S}{5} \tag{12}$$

be the number of population members that have had sufficient nutrients to reproduce with no mutations. For reproduction, the population is sorted in the order of ascending accumulated cost. A higher accumulated cost means that a bacterium has not received many nutrients during its lifetime of foraging, and hence is not 'healthy' enough, making it unlikely to reproduce. Accordingly, the $S - S_r$ least healthy bacteria die and the other S_r healthier bacteria are each copied into four places at the same location as their parents (S-Sr) bacteria had been copied. This method rewards bacteria that have encountered a lot of nutrients, and allows for a constant population size, which is convenient for coding the algorithm.

In the evolutionary process, any sudden unexpected event can occur that may drastically alter a smooth evolution, thus ultimately leading to the elimination of the set of bacteria and/or their dispersal into a new environment. Ironically, instead of disturbing the usual chemo tactic growth of the set of bacteria, this unknown event may place a newer set of bacteria nearer to the food location [18].

From a wide perspective, elimination and dispersal are parts of the population level's long-distance motion behaviour [18]. In its application to optimisation, it helps to reduce the behaviour of *stagnation* (i.e., trapping bacteria in a premature solution point or local optima) that is quite often encountered in such parallel search algorithms [18]. Let *Ned* be the number of elimination-dispersal events. In any elimination and dispersal event, every bacterium in the population is considered with probability *ped* to experience elimination and dispersal.

4.2 Problem-solving representation schema

To determine the solution representation schema, the two matrices x_{hj}^{p}, and y_{hj}^{p} have been used during each period, as shown in Figure 5. The matrix [X] denotes the allocation of operation-part to machine, and the matrix [Y] denotes the allocation of operation-part to cell. x_{hj}^{p} is the machine by which the operation j of part p must be performed, where $x_{hj}^{p} \in \varphi_{jp}$ and $\varphi_{jp} = \{m | a_{jpm} = 1\}$. Also, y_{hj}^{p} is the cell that operation j of part p is allocated to, where $1 \leq y_{hj}^{p} \leq C$.

$$\mathbf{X}=\begin{bmatrix} x_{11}^{p} & \cdots & x_{1Op}^{p} \\ \vdots & & \vdots \\ x_{h1}^{p} & \cdots & x_{HOp}^{p} \end{bmatrix} \qquad \mathbf{Y}=\begin{bmatrix} y_{11}^{p} & \cdots & y_{1Op}^{p} \\ \vdots & & \vdots \\ y_{h1}^{p} & \cdots & y_{HOp}^{p} \end{bmatrix}$$

Figure 1: Solution representation schema in each period h

To generate neighborhood solutions, four heuristic operators have been used with respect to the proposed structure, shown in Figure 1, as follows:

Local-mutation on [X] matrix: To implement this operator, an operation of a part in a period (e.g., operation j of part p in period h) is randomly selected and then randomly allocated to another machine as $x_{hj}^{p} \rightarrow x_{hj'}^{p}$, where $x_{hj}^{p} \neq x_{hj'}^{p}$, $and\ x_{hj'}^{p} \in \varphi_{jp}$.

Local-mutation on [Y] matrix: To implement this operator, an operation of a part in a period (e.g., operation j of part p in period h) is randomly selected and then randomly allocated to another cell as $y_{hj}^{p} \rightarrow y_{hj'}^{p}$, where $y_{hj'}^{p} \neq y_{hj}^{p}$ and$1 \leq y_{hj'}^{p} \leq C$.

Global-mutation on [X] matrix: To implement this operator, a period (e.g., period h of part p) is randomly selected and then all its operations in this period are randomly allocated to other machines as $x_{hj}^{p} \rightarrow x_{h'j}^{p'}\ \forall j$ where $x_{hj}^{p} \neq x_{h'j}^{p'}$ and$x_{h'j}^{p'} \in \varphi_{jp}\ \forall j$.

Global-mutation on [Y] matrix: To implement this operator, a period (e.g., period h of part p) is randomly selected and then all its operations in this period are randomly allocated to the random cell as $y_{hj}^{p} \rightarrow y_{h'j}^{p'}\ \forall j$ where $y_{hj}^{p} \neq y_{h'j}^{p'}$ and $1 \leq y_{h'j}^{p'} \leq C\ \forall j$.

These four operators are experimental and can be used with the same rate, but the system designer can tune this operator's applying rate to accelerate the convergence of the algorithm. In other words, when the tuning of the operators is changed, a significant improvement cannot be seen in the performance of BFA. After the implementation of each operator, the value of the decision variable N_{mch} must be updated with respect to the entries of matrices [X] and [Y]. This is done by using Eq. (10) and then the decision variables k^{+}_{mch} and k^{-}_{mch} are updated using Eq. (11).

If the intention is to find the minimum of $Z(\theta)$, $\theta \in R^P$, where the gradient $\nabla Z(\theta)$ cannot be measured or analytically described, then the bacterial foraging algorithm is employed to solve such a non-gradient optimisation problem. First, suppose that θ is the position of a bacterium and $Z(\theta)$ represents the combined effects of attractants and repellants from the environment with, for example, $Z(\theta) < 0$, $Z(\theta) = 0$, and $Z(\theta) > 0$ representing the bacterium location in nutrient-rich, neutral, and noxious environments respectively. Basically, chemo taxis is a foraging behaviour that implements a type of optimisation where bacteria tend to climb up the nutrient concentration (i.e., find lower and lower values of $Z(\theta)$), avoid noxious substances, and search for ways out of neutral media (avoid being at positions θ where $Z(\theta) \geq 0$). They implement a type of biased random walk [18].

