World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:9, No:10, 2015

Enhanced Imperialist Competitive Algorithm for the Cell Formation Problem Using Sequence Data

S. H. Borghei, E. Teymourian, M. Mobin, G. M. Komaki, S. Sheikh

***Abstract*—**Imperialist Competitive Algorithm (ICA) is a recent meta-heuristic method that is inspired by the social evolutions for solving NP-Hard problems. The ICA is a population-based algorithm which has achieved a great performance in comparison to other meta-heuristics. This study is about developing enhanced ICA approach to solve the Cell Formation Problem (CFP) using sequence data. In addition to the conventional ICA, an enhanced version of ICA, namely EICA, applies local search techniques to add more intensification aptitude and embed the features of exploration and intensification more successfully. Suitable performance measures are used to compare the proposed algorithms with some other powerful solution approaches in the literature. In the same way, for checking the proficiency of algorithms, forty test problems are presented. Five benchmark problems have sequence data, and other ones are based on 0-1 matrices modified to sequence based problems. Computational results elucidate the efficiency of the EICA in solving CFP problems.

***Keywords*—**Cell formation problem, Group technology, Imperialist competitive algorithm, Sequence data.

I. Introduction

CELLULAR MANUFACTURING SYSTEM (CMS) is one of the famous manufacturing systems which uses Group Technology (GT) concept to merges flexibility of job shops and high production rate of lines environments. The CMS take the advantages of reduced setup times and work in process, improved product quality, shorter lead times, less tool requirements, enhanced productivity, better overall control of operations, etc., (see [20], [22]). The well-known problem in CMS that seeks to group similar parts/components and different machines needed for processing these components in a same cell, called cell formation problem (CFP). The goal of this problem is finding such cells in order to optimize the chosen performance measures. There are various types of CF problems considering different aspects of cellular manufacturing systems, including: workers, products route, scheduling, layout, etc., for more information one can refer to [14], [21].

H. Borghei is with the department of Industrial Engineering, Mazandaran University of Science and Technology, Babol, Iran (e-mail address: hborgheiir@gmail.com)

E. Teymourian is with the School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR 97331-6001, USA (e-mail address: ehsan.teymorian@gmail.com)

M. Mobin is with the Department of Industrial Engineering and Engineering Management, Western New England University, Springfield, MA 01119 USA (Phone: 413-801-7845; e-mail: mm337076@wne.edu)

M. Komaki is with Case Western Reserve University, Cleveland, OH 44106 USA (phone: 216-368-4114; e-mail: gxk152@ case.edu).

S. Sheikh is with the Department of Management Science at New York Institute of Technology, New York, NY 10023 USA (e-mail: shaya.sheikh@case.edu).

The basic form of CFPs is one that uses binary machine-component incident matrices, in which, components must be processed on related machines, where the corresponding array is 1. Many researchers have mentioned in their works [24], [27] that CFPs are NP-hard, and then cannot be solved optimally in real and large sizes. Therefore, heuristic and meta-heuristic approaches have been considered thoroughly to solve the CFPs in reasonable computational times. In this sense, [8] used Genetic Algorithm (GA) to solve the CFP. Wu et al. [28] presented a Simulated Annealing (SA) algorithm for these problems and called it SACF; the algorithm improves the grouping efficacy in most of test problems. Wu et al. [26] have introduced a heuristic algorithm applying water flow-like algorithm (WFA) logic to solve the CFP called WFACF. WFA is mimicking water flowing behavior from higher to lower levels and dynamically changing number of flows in this movement. That is, changing number of water flows make WFA an agent population-varying method. Recently, [5] designed a hybrid GA that uses large neighborhood search and GA together.

Studies based on binary machine-component incident matrices do not contain processing routes. This information is critical in real world; therefore, incident matrices that demonstrate the sequence of operations to finish a component have been considered as other main form of CFPs. These kinds of CFPs try to reach a feasible solution with least inter-cell and/or intra-cell travels movements. Some researchers have considered this vital information in solving CF problems. Nair and Narendran [18] developed a new clustering method (CASE) and some new performance measures for CFP using sequence data. Spiliopoulos and Sofianopoulou [24] employed an efficient ant colony approach to deal with similar problems. A heuristic based on bacteria foraging algorithm (BFA) called BASE is presented by [20]. BASE was compared with CASE and modified ART (ART1 in [20]), and showed better performance of proposed algorithm. Mahdavi and Mahavedan [15] have introduced a heuristic called CLASS that not only identifies components families and machines groups but also outlines the layout (sequence) of the machines within each cell. Moreover, they compared their work with solutions of the CASE. More studies and articles about CFPs, solving methods and performance measures can be found in [21].

Imperialist Competitive Algorithm (ICA) is one of the population-based algorithms presented by [2] to solve a problem in continues fashion. This algorithm has been also recently adopted to exploit solution space of discrete problems, among them you can refer to: [3], [6], [10], [12], [17], [23]. On the other hand, good capability of ICA dealing

with various types of engineering optimization problems has studied, e.g., scheduling: [3], [6], [10], [17], [23]; balancing and sequencing problem: [12], [29]; process planning: [13]; portfolio optimization: [25]; skeletal structures: [9]; outsourcing: [19]; quadratic assignment problem: [16], etc.

In this study, ICA is adapted to solve the CF problem using sequence data. The proposed ICA contains two types of moves, designed in move-toward procedure. The ICA has a good ability to guide the searching agents (countries/colonies) within the solution space with a satisfactory convergence speed and then obtains good quality solutions itself; while local search (LS) concept may be included in the proposed algorithm to improve its performance. That is, an iterative local search is embedded in enhanced ICA, using different tested search strategies which bring good ability to the main algorithm in refining the explored regions of the solution space. The ICA in EICA plays the global search role, whereas the integrated LS mechanism applies the local search to balance the intensification and diversification aspects of the search.

Proposed algorithms, adopted ICA and EICA, solve two different sets of test problems; (1) benchmark sequential incident matrices and (2) modified binary incident matrices. Besides, bond efficiency (β) and group technology efficiency (GTE) are two of well-known measures applied for SFPs with sequence data, which are considered in this study.

The remainder of paper is organized as follows: Section II explains the original imperialist competitive algorithm (ICA). The proposed enhanced ICA for the CFP using sequence data are presented in Section III. Section IV includes the computational results which demonstrate and compare the efficiency of our algorithms. Finally, conclusion remarks are given in Section V.

