Int J Adv Manuf Technol (2016) 82:25–37
DOI 10.1007/s00170-015-7052-z

ORIGINAL ARTICLE

A Metaheuristic algorithm for the manufacturing cell formation problem based on grouping efficacy

Azadeh Noktehdan[1] · **Seyedmohammad Seyedhosseini**[1] · **Mohammad Saidi-Mehrabad**[1]

Received: 12 February 2014 / Accepted: 17 March 2015 / Published online: 3 June 2015

Abstract The cell formation problem determines decomposition of the manufacturing cells of a production system. Machines are assigned to the cells to process one or more part families so that each cell is operated independently and the inter-cellular movements are minimized. This paper proposes a new algorithm for grouping problems (bin packing, graph coloring, scheduling, etc.) which is a grouping version of an almost new algorithm (league championship algorithm (LCA)), and we used it to solve benchmarked instances of cell formation problem posing as a grouping problem. To evaluate the effectiveness of our approach, we borrow a set of 35 most widely used benchmark problem instances from literature and compare the performance of grouping LCA (GLCA) and several well-known algorithms published. The proposed algorithm can reach the best solution for 29 of the 35 benchmark problems and differs with the best-known solution of three benchmark problems only with 0.7 % average gap. We also used a new method to find the number of initial cells. The results show that GLCA may hopefully be a new approach for such kinds of difficult-to-solve problems. Moreover, a real-world industrial case is provided to show how the proposed algorithm works. Considering the performance of the GLCA algorithm on all test problems, the proposed algorithm should thus be useful to both practitioners and researchers.

✉ Seyedmohammad Seyedhosseini
seyedhosseini@iust.ac.ir

Azadeh Noktehdan
a_noktehdan@yahoo.com; a.noktehdan@utoronto.ca

Mohammad Saidi-Mehrabad
mehrabad@iust.ac.ir

1 Department of Industrial Engineering, Iran University of Science & Technology, Narmak, Tehran, Iran

Keywords Cell formation problem · Grouping genetic algorithm · League championship algorithm

1 Introduction

Group technology (GT) is a theory of management based on the principle that similar things should be done similarly. Products needing similar operations and a common set of resources are grouped into families, the resources being regrouped into production subsystems [18].

One application of GT is in cellular manufacturing (CM) which aims to decompose a manufacturing system into manufacturing cells where dissimilar machines are aggregated into cells and operate on a part family (the most similar parts in terms of design characteristics are aggregated into a part family). The main objective is thus minimizing inter-cell and intra-cell transportation costs, grouping efficiency, grouping efficacy, exceptional elements, etc. Among the most significant benefits of cellular manufacturing are reduced setup times, reduced work-in-process inventories, reduced lead times, reduced material handling costs, etc.

Most of the machine-part cell formation (MPCF) problems are NP-hard problems; therefore, optimization algorithms yield a globally optimal solution in a prohibitive computation time. The computational intractability of the problem makes the use of heuristic approaches appropriate. Given that MPCF is a grouping problem (in a grouping problem, one seeks for grouping of objects into disjoint groups with the aim of optimizing a given objective function of groups and a set of hard constraints), successful applications of grouping genetic algorithm (GGA) on MPCF have been reported by researchers [18, 11, 55, 10]. Up to now, there are a few algorithms that have been mainly modified to consider the structure of grouping problems, such as cell formation problem, like genetic algorithm, differential evolution algorithm, and so on. The resulting algorithms are called GGA and grouping differential evolution algorithm (GDE).

In this paper, we apply the notion of grouping representation [20] and operators in the body of league championship algorithm to obtain the grouping league championship algorithm (GLCA). The purpose is to develop a procedure that is efficient and effective for obtaining machine-part groupings when the manufacturing system is represented by a 0-1 machine-part incidence matrix. Considering the good performance of league championship algorithm (LCA) in numerical problems, we also selected it in this research as the basis for developing search methods for the CF problem. The LCA algorithm is initialized with a population of solutions. A number of individuals as sport teams compete in an artificial league for several weeks (iterations). Based on the league schedule in each week, teams play in pairs and the outcome is determined in terms of win or loss, given known the team's playing strength (fitness value) resultant from a particular team formation (solution). In the recovery period, each team devises the required changes in the formation/playing style (a new solution) for the next week contest and the championship goes on for a number of seasons (stopping condition) [25]. Performance of the proposed algorithm is tested in comparison with that of GGA [10], hybrid grouping genetic algorithm (HGGA) [10], GDE [40], hybrid grouping differential evolution algorithm (HGDE) [40], genetic algorithm and large neighborhood search (local search algorithm (LSA) and hybrid method (HM)) [19], and hybrid heuristic algorithm adopting both Boltzmann function and mutation operator (similarity coefficient method-Bayes multiple classification function, SCM-BMCF) [59].

The remainder of the paper is organized as follows: In Section 2, we briefly describe the problem definition. Section 3 provides a review of previous solution approaches developed for cell formation problem. An overview on league championship algorithm is given in Section 4. The proposed grouping league championship algorithm is introduced in Section 5. Section 6 deals with the computational results. The case study and its related results are presented in Section 7, and Section 8 concludes the paper.

2 Machine-part cell formation problem

Given a 0-1 machine-part incidence matrix, the machine-part cell formation problem can be formulated as a block diagonalization problem. In other words, we are seeking a proper rearrangement of rows and columns of the machine-part incidence matrix to create a block diagonal matrix indicating part families and machine cells (see Boctor [6] for the linear formulation of the machine-part cell formation problem). The objective is to determine a rearrangement so that the inter-cellular movement is minimized and machine utilization within a cell is maximized. Table 1 shows a machine-part incidence matrix relevant to a problem with five parts and five machines. In this matrix, no blocks can be observed directly. However, after diagonalization process, the block diagonal matrix is obtained as illustrated in Fig. 1b [58].

Table 1 Machine-part incidence matrix

P(i)/M(j)	1	2	3	4	5	6	7	8	9	10	11
1	1	0	1	0	0	0	1	0	0	0	1
2	1	1	0	0	0	1	0	0	0	0	0
3	0	1	0	0	0	1	0	0	1	0	0
4	0	0	0	1	1	0	0	0	0	1	0
5	0	0	1	0	0	0	1	0	0	0	0
6	0	0	1	1	0	0	0	0	0	0	1
7	0	0	0	0	1	0	0	1	0	1	0

There have been developed several measures for quantifying goodness of machine-part groups in MPCF [46]. Two measures frequently used are the grouping efficiency [14, 15] and the grouping efficacy [31]. In their paper, Goncalves and Resende [23] have justified several reasons for using grouping efficacy instead of grouping efficiency. Following Goncalves and Resende [23], we use grouping efficacy measure defined as follows:

$$\tau = \frac{e - e_0}{e + e_v} \tag{1}$$

where e is the total number of 1 s, e_0 is the total number of exceptional elements, and e_v is the total number of voids. Those 1 s outside the diagonal blocks are called exceptional elements, while those 0 s inside the diagonal blocks are called voids. The closer the grouping efficacy is to 1, the better will be the grouping. As grouping efficacy has been widely accepted in recent studies on CF problem, we will use it to measure the quality of a given solution.