Define a chemo tactic step to be a tumble followed by a tumble or a tumble followed by a run. Let t be the index for the chemo tactic step. Let k be the index for the reproduction step. Let l be the index for the elimination-dispersal event. Let

$$P(t,k,l) = \{\theta^i(t,k,l) \mid i = 1,2,\ldots,S\} \quad (13)$$

represent the position of each member in the population of the S bacteria at the t^{th} chemo tactic step, kth reproduction step, and l the elimination-dispersal event. Also let $Z(i, t,k,l)$ denote the cost at the location of the i^{th} bacterium $\theta^i(t,k,l) \in R^p$ (we may drop the indices and refer to the i^{th} bacterium position as θ^i). Note that we shall interchangeably refer to Z as being a 'cost' (using terminology from optimisation theory) and as being a nutrient surface (in reference to the biological connections) [18]. Let Nc be the length of the lifetime of the bacteria as measured by the number of chemo tactic steps they take during their life. The algorithm in Figure 2 represents the tumbling process.

a) For i = 1,2,…, S, take a chemo tactic step for bacterium i as follows.
b) Compute Z(i, t,k,l).
c) Let Z_{Last} = Z (i ,t,k,l) to save this value since we may find a better cost via a tumbling or swimming.
d) Tumble
If $p_{ytumble}$=1 and $p^{}_{ylocal}$>RND then DO local mutation for $y^p_{hj}(i,t,k,l)$ and $f^{}_{yl} = 1$
If $p_{ytumble}$=1 and $p^{}_{ylocal}$<RND then DO global mutation for $y^p_{hj}(i,t,k,l)$ and $f^{}_{yg}$=1
If $p_{ytumble}$=0 and $p^{}_{xlocal}$>RND then DO local mutation for $x^p_{hj}(i,t,k,l)$ and $f^{}_{xl}$=1
If $p_{ytumble}$=0 and $p^{}_{xlocal}$<RND then DO global mutation for $x^p_{hj}(i,t,k,l)$ and $f^{}_{xg}$=1
f) Compute Z(i, t + 1,k,l).
($p_{ytumble}$, $p^{}_{ylocal}$ and $p^{}_{xlocal}$ are probability of tumbling on y, local mutation on y and local mutation on x respectively)

Figure 2: Pseudo-code of tumbling steps

If at θ^i $(t+1,k,l)$ the cost $Z(i, t+1,k,l)$ is better (lower) than at θ^i (t,k,l), then another step of swimming in the same direction will be taken; and again, if that step results in a position with a lower cost value than that of the previous step, another step will be taken. The swimming continues as long as the cost reduces. However, this process is limited to a maximum number of steps, Ns. This means that the cell will tend to keep moving as long as it is heading in the direction of increasingly favourable environments. The algorithm in Figure 3 describes the swimming process:

Swim:
i) Let m = 0 (counter for swim length).
ii) While m < Ns (if bacteria have not climbed down too long)
Let m = m+ 1.
I f Z(i, t + 1,k,l) < Z_{Last} (if doing better), let Z_{Last} =Z(i, t + 1,k,l) and let

If f_{yg}=1 then DO global mutation for $y_{hj}^{p}(i,t+1,k,l)$ ∃ h ≠h_p, p ≠p_p
If f_{xg}=1 then DO global mutation for $x_{hj}^{p}(i,t+1,k,l)$ ∃ h ≠h_p, p ≠p_p
If f_{xl}=1 then DO local mutation for $x_{hj}^{p}(i,t+1,k,l)$ ∃ h ≠h_p, p ≠p_p ,j≠j_p
If f_{yl}=1 then DO local mutation for $y_{hj}^{p}(i,t+1,k,l)$ ∃ h ≠h_p, p ≠p_p ,j≠j_p
(h_p, p_p ,j_p are numbers of period,part and operation which randomize selected in pervious tumbling or swimming period, this strategy prevents to fall in local minima.)

Figure 3: Pseudo-code of the swimming process

To satisfy the definition of x_{jpmch} (= 1 if operation j of part type p is done on machine type m in cell c in period h) and constraint (2), the following subroutine is introduced (Figure 4). As indicated in the subroutine, if operation j of part p during period h to cell c ($y_{hj}^{p} = c$) and machine m ($x_{hj}^{p} = m$) is assigned and a_{jpm} =1 then $x_{jpmch} = 1$ else $x_{jpmch} = 0$.

If x_{hj}^{p} ≠0 and y_{hj}^{p}≠0 and a(j,p, x_{hj}^{p})=1
then $x_{jpmch} = 1$ else $x_{jpmch} = 0$ ∀ j,h,p

Figure 4: Definition of x_{jpmch} subroutine

From the viewpoint of factory management, satisfaction of constraint (4) is most critical. Therefore, for the changing solution representation schema, the application of a repairing programme is needed. The subroutine shown in Figure 5 is designed to solve this problem. Accordingly, when the algorithm reaches the final solution, it modifies the solution and tries to reallocate machines that occupy more than the maximum cell size to other cells that have fewer machines than their cell size constraint.

if $\sum_{m=1}^{M} N_{mch} > UB$ ∀ c,h, then

Find c′ ∈ $\{\sum_{m=1}^{M} N_{mc'h} < UB\}$
If exist c′ then
Move the machines are more than UB in cell c to cell c′ until $\sum_{m=1}^{M} N_{mc'h} \leq UB$
Set $x_{hj_{new}}^{p}$ and $y_{hj_{new}}^{p}$
Calculate $x_{jpmch_{new}}$ and $N_{mch_{new}}$
Calculate objective function
Assign $x_{hj_{new}}^{p}$ and $y_{hj_{new}}^{p}$ to x_{hj}^{p} and y_{hj}^{p} for tumbling/ swimming
End
End

Figure 5: Solution representation of schema repairing pseudo-code

Adding the inventory cost for all periods of any part according to Eq.(6), and considering $I_{pH} = 0\ \forall p$:

$$\begin{aligned}
&I_{p1} = I_{p0} + Q_{p1} + S_{p0} - D_{p1} && \forall p \\
&+ \quad + \quad + \quad + \quad + \\
&I_{p2} = I_{p1} + Q_{p2} + S_{p1} - D_{p2} && \forall p \\
&+ \quad + \quad + \quad + \quad + \\
&\vdots \quad \vdots \\
&I_{pH} = I_{p(H-1)} + Q_{pH} + S_{p(H-1)} - D_{pH} && \forall p
\end{aligned} \tag{14}$$

this summing up leads to:

$$SQ_p = \left(\sum_{h=1}^{H} S_{ph} + \sum_{h=1}^{H} Q_{ph}\right) = \sum_{h=1}^{H} D_{ph} - I_{p0} \quad \forall p \tag{15}$$