II. Original Imperialist Competitive Algorithm

Imperialist competitive algorithm (ICA) is a social inspired algorithm that uses the concept of imperialism. Imperialism is a policy of extending power and rule of a government beyond its own boundaries [2]. ICA is known as a population-based algorithm in which each solution is called country. It simulates different aspects of a country like culture, religion, military power, art, and so on. Thus, total cost of a country contains all these factors. ICA works with the power of countries or their scores based on corresponding imposed costs. As explained in [23], countries are divided into imperialists and colonies regarding their powers. Colonies are distributed among imperialists to create empires. Empires compete to conquer more colonies; weak empires lose their colonies and they collapse at last. This collapse mechanism will hopefully cause all countries to converge to a state where just one empire will be survived, whereas all other countries become the colonies of that empire. This situation means that there is no more competition, and algorithm is ended.

Procedure and formulations of original ICA in this paper are like [2] the architect of ICA. Based on cost of countries, they are divided into colonies and imperialists; imperialists are the most powerful countries and rules over other countries (colonies). An imperialist and its colonies are called an empire too. Imperialists with more power will have more chance to govern more colonies. Power of each imperialist is calculated by normalized cost$\left(C_n\right)$ as (1):

$$C_n = \max_i \{c_i\} - c_n \tag{1}$$

where c_n is the cost of nth imperialist and $\max_i \{c_i\}$ is maximum cost of all imperialists. Probability and number of colonies that an imperialist will rule are calculated by (2), (3):

$$\text{Probability} = \frac{C_n}{\sum_i C_i} \tag{2}$$

$$\text{Number of colonies in an empire} = round\left\{\text{Probability} \times \text{Number of all colonies}\right\} \tag{3}$$

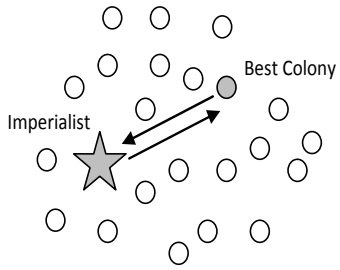

Fig. 1 Best colony becomes imperialist

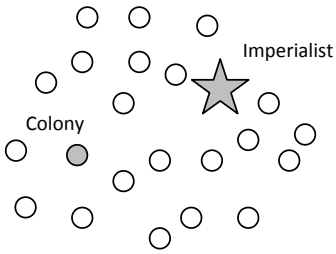

Fig. 2 New empire after exchanging

The colonies are forced to be like their imperialist, so they are moved toward their imperialist to resemble their leader in different aspects. Any time any of colonies reaches a higher score in its empire, it becomes the new imperialist itself (Figs. 1 and 2).

An empire score depends on two factors: imperialist score and colonies' score, in (4).

$$\text{Total cost of an empire} = Cost_{imprialist} + \left[\xi \times mean\left(Cost_{colonies\ of\ empire}\right)\right] \tag{4}$$

In (4), ξ is a predefined coefficient to impose the percentage effect of colonies' cost on empire's cost. The cost of empires will be normalized by (5); these normalized costs are used as their power (score).

$$NTC_n = \max_i \{TC_i\} - TC_n \quad (5)$$

Empires compete and some of them are getting weakened. Ergo, they lose their colonies until they will be eliminated, and one empire ultimately remains. Atashpaz-Gargari and Lucas [2] have presented a new Roulette Wheel method to choose the empire which will possess the weakest colony of the weakest empire in each evolution (iteration of the main ICA procedure). Computational effort of this method is much less than the conventional Roulette Wheel. Equation (6) shows the weights calculated for each empire.

$$\text{Weight of empire} = \frac{NCE_n}{\sum_i NCE_i} \quad (6)$$

$$W = \{w_1, w_2, ..., w_i\}$$

Then, a random vector, the same size as the empires, is generated between 0 and 1 (R).

$$R = \{r_1, r_2, ..., r_i\}$$

Finally, a new vector will be calculated by (7):

$$D = W - R \quad (7)$$

$$D = \{d_1, d_2, ..., d_i\}$$

where the biggest d_i shows the best empire that occupies the weakest colony of the weakest empire. According to [2], ICA flowchart is shown in Fig. 3.

As far as we know, ICA has never been used to solve CFP using sequence data. Since CFPs are discrete type problems, some adaptation on original ICA has been done to achieve a suitable algorithm. In this paper, we provide a new meta-heuristic method step by step to find better solutions for the considered CFP.

III. Enhanced ICA for the CFPs

In this section, some modifications are done on the original ICA so as to be capable for solving the CFP using sequence data. To illustrate the proposed algorithm, we gradually follow the basic steps of the ICA as given in [23]. The original ICA was not designed for problems with discrete space. So, we first turned the ICA to an algorithm which can deal with the CFP's solution space, and then we also add a new local search procedure to help the algorithm to find better solutions, called EICA. There are different parameters in this algorithm that affect on quality of solutions. Like most other evolutionary algorithms, ICA is concerned with *size of population* (*Popsize*). Other main parameters of ICA are *Number of empires* or *number of imperialists* (*NumImp*), *colonies participant in total cost of empire* (ξ), predefined probability for choosing the local search type (α), *Number of evolutions* and *Number of local searches*.

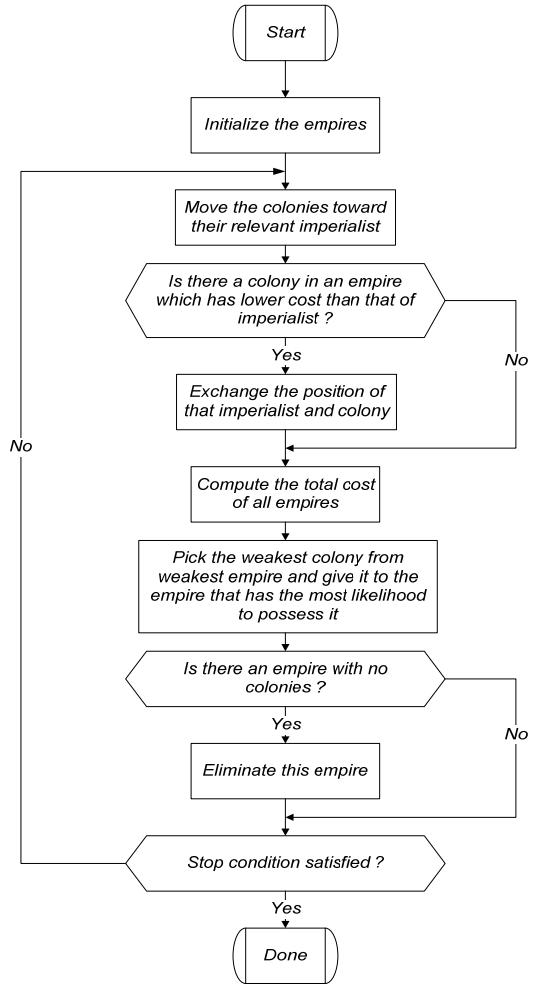

Fig. 3 Original ICA Flowchart presented in [2]

A. Representation

Different representations have been developed for CFP's solution. For instance, [4] used a suitable representation for their algorithm CF-GGA: a grouping genetic algorithm for the cell formation problem. In our paper, representation method is a string of numbers that show components, machines and their corresponding cells. ICA solutions called countries. Based on CF problem, a country includes components, machines and cells. For example, suppose six components going to be processed on four machines in tree cells. Component 1 is assigned to cell number 2, Machine 1 is located in cell number 3, and the other components and machines are assigned to their cell's number as illustrated in Fig. 4.