3 Literature review

Many methods of cell formation have been developed and published in the last decades. A simulated annealing approach is presented by Rao [44]. His measure to evaluate the effectiveness of his approach was grouping efficacy measure. A genetic algorithm approach is introduced by Roy and Komma [45]. They used fitness function measure to evaluate the effectiveness of their algorithm. Husseinzadeh Kashan et al. [27] presented a particle swarm optimization algorithm for machine-part cell formation problem. Sayadi et al. [47] developed a discrete firefly algorithm to minimize the total number of exceptional elements in CF problem. Boutsinas [9] presented a bi-clustering algorithm for cell formation problem. He used grouping efficacy grouping efficiency and bound energy measures to evaluate the effectiveness of his approach. Recently, Batsyn et al. [5]

Fig. 1 **a** Initial incidence matrix. **b** Incidence matrix after diagonalization process

a)

Machines \ Parts	p_1	p_2	p_3	p_4	p_5
m_1	1	0	0	1	0
m_2	0	1	1	0	1
m_3	1	0	0	1	0
m_4	0	1	1	0	1
m_5	1	0	0	1	0

b)

Machines \ Parts	p_2	p_3	p_5	p_1	p_4
m_2	1	1	1	0	0
m_4	1	1	1	0	0
m_1	0	0	0	1	1
m_3	0	0	0	1	1
m_5	0	0	0	1	1

have introduced a new pattern-based approach within the linear assignment model with the purpose to design a heuristic for MPCP. The suggested heuristic combines two procedures, namely the pattern-based procedure to build an initial solution and an improvement procedure to obtain a final solution with high grouping efficacy. The author's investigation with the most popular set of 35 benchmark instances showed that the proposed heuristic returns either the best-known or improved solutions to the MPCF.

The most recent examples of exact algorithms are presented in Krushinsky and Goldengorin [30] for solving the MPCF by means of MINpCUT model and Goldengorin et al. [22] by means of the p median model. Elbenani et al. [19] presented a local search procedure for the manufacturing cell formation problem which applies sequentially an intensification strategy to improve locally a current solution and a diversification strategy destroying more extensively a current solution to recover a new one. They also used the local search procedure with a steady-state genetic algorithm. A recent elaborated survey by Ghosh et al. [21] describes many different metaheuristics developed in the literature for MPCF problems. Pailla et al. [42] have also proposed two different algorithms based on an evolutionary local search algorithm and a simulated annealing approach for the cell formation problem and conducted a numerical comparison on 36 test problems. Anvari et al. [1] proposed a hybrid particle swarm optimization technique for MPCF. The initial solutions generated either randomly or using a diversification generation method and the technique also utilized mutation operator embedded in velocity update equation to avoid reaching local optimal solutions. Wu et al. [60] adopted the WFA logic, which mimics the natural behavior of water flowing from higher to lower levels, and designed a heuristic algorithm for solving the cell formation problem. Solimanpur et al. [51] solve the machine-part cell formation problem using an ant colony optimization method. Their results indicate that application of the proposed algorithm has resulted in 5.73 % improvement in the total number of inter-cellular movements and voids on average. Wu et al. [58] proposed a simple simulated annealing algorithm for the MPCF problem with the objective of maximizing grouping efficacy. Their comparative study revealed that their algorithm is able to improve grouping efficacy for many test problems, in comparison with other algorithms. An algorithm based on evolution strategy has been developed by Stawowy [53] which uses a modified permutation with a separator encoding scheme and a unique concept of separators' movement during the mutation process. Results showed that the proposed evolution strategy is able to outperform previously known heuristics. Selim et al. [50] have done an extensive review of prior research. Comprehensive summaries and taxonomies of studies devoted to part-machine grouping problems are presented also in Wemmerlov and Hyer [57]. For a concise review of procedures based on the type of general solution methodology used (e.g., cluster analysis, graph partitioning, mathematical programming, and other) see Goncalves and Resende [23]. Sun et al. [54] presented a short-term tabu search-based algorithm for solving the CF problem with the objective of minimizing inter-cellular part flows. With the objective of maximizing parts flow within cells, Lei and Wu [35] proposed a long-term tabu search-based algorithm. Goncalves and Resende [23] combined a genetic algorithm with a local search algorithm to develop a hybrid approach. Their approach is able to report good quality solutions. Cheng et al. [17] formulated the CF problem as a traveling salesman problem (TSP) and proposed a solution methodology based on genetic algorithm. Boulif and Atif [8] proposed an enhanced genetic algorithm with a branch-and-bound algorithm for the manufacturing cell formation problem. Lei and Wu [36] worked with multi-objective cell formation problem and proposed a Pareto-optimality based on multi-objective tabu search for minimization of the weighted sum of inter-cell and intra-cell moves and minimization of the total cell load variation. A hybrid methodology based on using Boltzmann function of simulated annealing and mutation operator of GA was proposed by Wu et al. [59] to optimize the initial clustering obtained from similarity coefficient method (SCM) and rank order clustering (ROC). Ateme-Nguem and Dao [3] developed an ant algorithm-based TS heuristic for cellular system design problem, where their approach proved to be much quicker than traditional methods when considering operational sequence, time, and cost. The authors further proposed quantized Hopfield network for CF problem to find optimal or near-optimal solution, and TS was employed to improve the performance and quality of the solution of network [4]. For the problem of minimizing total cell load variation and the total inter-cellular moves, Prabhaharan et al. [43] proposed an ant colony system. The parameters that they considered include demands for number of parts, routing sequences, processing time, machine capacities, and machine workload status. They have compared their results with those obtained by a genetic algorithm.

Recently, the grouping genetic algorithm, a heavily modified genetic algorithm to suit the structure of grouping problems, like CF, has attracted many researchers working on various grouping problems. Brown and James [10] presented a hybrid grouping genetic algorithm for the cell formation problem that combines a local search with a standard grouping genetic algorithm. Their computational experiments signify that the hybrid approach outperforms the standard grouping genetic algorithm in terms of grouping efficacy. DeLit et al. [18] proposed a grouping genetic algorithm for the cell formation problem to minimize traffic of items between the cells. Vin et al. [55] addressed the cell formation problem with alternative part routing, considering capacity constraints. For minimizing inter-cellular traffic, the authors proposed an integrated approach based on a multiple-objective grouping genetic algorithm for the preferential routing selection of each part and an integrated heuristic for the cell formation problem.