This relationship can be used to determine the solution representation schema for process planning parameters (Q_{ph} and S_{ph}), considering the following definition:

$$TS_p = \sum_{h=1}^{H} S_{ph} \; ; \; TQ_p = \sum_{h=1}^{H} Q_{ph} \qquad \forall p \tag{16}$$

Following this routine, TQ_p of part p at all periods is randomly selected from the summation of TQ_p and TS_p $(TQ_p + TS_p)$, and after that, each Q_{ph} at any period for each part is selected randomly from random part of TQ_p and then reduced from TQ_p of part p at all periods. Likewise, S_{ph} of part p at any period is selected randomly from random part of TS_p and then reduced from TS_p of part p at all periods. Therefore the procedure shown in Figure 6 is proposed to determine Q_{ph} and Q_{ph} at each period for all parts:

Step 1: calculate $TQ_p \quad \forall p$

$$TQ_p = RND \times SQ_p$$

Step 2: calculate $Q_{ph} \quad \forall p$

$$Q_{ph} = RND \times (TQ_p - (Q_{p1} + Q_{p2} + \cdots + Q_{p(h-1)})) \qquad \forall\, h < H$$
$$Q_{pH} = TQ_p - \left(Q_{p1} + Q_{p2} + \cdots + Q_{p(H-1)}\right)$$

Step 3: calculate $S_{ph} \quad \forall p$

$$TS_p = SQ_p - TQ_p$$

$$S_{ph} = RND \times (TS_p - (S_{p1} + S_{p2} + \cdots + S_{p(h-1)})) \qquad \forall\, h < H$$
$$S_{pH} = TS_p - \left(S_{p1} + S_{p2} \ldots + S_{p(H-1)}\right)$$

Step 4:calculate $I_{p(h-1)} = I_{ph} - Q_{ph} - S_{p(h-1)} + D_{ph}$, $I_{pH}=0 \qquad \forall p, h$

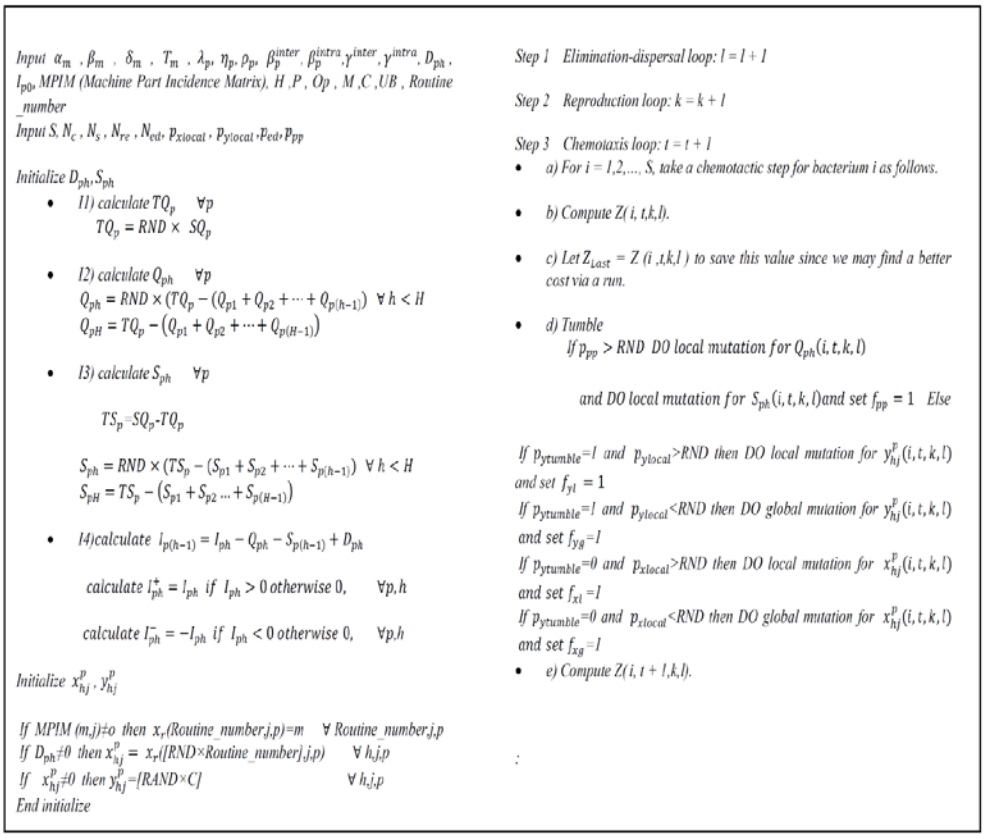
Step 5:calculate $I_{ph}^{+} = I_{ph}$ *if* $I_{ph} > 0$ *otherwise* $0, \qquad \forall p, h$

Step 6:calculate $I_{ph}^{-} = -I_{ph}$ *if* $I_{ph} < 0$ *otherwise* $0, \qquad \forall p, h$

Figure 6: Production planning data initialisation procedure pseudo-code

For tumbling and swimming, the local mutation operation has been applied on $[Q]_{p\times h}$ and the $[S]_{p\times h}$matrix. To implement this operator, a part is randomly selected, and then its TQ_p amount is randomly selected, and therefore, as shown in Figure 6, all new Q_{ph} and S_{ph} amounts are generated as $Q_{ph} \rightarrow Q_{ph'}$, $S_{ph} \rightarrow S_{ph'}$ $\forall h$ where p is not equal to the part that was selected in the previous tumbling or swimming to avoid local minima.