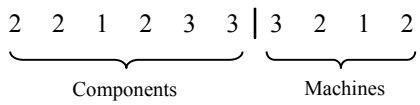

Fig. 4 A feasible solution representation

B. Generating Initial Empires (Population)

As previously mentioned, ICA is a population-based

algorithm; so it should generate *Popsize* number of countries in the beginning of algorithm procedure. Each solution must satisfy minimum and maximum limits on the number of components in a cell (*LnC* and *MnC*). There are also similar restrictions for machines (*LnM* and *MnM*). Whenever a solution violates these limits, it is defined as an infeasible one and needs to be fixed. Fig. 5 shows an infeasible solution where there is no component, nor any machine in Cell 1. In this figure, rows and columns represent components and machines respectively. In addition, ordinal numbers show the number of processes for each component and highlighted block-diagonals refer to the manufacturing cells. To generate least infeasible solutions in initial population, number of components/machines for each cell is determined based on predefined limits, and the algorithm accordingly assigns components/machines to considered cells. So, it creates a string of cells number. Generated countries will be evaluated and divided into imperialists and colonies. After that, colonies and their related imperialist will create empires. During the search procedure or initial population generating, infeasible solutions may be created. Here, a repairing mechanism as described follows will be utilized.

Infeasible solution: 2 3 2 2 2 3 | 3 2 2 2

	2	**3**	**4**	**1**
1	0	3	2	1
3	0	1	0	2
4	0	1	3	2
5	1	2	0	0
2	1	0	0	0
6	0	2	1	0

Fig. 5 An infeasible solution

C. Evaluation

In ICA, total cost of any country contains different aspects or parameters (e.g. culture, religion, etc.). Total cost or fitness function in our algorithm contains two elements: void elements (VE) and inter-cell travels (ICT) ((8)). Nouri et al. [20] have also tried to minimize the *VE* and *ICT*.

$$c_n = VE_n + ICT_n \tag{8}$$

After evaluation of colonies, ICA uses total cost of imperialists to calculate the normalized cost by (1). The algorithm finds the most powerful countries and names them *imperialists,* then allots other countries to them (*NumImp* imperialists) based on their power. The countries belonged to the imperialists called *colonies*. An imperialist and its colonies are defined as an *empire* together.

D. Moving the Colonies of an Empire toward Imperialist (Assimilating)

Every imperialist seeks to improve its colonies and as a result improves its empire. Two different ways to move colony members toward imperialists have designed here. Approach one focuses on reducing the void elements; for this reason, a number of components change their cells to imperialist pattern (Fig. 6). For example, components number 1, 3, 4 and 6 are selected to be in a cell that imperialist says. Other approach of moving diminishes the void elements and inter-cell travels as well. Hence, some components and their related machines gathered into one cell. Consider Colony B where component number 1 is going to process on machines number 1, 2 and 4. It is moving toward its imperialist (Fig. 7).

Imperialist: 2 2 1 2 3 3 | 3 2 1 2

Colony *A*: 1 3 2 1 1 2 | 2 3 1 2

New colony *A*: 2 3 1 2 1 3 | 2 3 1 2

Fig. 6 Moving-toward for colony *A* considering voids

Imperialist: 2 2 1 2 3 3 | 3 2 1 2

Colony *B*: 1 3 2 2 1 3 | 2 3 2 1

New colony *B*: 2 3 2 2 1 3 | 3 2 2 2

Fig. 7 Moving-toward for colony *B* considering voids and inter-cell travels

E. Local Search Operators

A new phase (local search) is added to the proposed ICA which did not exist in basic ICA. The LS mechanism has shown a good ability to provide search heuristics with better intensification characteristics [1].

The ICA is a social adopted algorithm and uses different social and political concepts. In like manner, local search (LS) resembles revolution in countries that has a great effect on different aspects and cost parameters of them. Integration of LS strategies into the ICA could control the balance between diversification and intensification in the EICA. The LS operators are exactly same as operators used in [28]: a single-move and an exchange-move. These changes happen with a probability of α in our proposed algorithm, and they are used in both components and machines to find better solutions.

Single-move: Considering the solution presented in Fig. 8; algorithm chooses component number 3, and transfers it from cell 1 to cell 2.

Exchange-move regarding another solution; algorithm chooses two components and exchanges their cells (Fig. 9). These operators can also be established on machines grouping of cell formation. When local searches execute, the algorithm checks whether the new colony is more powerful than the old one. If yes, the new colony replaces the old one. Otherwise, nothing happens. Fig. 10 represents the local search pseudo code in the proposed EICA.

The proposed algorithm is run with three different local search strategies and different *Number of local searches* (*NumLS*). Local search strategies are as fallows. *Random local search operator*: after moving-toward imperialist section, random local search operator chooses one of the local search

operators and uses it for each colony. *Best of all local searches*: this form of local search does several local searches and evaluates all of new colonies. The most powerful new colony will be compared with the old colony and decision will be made to replace the new one or not. *First better of local searches*: algorithm starts to perform a fix number of local searches until it finds a better solution than the old one.