4 An introduction to LCA

Population based stochastic local search techniques are a relatively new paradigm in the field of optimization. There are several nature-inspired techniques belonging to this family that use metaphors as guides in solving problems. The most famous members are genetic algorithms that use the metaphor of genetic and evolutionary principles of fitness selection for reproduction to search solution spaces. Inspired from the natural and social phenomena, metaheuristic algorithms have attracted many researchers from various fields of science in recent years. Beside all of the applications such as business, industry, and engineering, a new metaheuristic algorithm is introduced that uses a novel metaphor as guide for solving optimization problems.

League championship algorithm mimics the sport league championships. A number of individuals making role as teams compete in an artificial league for several weeks (iterations). Based on the league schedule in each week, teams play in pairs and the outcome is determined in terms of win or loss, given known the team's playing strength (fitness value) resultant from a particular team formation (solution). Keeping track of the previous week experiences, each team devises the required changes in the formation/playing style (a new solution) for the next week contest and the championship goes on for a number of seasons (stopping condition). The details of the pseudo code of the LCA is in [25].

Each team such as i plays based on its current best formation B_i^t, while devising some changes resulted from post $x_i^{t+1}=(x_{i1}^{t+1},x_{i2}^{t+1},\ldots,x_{in}^{t+1})$ for team $i(i=1, \ldots, L)$ by following the equations.

If i was the winner and l was the winner, then

$$x_{id}^{t+1} = b_{id}^t + y_{id}^t\left(C_1R_1\left(x_{id}^t - X_{kd}^t\right) + C_1R_2\left(x_{id}^t - X_{jd}^t\right)\right) \quad d = 1,\ldots,n \quad (2)$$

Else if i was the winner and l was the loser, then

$$x_{id}^{t+1} = b_{id}^t + y_{id}^t\left(C_2R_1\left(x_{kd}^t - X_{id}^t\right) + C_1R_2\left(x_{id}^t - X_{jd}^t\right)\right) \quad d = 1,\ldots,n \quad (3)$$

Else if i was the loser and l was the winner, then

$$x_{id}^{t+1} = b_{id}^t + y_{id}^t\left(C_1R_2\left(x_{id}^t - X_{kd}^t\right) + C_2R_1\left(x_{jd}^t - X_{id}^t\right)\right) \quad d = 1,\ldots,n \quad (4)$$

Else if i was the loser and l was the loser, then

$$x_{id}^{t+1} = b_{id}^t + y_{id}^t\left(C_2R_2\left(x_{kd}^t - X_{id}^t\right) + C_2R_1\left(x_{jd}^t - X_{id}^t\right)\right) \quad d = 1,\ldots,n \quad (5)$$

End if

In the above, d is the dimension index. R_1 *and* R_2 are uniform random numbers in [0,1]. C_1 and C_2 are constant coefficients used to scale the contribution of the strength and weakness components, respectively.

For each team i, we have

l=index of the team that will play with team i based on the league schedule at week t+1.

j=index of the team that has played with team i based on the league schedule at week t.

k=index of the team that has played with team l based on the league schedule at week t.

y_{id}^t is a binary change variable which indicates whether the dth element in the current best formation will change or not [25].

5 The grouping league championship algorithm

Several approaches have been used to deal with discrete variable optimization. Most of them round off the variable to the nearest available value before evaluating each trial vector. One of the key issues when designing the LCA algorithm lies in its solution representation where teams bear the necessary information related to the problem domain on hand.

5.1 GLCA encoding

This section describes how the LCA metaheuristic is modified to solve the cell formation problem which is discrete in nature. When designing the LCA algorithm, one of the key issues lies in solution representation where teams bear the necessary information related to the problem domain on hand. Given the fact that the MPCF problem is essentially a grouping problem, we use the encoding strategy proposed by Falkenauer [20], namely the grouping encoding.

In general, the grouping encoding consists of two parts: an object part and a group part. The object part consists of a $1\times n$ array where n is the number of objects to be grouped. The group part consists of a permutation of K group labels. Each member in the object part can take any of the K group labels as allele, indicating that the object in question belongs to a group of the given label. Consider, for example, the individual ABCB that encodes the solution where the first object is in group A, second in B, third in C, and fourth in B. The grouping representation related to this individual could be ABCB|CAB. Note that the order in which the groups are listed (the order CAB) does not matter.

A direct adaptation of the above encoding scheme for the MPCF problem can be the three-part representation shown in (6), where p_i denotes what group part i is assigned to (i=1,…, P), m_j denotes what group machine j is assigned to (j=1,…,M), and c_k denotes the cell number for cell (k=1,…,K) (note that P and M are the number of parts and machines, respectively).

$$p_1p_2\ldots p_p|m_1m_2\ldots m_M|c_1c_2\ldots c_k. \tag{6}$$

We may address a solution encoded in the form of (6) as a team. The cell segment (the third part in (6)) of the team can vary in length depending on the number of cells into which the machines and parts are grouped. Considering the example given in Fig. 1b, the relevant team can be encoded as 21121|21212|12.

5.2 Initial solution

To start the search for an optimal solution, it is required to initialize a system of teams. This means, a population of initial solutions feasible to the MPCF problem should be generated. An intuitive solution approach to the cell formation problem is to decompose the entire problem into two sub-problems dealing with part assignment and machine assignment, respectively [58]. After part assignment, a proper assignment of machines can be done and the generation of a solution is completed. The part and machine assignment steps are detailed as follows.

5.2.1 Parts assignment

As an index for assigning parts to cells, the Minkowski's metrics [24] is used to evaluate dissimilarity between parts. The dissimilarity measure d_{ij}, is defined as follows:

$$d_{ij}=\sum_{k=1}^{M}|a_{ik}-a_{jk}| \tag{7}$$

where

$$a_{ik}=\begin{cases}1 & \text{If part } i \text{ requires processing on machine } k\\ 0 & \text{Otherwise}\end{cases}$$

The dissimilarity index is calculated for each pair of parts and then parts are assigned in a greedy fashion based on their dissimilarity index. The lower dissimilarity measure for a pair of parts indicates the higher priority for grouping them in the same cell.

5.2.2 Machine assignment

The same strategy can be used for assigning machines to groups. Here, the dissimilarity measure is defined as

$$d_{ij}=\sum_{k=1}^{P}|b_{ki}-b_{kj}| \tag{8}$$

where

$$b_{ki}=\begin{cases}1 & \text{If machine } i \text{ has an opeartion on part } k\\ 0 & \text{Otherwise}\end{cases}$$

In this measure, p is the total number of parts. After calculating the dissimilarity matrix for each pair of machines, we generate the initial machine assignment.