To construct cells and plan production concurrently, initially matrices $[Q]_{p\times h}$ and $[S]_{p\times h}$ are formed. Then, based on the obtained Q_{ph}, the cells are formed simultaneously. This routine is followed in all tumbling and swimming steps in such a way that first Q_{ph} and S_{ph} are determined, and then, x_{hj}^{p} and y_{hj}^{p} are created. Likewise, to choose the chemo taxis representation, the probability P_{pp} is considered so that, if $P_{pp} > RND$, then doing chemo taxing of Q_{ph} and S_{ph}, otherwise, doing chemo taxing for x_{hj}^{p} and y_{hj}^{p}. The proposed bacteria foraging algorithm to solve the proposed mathematical model is shown in Figure 7.



Input α_m , β_m , δ_m , T_m , λ_p, η_p, ρ_p, β_p^{inter}, β_p^{intra}, γ^{inter}, γ^{intra}, D_{ph}, I_{p0}, MPIM (Machine Part Incidence Matrix), H ,P , Op , M ,C ,UB , Routine _number

Input S, N_c , N_s , N_{re} , N_{ed}, p_{xlocal} , p_{ylocal} ,p_{ed}, p_{pp}

Initialize D_{ph}, S_{ph}

- I1) calculate TQ_p $\forall p$

  $TQ_p = RND \times SQ_p$

- I2) calculate Q_{ph} $\forall p$

  $Q_{ph} = RND \times (TQ_p - (Q_{p1} + Q_{p2} + \cdots + Q_{p(h-1)})$ $\forall h < H$

  $Q_{pH} = TQ_p - (Q_{p1} + Q_{p2} + \cdots + Q_{p(H-1)})$

- I3) calculate S_{ph} $\forall p$

  $TS_p = SQ_p - TQ_p$

  $S_{ph} = RND \times (TS_p - (S_{p1} + S_{p2} + \cdots + S_{p(h-1)})$ $\forall h < H$

  $S_{pH} = TS_p - (S_{p1} + S_{p2} \ldots + S_{p(H-1)})$

- I4)calculate $I_{p(h-1)} = I_{ph} - Q_{ph} - S_{p(h-1)} + D_{ph}$

  calculate $I_{ph}^{+} = I_{ph}$ if $I_{ph} > 0$ otherwise 0, $\forall p,h$

  calculate $I_{ph}^{-} = -I_{ph}$ if $I_{ph} < 0$ otherwise 0, $\forall p,h$

Initialize x_{hj}^{p} , y_{hj}^{p}

If MPIM (m,j)≠o then x_r(Routine_number,j,p)=m ∀ Routine_number,j,p

If $D_{ph} \neq 0$ then $x_{hj}^{p} = x_r([RND \times Routine\_number],j,p)$ ∀ h,j,p

If $x_{hj}^{p} \neq 0$ then $y_{hj}^{p} = [RAND \times C]$ ∀ h,j,p

End initialize

Step 1 Elimination-dispersal loop: l = l + 1

Step 2 Reproduction loop: k = k + 1

Step 3 Chemotaxis loop: t = t + 1

- a) For i = 1,2,..., S, take a chemotactic step for bacterium i as follows.
- b) Compute Z(i, t,k,l).
- c) Let Z_{Last} = Z (i ,t,k,l) to save this value since we may find a better cost via a run.
- d) Tumble

  If $p_{pp} > RND$ DO local mutation for $Q_{ph}(i,t,k,l)$

  and DO local mutation for $S_{ph}(i,t,k,l)$and set $f_{pp} = 1$ Else

If $p_{ytumble}$=1 and p_{ylocal}>RND then DO local mutation for $y_{hj}^{p}(i,t,k,l)$ and set $f_{yl} = 1$

If $p_{ytumble}$=1 and p_{ylecal}<RND then DO global mutation for $y_{hj}^{p}(i,t,k,l)$ and set f_{yg}=1

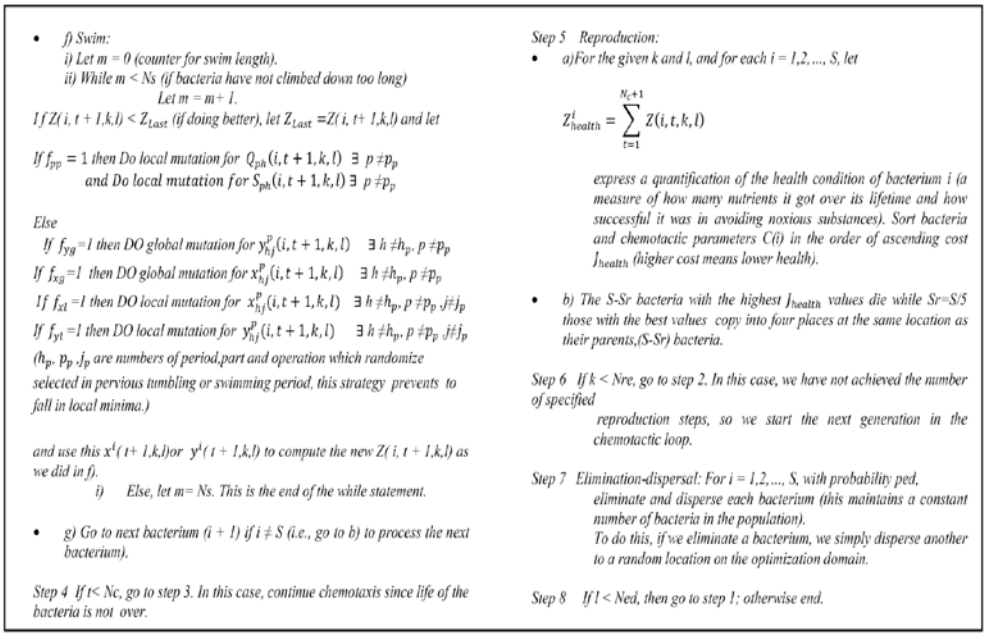
If $p_{ytumble}$=0 and p_{xlocal}>RND then DO local mutation for $x_{hj}^{p}(i,t,k,l)$ and set f_{xl} =1

If $p_{ytumble}$=0 and p_{xlocal}<RND then DO global mutation for $x_{hj}^{p}(i,t,k,l)$ and set f_{xg}=1

- e) Compute Z(i, t + 1,k,l).