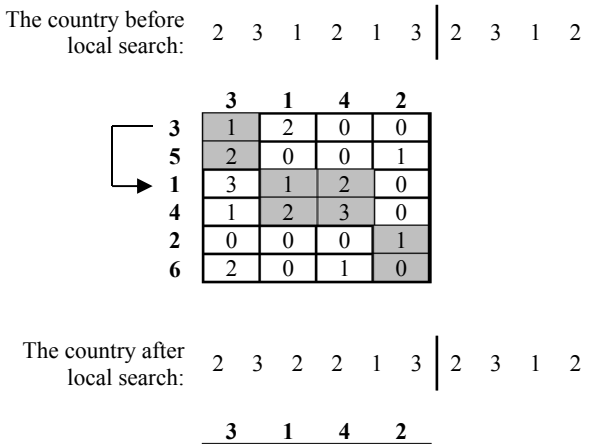

Fig. 8 Single-move local search

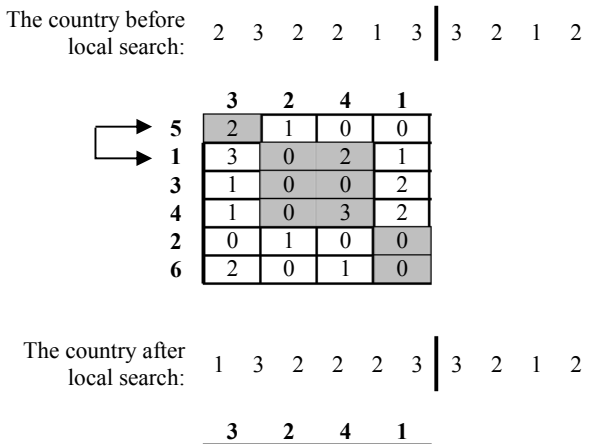

Fig. 9 Exchange-move local search

Procedure *Local search*

for each selected country **do**
r := rand (0,1);
if r <= α (predefined probability for choosing the local search … type) **do**
Single-move operator on its components;
Single-move operator on its Machines;
else
Exchange-move operator on its components;
Exchange-move operator on its Machines;
end if;
end for;

Fig. 10 Pseudo code of local search

F. Repairing Mechanism

After all changes like moving-toward and local search operators, countries maybe exceed the limits and bring about infeasible solutions. Other kind of infeasible solutions are ones in which there are no components or any machines in a cell. There are two strategies to deal with this situation. The first one is to assign a very high penalty for these solutions and as a result reduce their probability of being selected as new imperialist or solutions that create other solutions. The second approach is trying to fix the expected number of components and machines in cells. In small size problems that are more likely to create infeasible solutions, the second approach is used. Increasing size of problems, the latter is no more useful and needs exponential amount of computational effort; thus, the former is applied. Instead of these two approaches, operators can be designed so to only generate feasible solutions.

There are several studies which have been considered some kinds of repairing operators. Spiliopoulos and Sofianopoulou [24] proposed a heuristic method that always put a minimum number of components and machines in each cell. Brown and Sumichrast [4] repaired solutions and called it replacement operator that increases machines utilization. Keeling et al. [11] also developed a similar operator to make infeasible solutions feasible.

Repairing procedure needs some information to show that each machine and component cause how many void elements and inter-cell travels in each solution. Before colonies move toward imperialist, each colony is checked and components and machines situation are updated. Updated situations are used for further changes and repairing. The algorithm employs this information to transfer components or machines to better cells in order to avoid infeasible solutions and improve them as well. To illustrate how repairing operators work, an infeasible solution is figured out in Fig. 5. In considered solution, Cell 1 is empty; that is, there are no machines or components in that cell and the limit number of components/machines is not satisfied. Therefore, at least one of the components must be moved to the mentioned cell, and the algorithm makes a decision based on components situation. While Cell 2 has the most components, thus one of its components must be shifted to Cell 1. Components 1, 3 and 4 create the most number of void elements and inter-cell travels, so one of these components is better to leave Cell 2

(e.g. component 3 can move to Cell 1 as illustrated in Fig. 11).

An infeasible solution: 2 3 1 2 2 3 | 3 2 2 2

	2	3	4	1
3	0	1	0	2
1	0	3	2	1
4	0	1	3	2
5	1	2	0	0
2	1	0	0	0
6	0	2	1	0

Fig. 11 The solution is still infeasible after components repairing

LnC and *MnC* can be satisfied in the same way, but there is still a problem; cell number 1 needs machines to process components. Therefore, the algorithm must move at least one machine to Cell 1, now the machine should be determined. After components repairing, machines 4 and 1 cause the most number of void elements and inter-cell travels. Thus, machine 4 is instantly chosen to move to Cell 1 (Fig. 12). Pseudo code of repairing mechanism is presented in Fig. 13.

The feasible solution: 2 3 1 2 2 3 | 3 2 2 1

	4	2	3	1
3	0	0	1	2
1	2	0	3	1
4	3	0	1	2
5	0	1	2	0
2	0	1	0	0
6	1	0	2	0

Fig. 12 Feasible solution after component and machine repairing

G. Exchanging Positions of Imperialist and a Colony

After moving-toward and local search steps, all colonies and their imperialist are evaluated again to identify that if any colony gets better than its imperialist. The better colony will become the new imperialist of considered empire and the previous one will be an ordinary colony like others at the end of evolution (iteration). Colonies of the empire now will move toward the new imperialist.

```
Procedure Repairing
for each country which must be repaired do
        #Components repair#
     while (MnC <= number of components in each cell <= LnC ) do
            Rearrange the components based on their situation
     end while;
       #Machines repair#
     while (MnC <= number of machines in each cell <= LnC ) do
            Rearrange the machines based on their situation
     end while;
end for;
```

Fig. 13 Pseudo code of repairing mechanism

H. Total Power of an Empire and Empires Competition

Total power of an empire is calculated based on imperialist power and its colonies (see (4)). After each iteration, the empires' power changes and the weakest empire losses its weakest colony. This colony will join to another empire based on their power and chance.

```
EICA for manufacturing cell formation problems using sequence data
begin
    initialize parameters for ICA;            #Table I#
    generate random solutions (colonies);
    Evaluate colonies and initialize the empires;
    current number of empires:=initial number of empires;
    while stopping criteria (running time or number of evolutions)
    is not met do
        for i=1 to current number of empires do
            current number of colonies in empire:= number of
            colonies in empire i;
            for j=1 to current number of colonies in empire i
            do
                Move the colonies toward the imperialist;
                #Assimilating#
                procedure Repairing;
                procedure Local search;
                procedure Repairing;
                evaluate colonies;
            end for;
            update the weakest colony;
            If (there is a colony in empire which has lower
            cost than imperialist) do
                exchange the position of that colony and the
                imperialist;
            end if;
            update the empire power;
        end for;
        Pick the weakest colony/colonies from the weakest
        empire and give it/them to the empire that has the most
        likelihood to possess it;     #imperialist competition#
        Eliminate the powerless empire(s);
        If there is just one empire (the main criterion of ICA)
        do
        stop;
        end if;
    end while;
end;
```

Fig. 14 Pseudo code of proposed EICA

I. Eliminating the Powerless Empires (Imperialist Competition)

In ICA, the algorithm continues until empires lose all their colonies and just one empire remains. Section II described the procedure of moving weakest colony of weakest empire to one of other empires, as well as the new Rolette Wheel mechanism. The pseudo code of proposed EICA for the CFP is presented in Fig. 14.