The potential of any initial solution of MPCF depends on the number of its cells (L). Starting with a large number of cells would be undesirable while starting always with a small number of cells may results in a less diverse initial population. Thus, we use a truncated geometric distribution to generate randomly the number of cells [26]. Using a geometric distribution to simulate the number of cells ensures that the probability of starting with large number of cells would be small, against the high probability for starting with a lower number of cells. The following relation gives the number of cells used to generate an initial solution.

$$L=\left\lceil\frac{\ln\left(1-\left(1-(1-p)^{\iota}\right)R\right)}{\ln(1-p)}\right\rceil+1\ : L\in\{1,2,3,\ldots,\iota\} \tag{9}$$

where r is a uniformly distributed variable belonging to [0,1], p is the probability of success in the geometric distribution, and l is the maximum number of cells (l=min{P, M}). In our implementation we use p=0.5.

The probability value p controls the initial number of cells in an initial team. A relatively large value for p would probably results in opening of a small number of cells in comparison with l and assigning all products and machines to them, vice versa. We found suitable based on a series of initial computational efforts. This value provides a balance between the number of teams with small or large number of cells relative to the maximum number of cells, l. To generate an initial team, first the number of cells (L) is determined by (9) and parts and machines are assigned accordingly.

5.3 Updating equations

The process of generating a new team (a new solution) is carried out by the following equations:

$\mathrm{A}(P_g^t)$ = 2 3 1 1 1 2 3 2 | 1 2 3 3 | 1 2 3

$\mathrm{B}(X_k^t)$ = 3 2 1 2 4 1 3 4 | 1 2 3 4 | 1 2 3 4

C= A⊖B = 2 3 0 1 1 2 0 2 | 0 0 0 3 | 1 2 3 ⇨ Parts 3 and 7, and machines 1, 2 and 3 are missing and should be reassigned by a repair heuristic

Fig. 2 Illustration of the manner in which ⊖ operator performs

If *i* was the winner and *l* was the winner, then

$$x_i^{t+1} = b_i^t \oplus \left(R_1 \otimes \left(x_i^t \ominus \mathcal{X}_k^t\right) \oplus R_2 \otimes \left(x_i^t \ominus \mathcal{X}_j^t\right) \qquad d = 1, \ldots, n \qquad (10)$$

Else if *i* was the winner and *l* was the loser, then

$$x_i^{t+1} = b_i^t \oplus \left(R_1 \otimes \left(x_k^t \ominus \mathcal{X}_i^t\right) \oplus R_2 \otimes \left(x_i^t \ominus \mathcal{X}_j^t\right) \qquad d = 1, \ldots, n \qquad (11)$$

Else if *i* was the loser and *l* was the winner, then

$$x_i^{t+1} = b_i^t \oplus \left(R_2 \otimes \left(x_i^t \ominus \mathcal{X}_k^t\right) \oplus R_1 \otimes \left(x_j^t \ominus \mathcal{X}_i^t\right) \qquad d = 1, \ldots, n \qquad (12)$$

Else if *i* was the loser and *l* was the loser, then

$$x_i^{t+1} = b_i^t \oplus \left(R_2 \otimes \left(x_k^t \ominus \mathcal{X}_i^t\right) \oplus R_1 \otimes \left(x_j^t \ominus \mathcal{X}_i^t\right) \qquad d = 1, \ldots, n \qquad (13)$$

In the above equations, R_1 and R_2 are 1-by-n arrays comprising 0 or 1 elements. These random arrays are generated from a Bernoulli distribution in which the probability of getting 1 is equal to 0.3. B_i^t is the current best formation for team *i* experienced till week *t*. X_i^t is the formation of team *i* at week *t* while devising some changes resulted from post x_i^{t+1} for team $i(i=1, \ldots, L)$ on of the above equations. X_i^t is the formation of team *j* at week *t*, X_k^t is the formation of team *k* at week *t*, and X_l^t is the formation of team *l* at week *t*.

The subtract operator (⊖) Differences between the current formation of team *p*, X_p^t; and the current formation of team *q*, X_q^t; $(x_p^t \ominus X_q^t)$ can be presented by an array of elements in which each element shows that whether the content of the corresponding element in X_p^t is different from the desired one (the one which is the winner) or not. If yes, that element gets its value from the one which is the winner. Figure 2 illustrates the manner in which ⊖ operator performs. More precisely, the number of elements that do not the same value in both X_p^t and X_q^t are equal to the Hamming distance between X_p^t and X_q^t. Worth to mention that when X_p^t is exactly equal to X_q^t, we omit the terms $R_1 \otimes (X_p^t \ominus X_q^t)$ or $R_2 \otimes (X_p^t \ominus X_q^t)$, since they are null arrays. Figure 2 illustrates the manner in which ⊖ operator performs.

To reassign orphaned parts or machines (0's in the body of the solution) resulted from subtract operator, a repair heuristic, given by Brown and Sumichrast [11], strategically reassigns orphaned parts or machines to groups for which they are best suited.

It is important to note that since we are developing a new algorithm which is capable to be applied on any grouping problem and that all operators employed in an algorithm modified to suit the structure of grouping problems (i.e., a grouping algorithm) should work with groups (cells, in our case) rather than objects, thus, it is required that ⊖ operator be essentially a grouping operator (examples of grouping operators are crossover and mutation operators in the grouping genetic algorithm). If we consider a group (cell) as a set of different parts and different machines, then the result of ⊖ operator would be equal to the result of applying subtraction operator on two sets, sequentially. Figure 3 illustrates this justification with an example. Given the fact that the set subtraction operator can be regarded as an operator which is applied on two sets (groups), thus, ⊖ would be a grouping operator.