:

Figure 7: The proposed BFO algorithm (section 1)



- f) Swim:

  i) Let m = 0 (counter for swim length).

  ii) While m < Ns (if bacteria have not climbed down too long)

  Let m = m+ 1.

If Z(i, t + 1,k,l) < Z_{Last} (if doing better), let Z_{Last} =Z(i, t+ 1,k,l) and let

If $f_{pp} = 1$ then Do local mutation for $Q_{ph}(i,t+1,k,l)$ $\exists$ $p \neq p_p$ and Do local mutation for $S_{ph}(i,t+1,k,l)$ $\exists$ $p \neq p_p$

Else

If f_{yg}=1 then DO global mutation for $y_{hj}^{p}(i,t+1,k,l)$ $\exists$ $h \neq h_p$, $p \neq p_p$

If f_{xg}=1 then DO global mutation for $x_{hj}^{p}(i,t+1,k,l)$ $\exists$ $h \neq h_p$, $p \neq p_p$

If f_{xl}=1 then DO local mutation for $x_{hj}^{p}(i,t+1,k,l)$ $\exists$ $h \neq h_p$, $p \neq p_p$,$j \neq j_p$

If f_{yt}=1 then DO local mutation for $y_{hj}^{p}(i,t+1,k,l)$ $\exists$ $h \neq h_p$, $p \neq p_p$,$j \neq j_p$

(h_p, p_p ,j_p are numbers of period,part and operation which randomize selected in pervious tumbling or swimming period, this strategy prevents to fall in local minima.)

and use this x^i(t+ 1,k,l)or y^i(t + 1,k,l) to compute the new Z(i, t + 1,k,l) as we did in f).

i) Else, let m= Ns. This is the end of the while statement.

- g) Go to next bacterium (i + 1) if i ≠ S (i.e., go to b) to process the next bacterium).

Step 4 If t< Nc, go to step 3. In this case, continue chemotaxis since life of the bacteria is not over.

Step 5 Reproduction:

- a)For the given k and l, and for each i = 1,2,..., S, let

$$Z_{health}^{i} = \sum_{t=1}^{N_c+1} Z(i,t,k,l)$$

express a quantification of the health condition of bacterium i (a measure of how many nutrients it got over its lifetime and how successful it was in avoiding noxious substances). Sort bacteria and chemotactic parameters C(i) in the order of ascending cost J_{health} (higher cost means lower health).

- b) The S-Sr bacteria with the highest J_{health} values die while Sr=S/5 those with the best values copy into four places at the same location as their parents,(S-Sr) bacteria.

Step 6 If k < Nre, go to step 2. In this case, we have not achieved the number of specified reproduction steps, so we start the next generation in the chemotactic loop.

Step 7 Elimination-dispersal: For i = 1,2,..., S, with probability ped, eliminate and disperse each bacterium (this maintains a constant number of bacteria in the population).
To do this, if we eliminate a bacterium, we simply disperse another to a random location on the optimization domain.

Step 8 If l < Ned, then go to step 1; otherwise end.

Figure 7: The proposed BFO algorithm (section 2)

5. COMPUTATIONAL RESULTS

In this study, an efficient algorithm based on the bacteria foraging phenomenon is proposed to solve the integrated dynamic cell formation and process planning problem. The algorithm has been coded in PC MATLAB and tested on a Pentium 4 machine T3400 2GHz with 1 GB RAM. The number of bacteria and number of reproduction steps in the population varied from problem to problem within the range of 5 to 25 and 10 to 1000, respectively. In contrast, the number of chemo tactic steps per bacteria lifetime, the limits of the length of a swim, and the number of elimination–dispersal events based on some designed experiments were fixed at 2, 10, and 5, respectively. These parameters may vary, depending on the decision maker who adjusts the algorithm.

The aim of this section is to verify the applicability of the proposed model through hypothetical examples with randomly-generated data. These examples are solved by the branch and bound (B&B) method using Lingo 8.0 software. They are generated according to the information provided in Table 1. In this table, the term 'U' indicates uniform distribution [20, 21].

The instance 1-related data set is shown in Table 2. Instance 1 consists of five part types, five machine types, and three periods in which each part type is assumed to have three operations that must be processed respectively. Each operation can be performed on two alternative machines. Thus each part type has 2×3=6 process plans, and there are 6^8 combinations to select a process plan for each part type in each period. Therefore there are about $([6+3]^5)^3 = 9^{15}$ combinations (because H=3) for allocating operation-parts to machines throughout the planning horizon [20, 22].

First, the four columns of Table 2 (included at the end of this paper) include the machine information: capacity, constant cost, variable cost, and replacement cost. The quantity of demand for each part type, and the product mix for each period, are given in Table 2. The last rows of Table 2 show the inter- and intra-cell batch size, inventory and backorder costs, and the initial inventory for each part type. After linearisation, the proposed model consists of 1,125 integer variables, 1,575 non-integer variables, and 1,383 constraints in the example under consideration. The best solution was found after 29 seconds (Zbest) in detail, and the cell configurations for all periods corresponding to the best obtained solution from BFO for this instance are reported in Table 3 and shown in Figure 8.

In order to verify the performance of the BFO approach compared with B&B, 10 random instances have been solved according to Table 4. For simplicity, it is assumed that the capacities of the machines are independent of their types, but dependent on the length of the planning horizon [20].

Each problem is allowed to run on Lingo for 10,800 seconds (3 hours). However, because of the computational complexity, the proposed model could not be optimally solved within 10,800 seconds, or even over a longer time for medium- and large-sized instances. Thus, to solve the small- and medium-sized problems whose runtime was more than three hours, the software was interrupted after 3 hours and the best solution at that time was reported.