Besides the primary termination criterion of ICA, one remaining empire, we can use or define some criteria for this algorithm such as number of iterations (number of evolution) or CPU time too.

IV. Computational Results and Discussions

A. Parameters Tuning

Based on the problems we deal with, different levels of parameters are needed. There are many parameters in our proposed algorithms which affect the solutions; these

parameters are presented in Table I.

TABLE I
VALUES USED FOR ICA PARAMETERS

Parameter	Level
Zeta (ξ)	0.1
Alpha (α)	0.55-0.6
Number of imperialists (NumImp)	10-30*
Number of counties (Popsize)	100-600*
Number of evolutions	100-500*
Number of local searches	10-20*

* First values are used for small size test problems and greater values for bigger size problems

B. Experimental Results and Performance Measures

Some of performance measures can be found in [18] that shows how measures work, and use them in a grouping genetic algorithm (GGA). Some of these measures are practical for the CFPs dealing with the binary incidence matrices and some others are suitable for the CFPs using sequence data [18]. Bond efficiency (β) and grouping technology efficiency (*GTE*) calculation procedures are presented in [18] and [20]; despite this citing, the procedures are explained again here.

For calculating *GTE*, maximum number of inter-cell travels possible (I_p), and number of inter-cell travels required by the system (I_r) is calculated by (9) and (10), respectively.

$$I_p = \sum_{j=1}^{N} (n-1) \tag{9}$$

$$I_r = \sum_{j=1}^{N} \sum_{w=1}^{n-1} t_{njw} \tag{10}$$

So, (11) calculates the grouping technology efficiency (*GTE*):

$$GTE = \frac{I_p - I_r}{I_p} \tag{11}$$

where, $t_{njw} = 0$ if the consecutive operations *w* and *w* + 1 are performed in the same cell; = 1 otherwise. *n*= number of operations (*w*= 1, 2, 3,…, *n*). *N*= number of components.

The second measure (β) needs compactness of the system in addition to *GTE*. This value is defined by (12), while the measure itself is calculated by (13).

$$Compactness = \frac{\sum_{k=1}^{c} TOTOP_k}{\sum_{k=1}^{c} (TOTOP_k + NOP_k)} \tag{12}$$

where $TOTOP_k$ = Total number of operations in the *k*th cell. NOP_k = Total number of non-operations (void elements) in the *k*th cell.

$$\beta = q(GTE) + (1-q)\, Compactness \quad , \quad (0 < q < 1) \tag{13}$$

TABLE II
TEST PROBLEMS

Prob. No.	Source	Cells	Components	Machines
1	King and Nakornchai [30])	2	7	5
2	Waghodekar and Sahu [45]	2	7	5
3	Kusiak [46]	2	8	6
4	Nair, G. Jayakrishnan and Narendran, T. T. [18]	3	7	7
5	Kusiak and Chow [31]	3	11	7
6	Boctor [47]	3	11	7
7	Seiffodini [32]	2	18	5
8	Seiffodini and Wolfe [51]	3	12	8
9	Mosier and Taube [33]	3	10	10
10	Sudhakara Pandian R, Mahapatra SS. [48]	3	12	10
11	Chan and Milner [34]	3	15	10
12	Chandrasekaran and Rajagopalan [35]	3	20	8
13	Chandrasekaran and Rajagopalan [36]	2	20	8
14	Nair et al. [18]	3	20	8
15	Askin and Subramanian [37]	5	23	14
16	Stanfel [49]	5	24	14
17	McCormick et al. [50]	6	24	16
18	Mosier and Taube [38]	5	20	20
19	Nair et al. [18]	5	20	20
20	Carrie [39]	6	24	18
21	Kumar et al. [40]	5	23	20
22	King [41]	5	43	16
23	Carrie [39]	4	35	20
24	Boe and Cheng [42]	5	35	20
25	McCormick et al. [50]	4	27	27
26	Chandrasekaran and Rajagopalan [43]- Matrix5	9	40	24
27	Chandrasekaran and Rajagopalan [43] - Matrix7	9	40	24
28	Spiliopoulos and Sofianopoulou [24]	8	40	24
29	Nair et al. [18]	8	40	25
30	Chandrasekaran and Rajagopalan [44]	10	100	40

The proposed algorithms were tested on several problems. Results of all runs were organized into two sub sections: *benchmark problems* and *modified problems*.

C. Test Problems

Most heuristic approaches for the CFP using sequence data have been solved the *benchmark problems* presented in [18]; furthermore, 25 more problems that are different are presented in this paper to check the proposed ICA algorithms. These problems are based on binary incident component-machine matrices that their solutions are available in [7]. To modify these binary problems, we employ the same method applied in [24]. Whole test problems in this research are shown in Table II.

TABLE III
PERFORMANCE OF THE PROPOSED ICAS COMPARED WITH CASE, ART AND BFA ON BENCHMARK INSTANCES

CASE						
4	**3**	**6**	**53.85**	**76.92**	NA	Basic ICA
10	-	-	-	-	-	EICA
14	**10**	**17**	**58.54**	**79.27**	**0.03**	EICA
19	15	19	67.80	73.90	0.17	EICA
29	35	37	60.21	71.63	0.45	EICA
ART						
4	**3**	**6**	**53.85**	**76.92**	**0.08**	Basic ICA
10	4	8	69.23	76.08	NA	EICA
14	**10**	**17**	**58.54**	**79.27**	0.29	EICA
19	16	18	69.49	NA	0.55	EICA
29	35	37	60.21	71.63	0.95	EICA
BFA						
4	6	5	61.54	74.52	0.12	Basic ICA
10	**5**	**5**	**80.8**	**80.63**	NA	EICA
14	**10**	**17**	**58.54**	**79.27**	0.12	EICA
19	**19**	**16**	**72.88**	**73.90**	**0.56**	EICA
29	**36**	**35**	**62.40**	**71.92**	5.4	EICA
Proposed ICAs*						
4	**3**	**6**	**53.85**	**76.92**	0.10	Basic ICA
10	**5**	**5**	**80.08**	**80.63**	**0.23**	EICA
14	**10**	**17**	**58.54**	**79.27**	0.27	EICA
19	16	18	69.49	73.30	2.12	EICA
29	**36**	**35**	**62.40**	**71.91**	**5.44**	EICA

D. Benchmark Problems

This set of well-known benchmark problems has been used multiple times in literature of solving the CFPs. Summary of results for these problems is presented in Table III; some of information about CASE [18], Modified ART [8] named ART1 [20], and BFA [20] results are collected from [20] compared with best solutions found by the proposed ICA algorithms.