The add operator (⊕) This operator is a crossover operator that typically is used in genetic algorithms. Crossover is viewed as an operator that has the mission of interchanging structural information developed during the search. The crossover operator utilized here is a grouping crossover [20] similar to the one used in Brown and James [10]. The following example describes the crossover mechanism. Consider the following two parental solutions for a 7-part and 11-machine problem.

$$\text{Parent } 1 = 3\,1\,1\,1\,2\,3\,2\,|\,12332312311\,|\,123 \qquad (14)$$

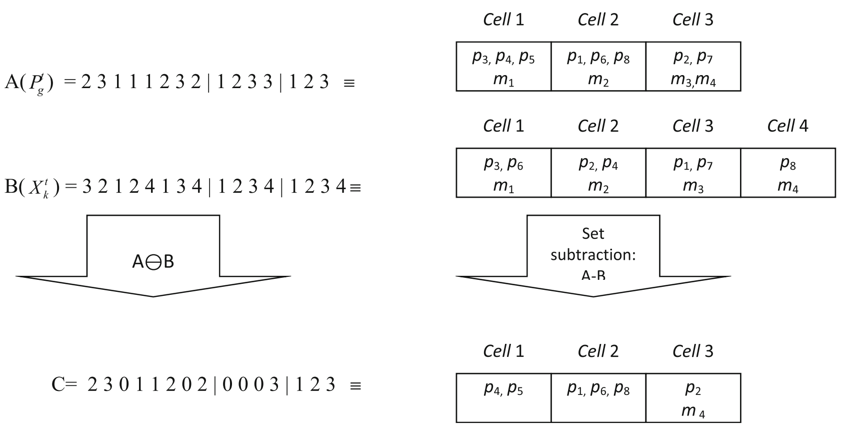

Fig. 3 ⊖ operator is a grouping operator

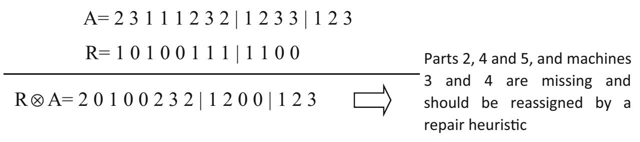
A= 2 3 1 1 1 2 3 2 | 1 2 3 3 | 1 2 3
R= 1 0 1 0 0 1 1 1 | 1 1 0 0
R⊗A= 2 0 1 0 0 2 3 2 | 1 2 0 0 | 1 2 3
Parts 2, 4 and 5, and machines 3 and 4 are missing and should be reassigned by a repair heuristic

Fig. 4 Illustration of the manner in which ⊗ operator performs

$$\text{Parent}\,2 = 2423124\,|\,11221234232|1234. \tag{15}$$

First two crossing sites are chosen from the cell segment of parent 2 as follows:

$$\text{Parent}\,2 = 2423124\,|\,11221234232|[123]4. \tag{16}$$

The selected groups are then inserted into the cell segment of the first parent. This leads to creation of the child 1 after the corresponding machine/part modifications are made to reflect the machine/part assignments of the inserted groups from the contributing parent (the modified values are underlined).

$$\text{Child1} = \underline{2}\ 1\ \underline{2}\ \ \underline{3}\ \underline{1}\ \underline{2}\ 2\,|\,\underline{1}\ \underline{1}\ \underline{2}\ \underline{2}\ \underline{1}\ \underline{2}\ \underline{3}\ 2\ \underline{2}\ \underline{3}\ \underline{2}\,|\,1\ 2\ 3\ \ \underline{1}\ \underline{2}\ \underline{3}. \tag{17}$$

Since the original cell 3 from parent 1 no longer contains any parts or machines, it should be eliminated. The modified child solution is thus as follows:

(18)

At this step, a repair strategy should be devised to eliminate infeasibilities (the original group 1 from parent 1 contains part 2, but no machines). For this purpose, once again we call the repair heuristic [11].

The multiply operator (⊗) By this operator, we can add exploration ability to the GLCA system. This ability is added by selecting a random number of parts/machines and removing them from existing cells. To select the items being eliminated, we generate a random binary array (R_1 or R_2) of size 1-by-$P+M$. For each element in the array, if it is zero, the corresponding part/machine is removed; otherwise, it is retained. The missing parts/machines are reassigned by the repair heuristic. Figure 4 illustrates the manner in which multiply operator performs.

5.4 Repair heuristic

To fix infeasibilities arise when applying GLCA operators, and reassigns orphaned parts or machines, a repair heuristic given by Brown and Sumichrast [11] is adopted. Recalling the previous example:

(19)

There is only one part (part 2) in group 1. Therefore, part 2 must be reassigned to a new group. Considering the *M-P* incidence matrix given in Table 1, we locate all the machines part 2 needs, machines={1, 2, 6}. The machine section of C_1 is then examined to locate which groups machines 1, 2, and 6 are assigned. In this example, machine 6 is in groups 2, while machines 1 and 2 are in group 1. As group 1 contains more of the required machines for part 2, part 2 is reassigned to group 1, and group 1 is eliminated, resulting in the following solution:

(20)

By renumbering the groups, the resulting new child appears as

$$C_1 = 3234231\,|22332341343|1234. \tag{21}$$

This repair operator is applied to all orphaned machines or parts.

We evaluated the partial efficacy measure for each part in machine-part incidence matrix in Table 1. You can see the results in Table 2.

6 Computational results

To exhibit potentials of our approach [41] more, we have compared our results with the results of several algorithms developed in the literature for the MPCF problem. Results are compared with grouping genetic algorithm (GGA) [10], hybrid grouping genetic algorithm (HGGA) [10], genetic al-

Table 2 Partial efficacy matrix

		Group 1	Group 2	Group 3	Group 4
Parts	Required Machines	8	1,2,5	3,4,6,9,11	7,10
1	1,3,7,11	77.27	78.26	79.17	81.82
2	1,2,6	81.82	90.91	76.00	78.26
3	2,6,9	81.82	82.61	83.33	78.26
4	4,5,10	81.82	82.61	76.00	86.36
5	3,7	86.36	79.17	80.00	90.91
6	3,4,11	81.82	75.00	91.30	78.26
7	5,8,10	90.48	82.61	69.23	86.36