Parameter	Value	Parameter	Value	Parameter	Value
H	3	D_{ph}	U(100,1000)	δ_m	$\alpha_m/2$
M	(p/2)+2	t_{jpm}	U(0,1)	β_p^{Inter}	U(10,50)
O_p	2	$\sum_m a_{jpm}$	2 $\forall j,p$	β_p^{Intra}	$\beta_p^{Inter}/5$
C	3	α_m	U(1000,2000)	γ^{inter}	50
UB	$\lceil\sqrt{p}\rceil + 1$	β_m	U(1,10)	γ^{intra}	5
λ_p	U(1,5)	η_p	U(1,5)	ρ_p	U(20,30)
I_{0p}	χ^2 (0,max[D_{ph}])				

Table 1: Test problem generation standardisation [20, 22]

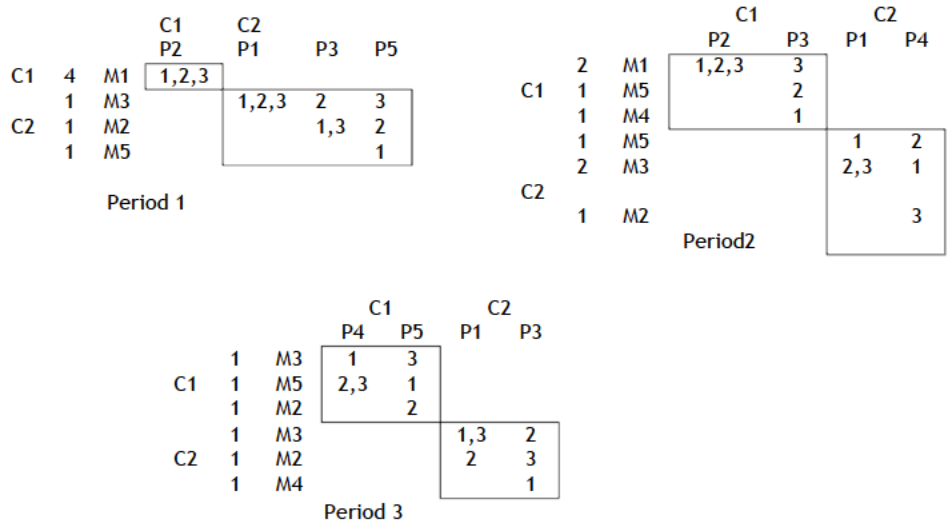


Figure 8: Formed cells resulted from the proposed BFO for instance No.1.

Machine info.					P1			P2			P3			P4			P5		
T_m	α_m	β_m	δ_m		1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
500	1800	9	900	M1	0	0	0	.76	.65	.39	0	0	.46	.49	.83	0	0	0	0
500	1500	7	750	M2	0	.79	0	0	0	0	.99	0	.33	0	0	.74	.40	.63	0
500	1800	5	900	M3	.73	.93	.44	0	0	0	0	.57	0	.45	0	0	0	0	.59
500	1700	9	850	M4	0	0	.46	.80	.81	0	.14	0	0	0	0	0	0	.26	0
500	1300	8	650	M5	.54	0	0	0	0	.93	0	.48	0	0	.67	.62	.12	0	.75
				Period 1		300			850			200			0			250	
D_{ph}				Period 2		400			700			700			200			0	
				Period 3		400			0			450			950			350	
β_p^{inter}						35			25			20			40			45	
β_p^{intra}						7			5			4			8			9	
λ_p						13			12			15			12			11	
η_p						14			13			16			13			12	
ρ_p						40			39			30			36			34	
I_0						0			0			300			150			0	

I =1, C=3 , UB=4

Table 2: Data set for instance No.1

	h=1					h=2					h=3				
	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
Q_{ph}	170	882	70	676	79	689	441	362	26	208	218	89	363	41	313
S_{ph}	23	138	255	162					95						
I_{ph}	-130	32	170	826	-171	182	-90	86	809	37					
D_{ph}	300	850	200	0	250	400	700	700	200	0	400	0	450	950	350

Table 3: Optimal production plans resulting from the proposed BFO for instance No.1

Table 4 shows the comparison between the results obtained using B&B and the proposed BFO, corresponding to 10 small- and medium-sized problems in terms of computational time and the best solution obtained. To verify the performance of the proposed BFO algorithm, each problem has been run 20 times. The average CPU runtime and best solution obtained for each problem are reported in Table 4. As shown in Table 4, the proposed BFO algorithm solved problems in 8% $(861/10{,}440)$ of the computational time of B&B, while qualified solutions show a 6% improvement – i.e., $(497823 - 467743)/497823$.

No.	Problem info.			B&B		BFO	
	P×M×H	C	UB	F^{best}	$T_{B\&B}$	Z^{mean}	T_{BFA}
1	5×5×3	3	4	144612	7205	139720	29.3
2	10×7×3	3	5	211019	10800	199960	135
3	14×9×3	3	5	279306	10800	268040	298
4	18×11×3	3	6	336751	10800	296540	549
5	22×13×3	3	7	540408	10800	478780	586
6	26×15×3	3	8	539170	10800	536770	767
7	28×16×3	3	9	542248	10800	539190	1193
8	28×16×4	3	9	898763	10800	874970	1585
9	30×17×3	3	10	622107	10800	572650	1694
10	30×17×4	3	10	863846	10800	770810	1777
Avg				**497823**	**10440**	**467743**	**861.3**

Table 4: Performance of the proposed BFO compared with B&B in term of approximately the same cost function

If the reproduction steps of BFO are relaxed until they reach the best solutions, then the results in Table 5 can be obtained for all mentioned data sets. As shown in this table, for all data sets, BFO obtained the best results in terms of the amount of cost function and computational time. Standard deviation in terms of the amount of cost function and computational time has been shown. Because of the one tailed p-values used in all instances, in terms of the resulting population, the mean difference between B&B and BFO is equal to zero. Therefore the statistical tests signify a better performance by the BFO algorithm in terms of the amount of the cost function and the computational time, compared with the B&B method.