Table III contains number of exception elements (*EE*), inter-cell travels (*ICT*) and two performance measures: *group technology efficiency* (*GTE*) and *bond efficiency* (β) for each algorithm. In addition to all these information, *CPU time* for considered algorithms is presented.

Bold values in Table III show which algorithm is better in performance measures; furthermore, the highlighted values identify the algorithm which found the best solution in least elapsed time. As Table III points out, proposed algorithms in this paper have a great performance. Our methods can reach the best-known solutions presented in [20]; even in test problems 10 and 29, they find the best-known solutions in less time.

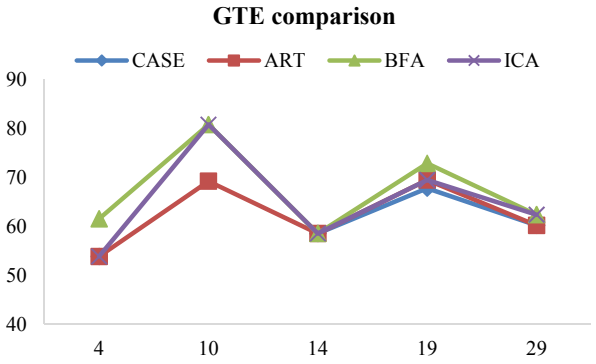

Fig. 15 Comparison of *GTE* performance measure of different methods for the benchmark instances

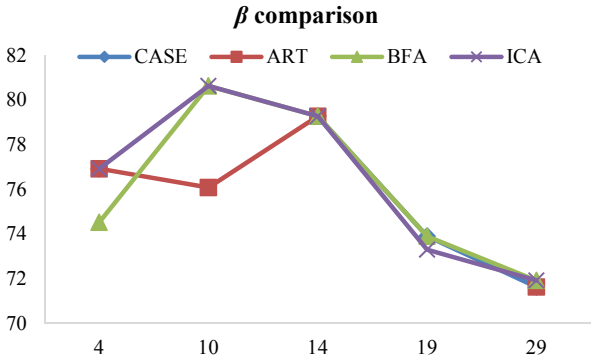

Fig. 16 Comparison of β performance measure of different methods for the benchmark instances

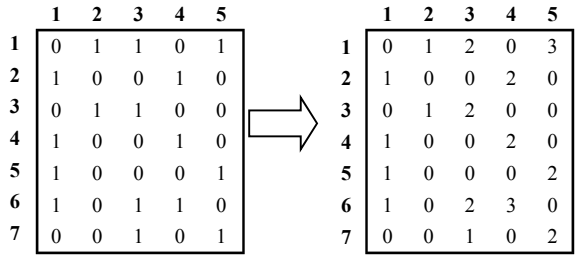

Fig. 17 Binary incidence matrix modification

E. Modified Problems

The method used to modify the binary incidence matrices to sequential matrices is so simple; it assumes that the processing sequence of each component increases with machine number (Fig. 17) [24].

TABLE IV
COMPARISON OF COMPUTATIONAL RESULTS OF THE BASIC ICA VS. EICA ON MODIFIED PROBLEMS

Prob. No.	Best found solution				Basic ICA						EICA					
			GTE					*β*			*GTE*			*β*		
	EE	ICT	*GTE*	*β*	≤1 sec	5 sec	10 sec	≤1 sec	5 sec	10 sec	≤1 sec	5 sec	10 sec	≤1 sec	5 sec	10 sec
1	2	3	66.67	74.5	**100%**	**100%**	**100%**	99%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
2	5	7	46.15	64.7	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	99%	**100%**	**100%**	99%	**100%**	**100%**
3	2	3	78.57	84.7	99%	**100%**	**100%**	99%	**100%**	**100%**	98%	**100%**	**100%**	98%	99%	**100%**
5	6	6	50	56.3	99%	99%	**100%**	98%	99%	99%	99%	99%	**100%**	99%	**100%**	**100%**
6	2	2	80	78	**100%**	**100%**	**100%**	99%	**100%**	**100%**	99%	**100%**	**100%**	99%	**100%**	**100%**
7	7	14	50	71.4	99%	99%	**100%**	**100%**	**100%**	**100%**	99%	**100%**	**100%**	**100%**	**100%**	**100%**
8	7	6	73.91	78.1	99%	**100%**	**100%**	99%	**100%**	**100%**	97%	**100%**	**100%**	97%	**100%**	**100%**
9	0	0	100	86.4	99%	**100%**	**100%**	99%	99%	**100%**	99%	99%	**100%**	99%	99%	**100%**
11	0	0	100	96	98%	99%	99%	98%	99%	99%	98%	99%	**100%**	98%	**100%**	**100%**
12	9	12	70.73	85.4	97%	98%	99%	97%	98%	99%	95%	99%	**100%**	95%	99%	**100%**
13	27	43	39.44	58.7	98%	99%	**100%**	98%	99%	99%	97%	98%	99%	97%	99%	99%
15	7	7	80	78.6	97%	99%	**100%**	96%	98%	99%	96%	99%	**100%**	97%	99%	**100%**
16	9	8	78.38	78.6	98%	99%	**100%**	98%	99%	99%	97%	**100%**	**100%**	97%	99%	**100%**
17	33	42	31.15	54	97%	98%	99%	97%	98%	98%	98%	**100%**	**100%**	98%	99%	**100%**
18	53	64	26.44	48.8	97%	98%	**100%**	97%	99%	**100%**	97%	99%	**100%**	97%	99%	**100%**
20	27	25	60.94	66.4	98%	99%	99%	98%	99%	**100%**	96%	**100%**	**100%**	96%	99%	**100%**
21	43	56	37.78	54.6	97%	99%	99%	97%	99%	99%	97%	**100%**	**100%**	97%	**100%**	**100%**
22	30	45	45.78	56	95%	96%	96%	95%	96%	96%	95%	98%	**100%**	95%	98%	**100%**
23	1	2	98	87.5	97%	99%	**100%**	97%	99%	99%	97%	**100%**	**100%**	97%	**100%**	**100%**
24	41	55	51.75	63.1	96%	98%	98%	96%	98%	98%	96%	99%	**100%**	96%	99%	**100%**
25	60	52	72.92	75.2	96%	98%	98%	96%	98%	98%	95%	99%	**100%**	95%	99%	**100%**
26	47	57	37.36	54.9	95%	97%	97%	96%	97%	97%	92%	96%	**100%**	92%	96%	**100%**
27	5	66	27.47	47.4	93%	94%	96%	93%	94%	96%	93%	95%	**100%**	93%	95%	99%
28	2	60	34.07	47.5	92%	95%	96%	92%	94%	96%	91%	97%	**100%**	91%	97%	98%
30	7	56	82.5	86.9	91%	93%	93%	91%	93%	93%	90%	96%	**100%**	89%	96%	**100%**

Table IV is about the performance of ICAs for the *modified problems*. In this table, relative percentage deviation (*RPD*) of performance measures derived from the best found solutions and the proposed algorithms' solutions is presented. *RPD* value is calculated by (14):

$$RPD = \left| \frac{Max_{mea} - A\lg_{mea}}{Max_{mea}} \right| \times 100 \quad (14)$$

where Max_{mea} is the performance measure of best found solution and $A\lg_{mea}$ is the proposed algorithms' performance measure.