Table 3 Computational results on 35 benchmarked problems

	Problem source	Size	Other approaches							Best among 7 algorithms	GLCA				% gap (max sol and best solution)
			GGA Max sol	HGGA Max sol	SCM-BMCF Max sol	GDE Max sol	HGDE Max sol	LSA Max sol	HM Max sol		Min sol	Avg sol	Max sol	Avg time	
1	King and Nakornchai [29]	5×7	82.35	82.35	–	82.35	82.35	82.35	82.35	82.35	82.35	82.35	82.35	24.02	0
2	Waghodekar and Sahu [56]	5×7	69.57	69.57	69.57	69.57	69.57	68	69.57	69.57	69.57	69.57	69.57	24.37	0
3	Seifoddini [48]	5×18	79.59	79.59	79.59	79.59	79.59	79.59	79.59	79.59	79.59	79.84	80.85[a]	35.95	−0.016
4	Kusiak and Chow [33]	6×8	76.92	76.92	76.92	76.92	76.92	76.92	76.92	76.92	76.92	76.92	76.92	25.72	0
5	Kusiak and Chow [34]	7×11	60.87	60.87	60.87	60.87	60.87	60.87	60.87	60.87	60.87	60.87	60.87	31.48	0
6	Boctor [6]	7×11	70.83	70.83	70.83	70.83	70.83	70.37	70.83	70.83	70.83	70.83	70.83	30.73	0
7	Seifoddini and Wolfe [49]	8×12	69.44	69.44	68.29	69.44	69.44	68.29	69.44	69.44	69.44	69.44	69.44	32.33	0
8	Chandrasekharan and Rajagopalan [14]	8×20	85.25	85.25	85.25	85.25	85.25	85.25	85.25	85.25	85.25	85.25	85.25	41.76	0
9	Chandrasekharan and Rajagopalan [15]	8×20	55.32	58.72	58.72	58.72	58.72	58.72	58.72 5	58.72	58.72	58.72	58.72	42.72	0
10	Mosier and Taube [38]	10×10	75.00	75.00	75.00	75.00	75.00	75.00	75.00	75.00	75.00	75.00	75.00	33.74	0
11	Chan and Milner [13]	10×15	92.00	92.00	92.00	92.00	92.00	92.00	92.00	92.00	92.00	92.00	92.00	39.01	0
12	Askin and Subramanian [2]	14×24	72.06	72.06	–	72.06	72.06	69.70	72.06	72.06	73.53	73.53	73.53[a]	53.77	−0.020
13	Stanfel [52]	14×24	71.83	71.83	71.83	71.83	71.83	71.83	71.83	71.83	71.83	71.83	71.83	54.89	0
14	McCormick et al. [37]	16×24	51.58	52.75	–	53.85	53.41	52.08	53.26	53.85	53.33	53.39	53.41	58.27	0.008
15	King [28]	16×43	55.48	57.53	56.38	57.53	57.53	57.23	57.53	57.53	57.31	57.46	57.53	80.61	0
16	Carrie [12]	18×24	57.43	57.73	54.46	57.73	57.73	57.14	57.73	57.73	57.73	57.73	57.73	60.59	0
17	Mosier and Taube [39]	20×20	40.74	43.18	43.26	43.26	43.45	43.06	43.45	43.45	42.55	42.78	43.45	57.75	0
18	Kumar et al. [32]	20×23	49.65	50.81	50.81	50.81	50.81	50.81	50.81	50.81	50.81	50.81	50.81	61.42	0
19	Carrie [12]	20×35	77.02	77.91	78.40	77.91	77.91	76.02	77.91	78.40	77.91	77.91	77.91	75.09	0.006
20	Boe and Cheng [7]	20×35	57.14	57.98	57.61	57.98	57.98	56.54	57.98	57.98	57.98	57.98	57.98	76.24	0
21	Chandrasekharan and Rajagopalan [16]	24×40	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	85.11	0
22	Chandrasekharan and Rajagopalan [16]	24×40	85.11	85.11	85.11	85.11	85.11	85.11	85.11	85.11	85.11	85.11	85.11	86.47	0
23	Chandrasekharan and Rajagopalan [16]	24×40	73.51	73.51	73.51	73.51	73.51	73.51	73.51	73.51	73.51	73.51	73.51	86.61	0
24	Chandrasekharan and Rajagopalan [16]	24×40	52.41	53.29	53.29	53.29	53.29	52.94	53.29	53.29	53.29	53.29	53.29	88.20	0
25	Chandrasekharan and Rajagopalan [16]	27×27	52.53	54.02	54.27	54.82	54.82	53.54	54.82	54.82	54.69	54.79	54.82	75.92	0
26	Stanfel [52]	30×50	57.95	59.77	60.12	58.89	59.77	59.77	60.12	60.12	59.44	59.59	59.77	110.43	0.006
27	McCormick et al. [37]	37×53	52.47	60.64	59.04	60.40	60.64	60.34	60.63	60.64	58.03	59.48	60.64	125.86	0
28	Chandrasekharan and Rajagopalan [16]	16×30	45.77	52.05	–	52.07	52.29	–	–	52.29	52.07	52.18	52.29	64.65	0
29	Chandrasekharan and Rajagopalan [16]	16×30	61.16	62.99	–	63.04	63.04	–	–	63.04	63.04	63.04	63.04	63.16	0
30	Chandrasekharan and Rajagopalan [16]	16×30	64.81	68.38	–	68.38	68.38	–	–	68.38	68.38	68.38	68.38	60.59	0
31	Chandrasekharan and Rajagopalan [16]	16×30	48.18	49.65	–	50.00	50.00	–	–	50.00	49.64	49.96	50.00	60.63	0
32	Chandrasekharan and Rajagopalan [16]	16×30	70.54	72.37	–	72.37	72.37	–	–	72.37	72.37	72.37	72.73[a]	61.32	−0.005
33	Chandrasekharan and Rajagopalan [16]	16×30	77.31	77.31	–	77.31	77.31	–	–	77.31	77.31	77.31	77.31	61.29	0
34	Chandrasekharan and Rajagopalan [16]	16×30	71.43	73.24	–	73.24	73.24	–	–	−73.24	73.24	73.24	73.24	61.43	0
35	Kumar et al. [32]	30×41	61.39	63.31	59.52	62.59	63.31	61.58	63.12	63.31	62.00	62.95	63.31	95.69	0

[a] The solution which singleton appears

Table 4 Comparison between GLCA and the other algorithms in terms of finding better result

		GGA	HGGA	SCM-BMCF	GDE	HGDE	LSA	HM
GLCA	Number of problems which GLCA found better results	21	9	8 out of 25	7	3	14 out of 28	5 out of 28
	Percentage of dominance	60	26	32	20	9	50	18

Table 5 Comparison among all eight algorithms in terms of finding best solution

	GGA	HGGA	SCM-BMCF	GDE	HGDE	LSA	HM	GLCA
Number of best solution	16	27	17 out of 25	29	32	12 out of 28	24 out of 28	Reach 29 and improve 3 solutions
Percentage of reach best solution	46	77	68	83	91	43	86	91 with 3 dominances

gorithm and large neighborhood search (LSA and HM) [19], a hybrid heuristic algorithm adopting both Boltzmann function and mutation operator (SCM-BMCF) [59], and grouping differential evolution algorithm (GDE) and hybrid grouping differential evolution algorithm (HGDE) [40].

The proposed algorithm was coded in Matlab7.4 and implemented on a laptop computer with a 2.5-GHz CPU speed and 2 GB of main memory. Since the parameter combination (L=30, S=10) produces the best improvement in a relatively efficient manner, we decide to use it as the suggested parameter setting for use in GLCA algorithm.