No.	Problem info.			B&B		BFO					
	P×M×H	C	UB	F^{best}	$T_{B\&B}$	Z^{mean}	T_{BFA}	St Dev Z	St Dev Time	P-Value Z	P-Value Time
1	5×5×3	3	4	144612	7205	126190	160	3803	29	0.0	0.0
2	10×7×3	3	5	211019	10800	179504	612	6107	109	0.0	0.0
3	14×9×3	3	5	279306	10800	227593	1293	7424	218	0.0	0.0
4	18×11×3	3	6	336751	10800	245980	2644	7450	414	0.0	0.0
5	22×13×3	3	7	540408	10800	420800	2794	15490	460	0.0	0.0
6	26×15×3	3	8	539170	10800	437682	3588	16063	606	0.0	0.0
7	28×16×3	3	9	542248	10800	460791	5501	15754	922	0.0	0.0
8	28×16×4	3	9	898763	10800	750374	6445	27441	1135	0.0	0.0
9	30×17×3	3	10	622107	10800	472951	7672	14747	1303	0.0	0.0
10	30×17×4	3	10	863846	10800	698274	7038	26656	1309	0.0	0.0
Avg				**497823**	**10441**	**402014**	**3775**				

Table 5: Performance of the proposed BFO compared with B&B in term of finding the minimum cost function

Table 6 shows the details of statistical test results obtained from the MINITAB software for data set 1 (smallest instance) and data set 10 (largest instance).

Finally, the data set 1 of size 5 × 5 × 3 is taken as an example to illustrate the convergence curve of the proposed BFO during reproduction steps. After 35 reproduction steps, the objective value Z is to be 160,000. It is reduced when the number of reproduction steps

decrease to 20%. At the 100th reproduction step, it reached to the value of 148,000, a decrease of 7.5%. During the 170th reproduction, the Z value is 139,720, a decrease of 5.6%. So it is terminated at this point, because the algorithm reaches approximately the same value around B&B's value. The convergence curve is shown in Figure 9.

Two-Sample *No.1* **T-Test and CI: B&B_Z, BFA_Z** N Mean St Dev SE Mean B&B_Z 20 144612 0.0 0.0 BFA_Z 20 126190 3803 850 Difference = mu (B&B_Z) - mu (BFA_Z) Estimate for difference: 18422 99.999% lower bound for difference: 13637 T-Test of difference = 0 (vs >): T-Value = 21.66 P-Value = 0.000 DF = 19	**Two-Sample *No.1*** **T-Test and CI: B&B_Time, BFA_Time** N Mean St Dev SE Mean B&B_Time 20 7205 0.0 0.0 BFA_Time 20 160.3 29.2 6.5 Difference = mu (B&B_Time) - mu (BFA_Time) Estimate for difference: 7044.70 99.999% lower bound for difference: 7007.99 T-Test of difference = 0 (vs >): T-Value = 1079.67 P-Value = 0.000 DF = 19
Two-Sample *No.10* **T-Test and CI: B&B_Z, BFA_Z** N Mean St Dev SE Mean B&B_Z 20 863846 0.0 0.0 BFA_Z 20 698274 26656 5960 Difference = mu (B&B_Z) - mu (BFA_Z) Estimate for difference: 165572 99.999% lower bound for difference: 132035 T-Test of difference = 0 (vs >): T-Value = 27.78 P-Value = 0.000 DF = 19	**Two-Sample *No.10*** **T-Test and CI: B&B_Time, BFA_Time** N Mean St Dev SE Mean B&B_Time_1 20 10800 0.0 0.0 BFA_Time_1 20 7038 1309 293 Difference = mu (B&B_Time) - mu (BFA_Time) Estimate for difference: 3762 99.999% lower bound for difference: 2116 T-Test of difference = 0 (vs >): T-Value = 12.86 P-Value = 0.000 DF = 19

Table 6: Statistical test results obtained from MINITAB for data sets 1 and 10

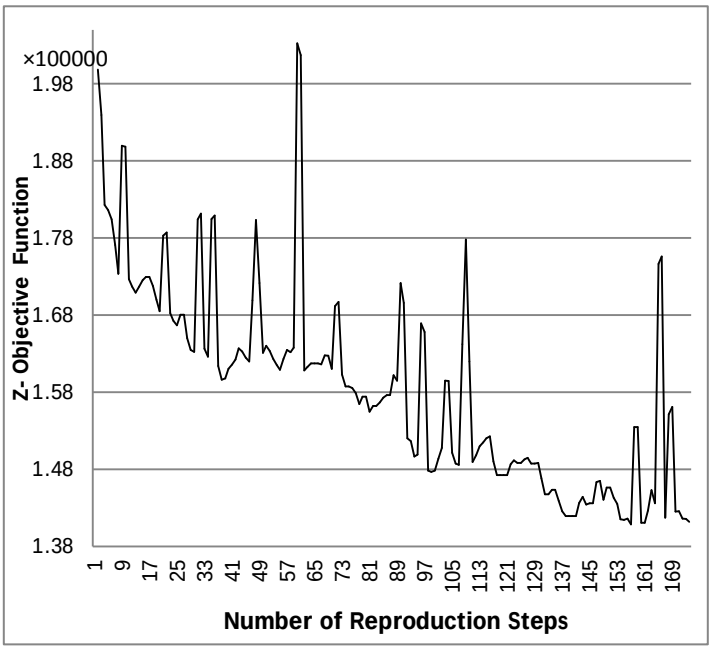


Figure 9: Convergence curve of proposed BFO for data set No.4

6. CONCLUSION

In this paper, an optimisation model of the integrated dynamic cellular manufacturing system (DCMS) and production process planning has been introduced along with a solution approach based on BFO. The advantages of the proposed model are as follows: simultaneous analysis of inter- and intra-cell material handling by assuming the sequence of operations, alternative process plans for part types, machine copying, and process planning costs including inventory, backorder, and partial subcontracting costs.

The main difference between the Branch and Bound method and the proposed BFO – i.e., generating a high quality solution in a shorter computational time – has been presented in this research work. The results show that BFO provides significantly better solutions in a considerably shorter time, where the average gap between the quality of the solution found by BFO and the best solution found by the Branch and Bound (B&B) method has been significant. Also, BFO can be used to generate solutions in evolutionary algorithms, and is suggested for future research that aims to minimise tooling costs, setup costs, and hiring, firing, and salary costs. Such methods emphasise the effect of the trade-off between machine adjacency constraints, workload balancing, product quality, available time of workers, and worker assignment for the reconfiguration of cells in dynamic cellular manufacturing systems.