Table IV shows the RPD of *GTE* of different proposed ICA algorithms in less than 1 second, 5 and 10 seconds for all modified test problems. Algorithms' behaviors are so alike for *GTE* and *β* metrics. These values illustrate that algorithms perform better in longer runs as it is expected. Algorithms' results are almost the same in less than 1 second. Table IV demonstrates that the EICA outperforms the basic ICA clearly. The EICA finds most well-known solutions, and in a few large scale problems there are little deviation to the best solutions.

V. CONCLUSION

This paper presented new versions of imperialist competitive algorithm (ICA) as perfect solver for the CFP using sequence data component-machine incident matrices. Besides the modifications done on the conventional ICA to deal with the considered problem, an enhanced version of ICA (EICA), that make use of local search operators, have been proposed too. The proposed ICAs solved the CFP substantially more efficient than other state of the art algorithms. Operators of the main ICA and embedded local search have been successfully designed to improve intensification and diversification features of proposed algorithms. ICA algorithms were implemented on some benchmark problems in literature and some others modified to sequence data. Results showed that the presented methods work as well as other satisfactory algorithms in some cases, even better than other ones. Moreover, it is obvious that EICA outperforms basic ICA in large size problems; whereas in smaller sizes, they have the same results with less deviation.

ACKNOWLEDGEMENT

The authors express sincere thanks to Zahra Booyavi whose valuable remarks and comments helped to improve the paper.

REFERENCES

[1] E. Aarts, E., Lenstra, J.K., 1997. Local Search in Combinatorial Optimization. *John Wiley & Sons, New York*, ISBN: 0471948225.

[2] E. Atashpaz-Gargari, C. Lucas, "Imperialist competitive algorithm: An algorithm for optimisation inspired by imperialistic competition," In: *IEEE Congress on Evolutionary Computation*, pp. 4661–4667, 2007.

[3] A.H. Banisadr, M. Zandieh, I. Mahdavi, "A hybrid imperialist competitive algorithm for single-machine scheduling problem with linear earliness and quadratic tardiness penalties," *Int. J. Adv. Manuf. Technol.,* vol. 65, no. 5-8, pp.981–989, 2013.

[4] E.C. Brown, R.T. Sumichrast, "CF-GGA: a grouping genetic algorithm for the cell formation problem," *Int. J. Prod. Res.*, vol. 39, no. 16, 3651-3669, 2001.

[5] B.A. Elbenani, J. Ferland, J. Bellemare, "Genetic algorithm and large neighborhood search to solve the cell formation problem," *Expert Syst. Appl.*, vol. 39, no. 3,pp. 2408–2414, 2012.

[6] S. Forouharfard, M. Zandieh, "An imperialist competitive algorithm to schedule of receiving and shipping trucks in cross-docking systems," *Int. J. Adv. Manuf. Technol.,* vol. 51, no. 9-12, pp. 1179–1193, 2010.

[7] J.F. Gonçalves, M.G.C. Resende, "An evolutionary algorithm for manufacturing cell formation," *Comput. Ind. Eng.*, vol. 47, no. 2-3, pp. 247–273, 2004.

[8] M. Gravel, A.L. Nsakanda, W. Price, "Efficient solutions to the cell-formation problem with multiple routings via a double-loop genetic algorithm," *Eur. J. Oper. Res.*, vol. 109, no. 2, pp. 286–298, 1998.

[9] Kaveh, A., Talatahari, S., 2010. Optimum design of skeletal structures using imperialist competitive algorithm. *Computers and Structures*, 88(21-22), 1220–1229, 2010.

[10] V. Kayvanfar, M. Zandieh, "The economic lot scheduling problem with deteriorating items and shortage: an imperialist competitive algorithm," *Int. J. Adv. Manuf. Technol.*, vol. 62, no. 5-8, pp. 759-773, 2012.

[11] K.B. Keeling, E.C. Brown, T.L. James, "Grouping efficiency measures and their impact on factory measures for the machine-part cell formation problem: A simulation study," *Eng. Appl. Artif. Intel.*, vol. 20, no. 1, pp. 63–78,2007.

[12] K. Lian, C. Zhang, L. Gao, X. Shao, "A modified colonial competitive algorithm for the mixed-model U-line balancing and sequencing problem," *Int. J. Prod. Res.*, vol. 50, no. 18, pp. 5117-5131, 2011.

[13] K. Lian, C. Zhang, X. Shao, L. Gao, "Optimization of process planning with various flexibilities using an imperialist competitive algorithm," *Int. J. Adv. Manuf. Technol.*, vol. 59, no. 5-8, pp. 815–828, 2012.

[14] I. Mahdavi, E. Teymourian, N. Tahami Baher, V. Kayvanfar, "An integrated model for solving cell formation and cell layout problem simultaneously considering new situations," *J. Manuf. Sys.,* vol. 32, no. 4, 655-663, 2013

[15] I. Mahdavi, B. Mahadevan, "CLASS: An algorithm for cellular manufacturing system and layout design using sequence data," *Robot. Com-Int. Manuf.*, vol. 24, no. 3, pp. 488–497, 2008.

[16] A.S. Mamaghani, M.R. Meybodi, "An application of Imperialist Competitive Algorithm to solve the quadratic assignment problem," In: *Internet Technology and Secured Transactions (ICITST), International Conference*, pp. 562-565, 2011.

[17] H. Moradi, M. Zandieh, "An imperialist competitive algorithm for a mixed-model assembly line sequencing problem," *J. Manuf. Sys.*, vol. 32, no. 1, pp.46–54, 2013.

[18] G.J. Nair, T.T. Narendran, "CASE: A clustering algorithm for cell formation with sequence data," *Int. J. Prod. Res.*, vol. 36, no. 1, pp. 157–180, 1998.

[19] S. Nazari-Shirkouhi, H. Eivazy, R. Ghodsi, K. Rezaie, E. Atashpaz-Gargari, "Solving the integrated product mix-outsourcing problem using the Imperialist Competitive Algorithm," *Expert Syst. Appl.*, vol. 37, no. 12, pp. 7615–7626, 2010.