Computational results on all 35 problems are given in Table 3. In this table, we put the worst solution, the average solution, and the best solution among 10 runs of GLCA algorithm. Average computational time in seconds is also reported. In terms of required running time, we do not compare algorithms, since we took the results on LSA, HM, SCM-BMCF, GGA, HGGA, GDE, and HGDE algorithms. Comparison between GLCA and all other problems are given in figures number 6 to 10. It can be verified that GLCA outperforms GGA, HGGA, HGDE, and LSA algorithms in terms of solution quality on all benchmark problems. So, there are no problems for which so-called algorithms are superior to GLCA. GLCA improves the best values of the grouping efficacy found in 3, 12, and 32. In problems number 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 33, 34, and 35 (29 problems), GLCA matches the solution quality which is the best reported one. GLCA outperforms SCM-BMCF in terms of solution quality in all benchmark problems except problem numbers 19 and 26. In problem numbers 3, 7, 15, 16, 17, 20, 25, 27, and 35, GLCA found better results than SCM-BMCF. GLCA exceeds HGGA in problem numbers 3, 12, 14, 17, 25, 28, 29, 31, and 32. In terms of the best case performance, GLCA can compete with HM. There are five problems (3, 12, 14, 17, 27, 35) for which GLCA could provide a better result than HM. There is only one problem for which GLCA performs a little bit worse than HM. This problem is 26. It can be verified that GLCA outperforms GDE in terms of solution quality on all benchmark problems except problem number 14; furthermore, GLCA exceeds GDE in problem numbers 3, 12, 17, 26, 27, 28, 32, and 35. You can see the number of problems which GLCA outperforms other algorithms in Table 4. The percentage of the dominance of the GLCA algorithm to so-called algorithms is also reported. The proposed algorithm differ in solution quality with the best solution found so far in problem numbers 14, 19, and 26 only with 0.7 % average gap.

$$\%\text{gap} = (\text{best solution}-F(X))/\text{best solution}$$

In Table 5, we compare all eight algorithms in terms of finding the best solution. As it is obvious in the table,

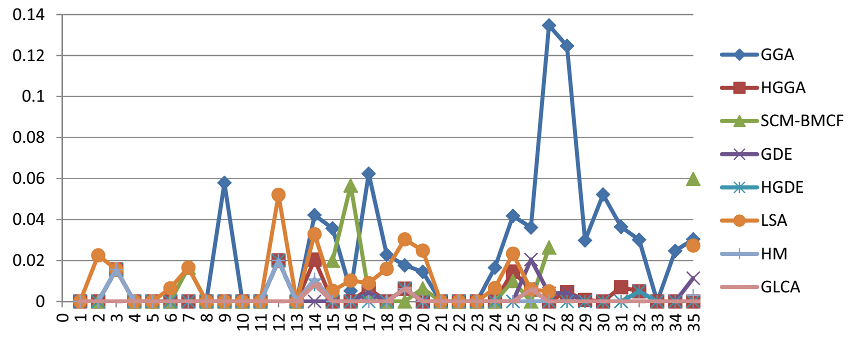

Fig. 5 Comparison among all eight algorithms in terms of finding best solution

Table 6 *M-P* incidence matrix of the Hadi Company

		Machine					
		1 Cut	2 Blend	3 Weld	4 Machinery	5 Dye	6 Assemble
Part	1 Sprinkler	1	0	0	1	1	1
	2 Combine blades	0	1	0	1	0	1
	3 Cultivator	1	0	1	0	1	1
	4 Transplant tray	0	1	0	1	0	1
	5 Chisel plough	1	1	0	1	1	1
	6 Castration forceps	1	0	0	0	1	1
	7 Milking cows	1	0	1	0	0	1
	8 Dripper	0	1	0	1	0	1

GLCA is the best algorithm in this term because it reaches 29 best solutions and improves 3 best so far solutions.

Results indicate that the performance of GLCA is rather constant on all types of problems ranging from small to large. This means that GLCA performs dependably in the sense that it does its best on almost all types of problems. Therefore, GLCA is not only able to produce good solutions but also can be considered as a robust heuristic algorithm. Considering the performance of the GLCA algorithm on all test problems, the proposed algorithm should thus be useful to both practitioners and researchers. GLCA improved the best-known solution of three problems with the 1.4 % average gap.

In Fig. 5, you can see the gap between the result of each of eight algorithms and the best solution among all eight algorithms. As you can see, GLCA has a minimum difference with the best solution.

7 Case study

To illustrate the performance of the proposed algorithm, a real industrial case study is presented.

7.1 Outline of the case study

Case study is conducted for a typical equipment manufacturer so-called Hadi Company (was founded in 1982) in the north of Iran. The company is a type of small and medium enterprises (SMEs); thus, application of CM may obtain considerable advantages. Hadi Company produces different types of farm equipment. Along with an increment in production rate, the variety of other types of equipment is increased. The area of the factory is 1500 m^2 with the production section in which 80 workers and specialist work 7 days a week.

Eight part types (farm equipment) are produced in the company including sprinkler, combine blades, cultivator, transplant tray, chisel plough, castration forceps, milking cows, and dripper.

The company consist of six stations: cutting, bending, welding, machinery, dyeing, and assembly.

To propose implementation of our algorithm, we considered each station as a unique machine.

7.2 Results of the case study

The machine-part incidence matrix of Hadi Company is presented in Table 6. The proposed algorithm is coded in Matlab7.4 and implemented on a laptop computer with a 2.5-GHz CPU speed and 2 GB of main memory. Parameter combination (L=30, S=10) is implemented on GLCA.

After rearranging the rows and colons of the *M-P* incidence matrix, the cell formation schema of the case study is presented in Table 7.

As a result of the algorithm, it can be seen that we should have two cells to have the most efficient cell configuration. Specifically, cell 1 contains four machines: cutter, welder, dyer, and assembler, and five parts consisting sprinkler, cultivator, chisel plough, castration forceps, and milking cows.

Table 7 *M-P* incidence matrix of the case study after diagonalization process

		Machine					
		1	3	5	6	2	4
Part	1	1	0	1	1	0	1
	3	1	1	1	1	0	0
	5	1	0	1	1	1	1
	6	1	0	1	1	0	0
	7	1	1	0	1	0	0
	2	0	0	0	1	1	1
	4	0	0	0	1	1	1
	8	0	0	0	1	1	1

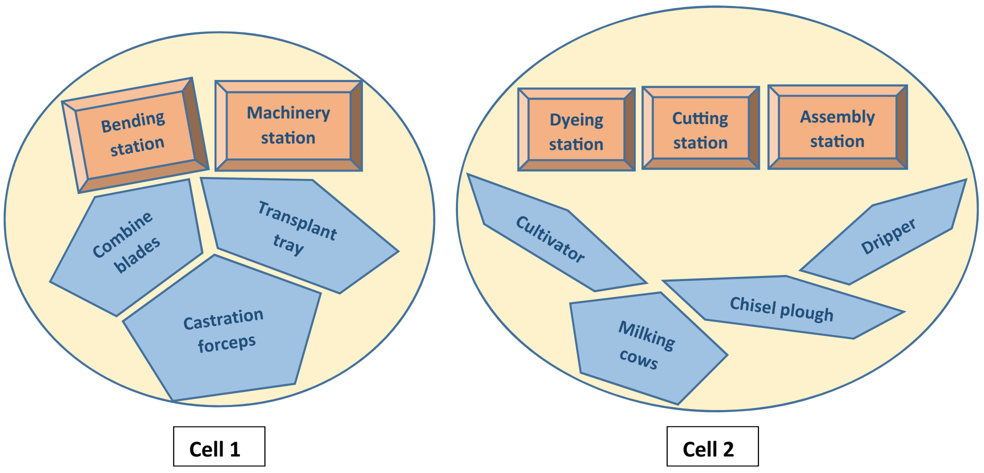

Fig. 6 Cell formation schema of the case study

Cell 2 contains two machines: blender and dyer, and three parts containing combine blades, transplant tray, and dripper.