7. REFERENCES

[1] **Acharya, D.P., Panda, G., Mishra, S. & Lakhshmi, Y.V.S.** 2007. *Bacteria foraging based independent component analysis*, In Proceedings of the international conference on computational intelligence and multimedia applications, Vol. 2, pp 527–531

[2] **Ah Kioon, S. et al.** 2009. *Integrated cellular manufacturing systems design with production planning and dynamic system reconstruction*, European Journal of Operational Research, 192, pp 414–428

[3] **Bajestani, M.A. et al.** 2009. *A multi-objective scatter search for a dynamic cell formation problem*, Computers & Operations Research, 36 2009, pp 777–794

[4] **Baykasogylu, A. & Gindy, N.Z.** 2001. *A simulated annealing algorithm for dynamic layout problem*, Computers & Operations Research, 28, pp 1403-1426

[5] **Boulif, M. & Atif, K.** 2008. *A new fuzzy genetic algorithm for the dynamic bi-objective cell formation problem considering passive and active strategies*, International Journal of Approximate Reasoning, 47, pp 141–165

[6] **Defersha, F.M. & Chen, M.** 2006. *A comprehensive mathematical model for the design of cellular manufacturing systems*, International Journal of Production Economics, 103, pp 767–783

[7] **Defersha, F.M. & Chen, M.** 2008. A linear programming embedded genetic algorithm for an integrated cell formation and lot sizing considering product quality. *European Journal of Operational Research*, 187, 46–69.

[8] **Deljoo, V. et al.** 2010. *Using genetic algorithm to solve dynamic cell formation problem*, Applied Mathematical Modelling, 34, pp 1078–1092

[9] **Jeon, G. & Leep, H.R.** 2006. *Forming part families by using genetic algorithm and designing machine cells under demand changes*, Computers & Operations Research, 33, pp 263–283

[10] **Mahdavi, I. et al.** 2010. *Designing a mathematical model for dynamic cellular manufacturing systems considering production planning and worker assignment*, Computers and Mathematics with Applications, doi:10.1016/j.camwa.2010.03.044

[11]**Majhi, B. & Panda, G.** 2007a. *Bacterial foraging based identification of nonlinear dynamics system,* In Proceedings of IEEE international congress on evolutionary computation (CEC-2007), pp 1636–1641, Singapore.

[12] **Majhi, B. & Panda, G.** 2007b. *Particle swarm optimization based efficient adaptive channel equalizer for digital communication*, In Proceedings of international conference on trends in intelligent electronics systems (2007). Chennai.

[13] **Majhi, B., Panda, G. & Choubey, A.** 2006. *On the development of a new adaptive channel equalizer using bacterial foraging optimization technique*, In Proceedings of IEEE annual India conference (INDICON-2006), pp 1–6, New Delhi, India.

[14] **Majhi, R., Panda, G., Majhi, B. & Sahoo, G.** 2009. *Efficient prediction of stock market indices using adaptive bacterial foraging optimization (ABFO) and BFO based techniques,* International Journal of Expert Systems with Applications 36, pp 10097–10104

[15] **Mishra, S.** 2005. *Hybrid least square adaptive bacterial foraging strategy for harmonic estimation*, IEE Proceedings on Generation Transmission and Distribution, 152(3), pp 379–389

[16] **Mishra, S. & Bhende, C.N.** 2007. *Bacterial foraging technique based optimized active power filter for load compensation,* IEEE Transactions on Power Delivery, 22(1), pp 457–465
[17] **Panda, G., Majhi, B. & Mishra, S.** 2006. *Nonlinear system identification using bacterial foraging based learning,* In Proceedings of third international conference on artificial intelligence in engineering technology (ICAIET-2006), pp 120–125, Kota Kinabalu, Malaysia.
[18] **Passino, K.M.** 2002. *Biomimicry of bacterial foraging for distributed optimization and control*, IEEE Control Systems Magazine, 22(3), pp 52–67
[19] **Rheault, M., Drolet, J. & Abdulnour, G.** 1995. *Physically reconfigurable virtual cells: A dynamic model for a highly dynamic environment*, Computers & Industrial Engineering 29 (1–4), pp 221–225
[20] **Safaei, N. et al.** 2008. *A hybrid simulated annealing for solving an extended model of dynamic cellular manufacturing system*, European Journal of Operational Research 185, pp 563–592
[21] **Safaei, N. et al.** 2008. *A fuzzy programming approach for a cell formation problem with dynamic and uncertain conditions*, Fuzzy Sets and Systems 159, pp 215 – 236
[22] **Safaei, N. et al.** 2009. *Integrated multi-period cell formation and subcontracting production planning in dynamic cellular manufacturing systems*, International Journal of Production Economics 120, pp 301–314
[23] **Seifoddin,H. & Djassemi, M.** 1996. *Improving the performance of cellular manufacturing by a dynamic part assignment approach*, Computers in Industrial Engineering, 30(4), pp 719-726
[24] **Tavakkoli-Moghaddam, R. & Aryanezhad, M.B.** 2005. *Solving a dynamic cell formation problem using metaheuristics*, Applied Mathematics and Computation 170, pp 761–780
[25] **Tripathy, M. & Mishra, S.** 2007. *Bacteria foraging based to optimize both real power loss and voltage stability limit*, IEEE Transactions on Power Systems, 22(1), pp 240–248
[26] **Tripathy, M., Mishra, S., Lai, L.L. & Zhang, Q.P.** 2006. *Transmission loss reduction based on FACTS and bacteria foraging algorithm*, *PPSN*, pp 222–231
[27] **Ulagammai, L., Venkatesh, P., Kannan, S.P. & Padhy, N.P.** 2007. *Application of bacteria foraging technique trained and artificial and wavelet neural networks in load forecasting*, Neurocomputing, pp 2659–2667