[20] H. Nouri, S.H. Tanga, B.T. Hang Tuaha, M.K. Anuara, "BASE: A bacteria foraging algorithm for cell formation with sequence data," *J. Manuf. Sys.*, vol. 29, no. 2-3, pp. 102–110, 2010.

[21] G. Papaioannou, J.M. Wilson, "The evolution of cell formation problem methodologies based on recent studies (1997–2008): review and directions for future research," *Eur. J. Oper. Res.*, vol. 206, no. 3, pp. 509–521, 2010.

[22] D.T. Pham, A.A. Afify, E. Koç, 2007. Manufacturing cell formation using the Bees Algorithm. In: *innovation production machines and systems virtual conference*. Cardiff, UK.

[23] E. Shokrollahpour, M. Zandieh, B. Dorri, "A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem," *Int. J. Prod. Res.*, vol. 49, no. 11, pp. 3087–3103, 2010.

[24] K. Spiliopoulos, S. Sofianopoulou, "An efficient ant colony optimization system for the manufacturing cells formation problem," *Int. J. Adv. Manuf. Technol.*, vol. 36, no. 5-6, pp.589–597, 2008.

[25] A. Talebi, M.A. Molaei, B. Ashrafi, "Application of an Imperialist Competitive Algorithm in Portfolio Optimization," *World Appl. Sci. J.*, vol. 14, no. 10, pp. 1576-1598, 2011.

[26] T.-H. Wu, S.-H., Chung, C.-C., Chang, "A water flow-like algorithm for manufacturing cell formation problems," *Expert Syst. Appl.*, vol. 205, no. 2, pp. 346–360, 2010.

[27] T.-H. Wu, C. Low, W.-T. Wu, "A tabu search approach to the cell formation problem," *Int. J. Adv. Manuf. Technol.*, vol. 23, no. 1, pp. 916–924, 2004.

[28] T-H. Wu, C-C. Chang, S-H. Chung, "A simulated annealing algorithm for manufacturing cell formation problems," *Eur. J. Oper. Res.*, vol. 34, no. 3, pp. 1609–1617, 2008.

[29] M. Yousefi-khoshbakht, M. Sedighpour, "A New Imperialist Competitive Algorithm to Solve the Traveling Salesman Problem," *Int. J. Comput. Math.,* vol. 90, no. 7, pp. 1495-1505. 2013

[30] J.R. King, and V. Nakornchai, "Machine-component group formation in group technology: Review and extension," *Int. J. Prod. Res.*, vol. 20, no. 2, pp. 117-133, 1982.

[31] A. Kusiak, W. and Chow, "Efficient solving of the group technology problem," *J. Manuf. Sys.,* vol. 6, no. 2, pp. 117-124, 1987.

[32] H. Seifoddini, "Single linkage versus average linkage clustering in machine cells formation applications", *Comput. Ind. Eng.,* vol. 16, no. 3, pp. 419-426, 1989.

[33] C.T. Mosier, L. Taube, "The facets of group technology and their impact on implementation," *OMEGA*, vol. 13, no. 6, pp. 381-391, 1985.

[34] H.M. Chan, D. A. Milner, "Direct clustering algorithm for group formation in cellular manufacture," *J. Manuf. Sys.*, vol. 1, pp. 65-75, 1982.

[35] M. P. Chandrashekharan, R. Rajagopalan, "An ideal seed non-hierarchical clustering algorithm for cellular manufacturing," *Int. J. Prod. Res*., vol. 24, no. 2, pp. 451-464, 1986.

[36] M. P. Chandrashekharan, R. Rajagopalan, "MODROC: An extension of rank order clustering for group technology," *Int. J. Prod. Res.*, vol. 24, no. 5, pp. 1221-1233, 1986.

[37] R. G. Askin, S. Subramanian, "A cost-based heuristic for group technology configuration," *Int. J. Prod. Res.*, vol. 25, no. l, pp. 101-113, 1987.

[38] C.T. Mosier, L. Taube, "Weighted similarity measure heuristics for the group technology machine clustering problem," *OMEGA*, vol. 13, no. 6, pp. 577-583, 1985.

[39] S. Carrie, "Numerical Taxonomy applied to Group Technology and Plant Layout," *Int. J. Prod. Res.*, vol. 11, pp. 399-416, 1973.

[40] K.R. Kumar, A. Kusiak, A. Vannelli, "Grouping of parts and components in flexible manufacturing systems," *Eur. J. Oper. Res.*, vol. 24, pp. 387-397, 1986.

[41] J.R. King, "Machine-component grouping in production flow analysis: An approach using a rank order clustering algorithm," *Int. J. Prod. Res.*, vol. 18, no. 2, pp. 213-232, 1980.

[42] W. Boe, C.H. Cheng, "A close neighbor algorithm for designing cellular manufacturing systems," *Int. J. Prod. Res.*, vol. 29, no. 10, pp. 2097-2116, 1991.

[43] M.P. Chandrasekharan, R. Rajagopalan, "GROUPABILITY: Analysis of the properties of binary data matrices for group technology," *Int. J. Prod. Res*., vol. 27, no. 6, pp. 1035-1052, 1989.

[44] M.P. Chandrasekharan, R. Rajagopalan, "ZODIAC - An algorithm for concurrent formation of part families and machine cells," *Int. J. Prod. Res*., vol. 25, no. 6, pp. 835-850, 1987.

[45] P. H. Waghodekar, S. Sahu, "Machine-component cell formation in group technology: MACE," *Int. J. Prod. Res.*, vol. 22, no. 6, pp. 937-948, 1984.

[46] A. Kusiak, Group technology: Models and solution approaches. In First Industrial Engineering Research Conference (pp. 349-352), 1992.

[47] F. F. Boctor, "A linear formulation of the machine-part cell formation problem," *Int. J. Prod. Res.*, vol. 29, no. 2, pp. 343-356, 1991.

[48] R.S. Pandian, S. S. Mahapatra, "Manufacturing cell formation with production data using neural networks," *Comput. Ind. Eng*., vol. 56, no. 4, pp. 1340-1347, 2009.

[49] L.E. Stanfel, "Machine clustering for economic production," *Eng. Costs Prod. Econ.*, vol. 9, no. 1, pp. 73-81, 1985.

[50] W. T. McCormick, P. J. Schweitzer, T. W. White, "Problem decomposition and data reorganization by a clustering technique," *Oper. Res.*, vol. 20, no. 5, pp. 993–1009, 1972.

[51] H. Seifoddini, P.M. Wolfe, "Application of the similarity coefficient method in group technology," *IIE Trans*., vol. 18, no. 3, pp. 271–277, 1986.