Part families, machine groups, and exceptional elements are also depicted in Table 7. As it can be seen, there are six exceptional elements (1 s outside the diagonal blocks) and four voids (0 s inside the diagonal blocks). The cell formation schema of the case study is presented in Fig. 6.

Grouping efficacy measure of this case study is 68.75 which is multiplied by 100 base of the comparing table (Table 3).

Considering the results, it can be seen that the proposed solution procedure can be easily applied to SMEs exploiting the profits of CM.

8 Conclusions

In this paper, we present a new solution approach for the machine-part cell formation problem based on the grouping representation used in the grouping genetic algorithm. We proposed equations analogous to those of the classical LCA equations, yielding a grouping version of LCA (GLCA). We used a truncated geometric algorithm to evaluate the number of initial cells.

The results justify that GLCA outperforms other algorithms in the terms of solution quality. We compared the performance of GLCA with the grouping genetic algorithm (GGA), hybrid grouping genetic algorithm (HGGA), simulated annealing algorithm, genetic algorithm and large neighborhood search (LSA and HM), hybrid heuristic algorithm adopting both Boltzmann function and mutation operator (SCM-BMCF), and grouping differential evolution algorithm and hybrid grouping differential evolution algorithm (GDE and HGDE). We found that GLCA algorithm outperforms other algorithms in terms of solution quality in 32 out of 35 benchmarked problem instances. For the rest of the problems, GLCA differ in solution quality with 0.7 % average gap. Furthermore, GLCA found better results for three problems. For future research, the extension of our approach could be investigated for other types of grouping problems, e.g., bin packing problem, graph coloring problem, etc.

References

1. Anvari M, Mehrabad M, Barzinpour F (2010) Machine-part cell formation using a hybrid particle swarm optimization. Int J Adv Manuf Technol 2010:745–754
2. Askin RG, Subramanian SP (1987) A cost-based heuristic for group technology configuration. Int J Prod Res 25:101–113
3. Ateme-Nguem B, Dao T (2007) Optimization of cellular manufacturing systems design using the hybrid approach based on the ant colony and tabu search techniques. In Proceedings of the IEEE IEEM, pages 668–673
4. Ateme-Nguema B, Dao T (2009) Quantized Hopfield networks and tabu search for manufacturing cell formation problems. Int J Prod Res 121:88–98
5. Batsyn M, Bychkov I, Goldengorin B, Pardalos P, Sukhov P (2013) Pattern-based heuristic for the cell formation problem in group technology. B. Goldengorin et al. (eds.), Models, algorithms, and technologies for network analysis, Springer Proceedings in Mathematics & Statistics, 32, 11-50
6. Boctor FF (1991) A linear formulation of the machine-part cell formation problem. Int J Prod Res 29:343–356
7. Boe WJ, Cheng CH (1991) A close neighbor algorithm for designing cellular manufacturing systems. Int J Prod Res 29:2097–2116
8. Boulif M, Atif K (2006) A new branch-&-bound-enhanced genetic algorithm for the manufacturing cell formation problem. Comput Oper Res 33:2219–2245
9. Boutsinas B (2013) Machine-part cell formation using biclustering. Eur J Oper Res 230:563–572

10. Brown E, James T (2007) A hybrid grouping genetic algorithm for the cell formation problem. Comput Oper Res 34: 2059–2079
11. Brown E, Sumichrast R (2001) CF-GGA: a grouping genetic algorithm for the cell formation problem. Int J Prod Res 36:3651–3669
12. Carrie AS (1973) Numerical taxonomy applied to group technology and plant layout. Int J Prod Res 399-416
13. Chan HM, Milner DA (1982) Direct clustering algorithm for group formation in cellular manufacturing. Journal of Manufacturing Systems 1: 65–74
14. Chandrasekharan MP, Rajagopalan R (1986) MODROC: an extension of rank order clustering for group technology. Int J Prod Res 24:1221–1233
15. Chandrasekharan MP, Rajagopalan R (1986) An ideal seed nonhierarchical clustering algorithm for cellular manufacturing. Int J Prod Res 24:451–463
16. Chandrasekharan MP, Rajagopalan R (1989) Groupability: an analysis of the properties of binary data matrices for group technology. Int J Prod Res 27:1035–1052
17. Cheng C, Gupa Y, Lee WA (1998) TSP-based heuristic for forming machine groups and part families. Int J Prod Res 36:1325–1337
18. DeLit P, Falkenauer E, Delchambre A (2000) Grouping genetic algorithms: an efficient method to solve the cell formation problem. Math Comput Simul 51:257–271
19. Elbenani B, Ferland J, Bellemare J (2012) Genetic algorithm and large neighborhood search to solve the cell formation problem. Expert Syst Appl
20. Falkenauer E (1998) Genetic algorithm for grouping problems. Wiley, New York
21. Ghosh T, Sengupta S, Chattopadhyay M, Dan KP (2011) Metaheuristics in cellular manufacturing: a state-of-art review. Int J Ind EngComput 2:87–122
22. Goldengorin B, Krushinsky D, Slomp J (2012) Flexible PMP approach for large size cell formation. Oper Res 60:1157–1166
23. Goncalves J, Resende M (2004) An evolutionary algorithm for manufacturing cell formation. Comput Ind Eng 47:247–273
24. Heragu SS (1997) Facility design. PWS Publishing Company, Boston
25. Husseinzadeh Kashan A (2009) League championship algorithm: a new algorithm for numerical function optimization. International Conference of Soft Computing and Pattern Recognition
26. Husseinzadeh Kashan A, Karimi B, Jolai F (2006) Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes. Int J Prod Res 44:2337–2360
27. Husseinzadeh Kashan A, Karimi B, Noktehdan A (2014) A novel discrete particle swarm optimization algorithm for the manufacturing cell formation problem. Int J Adv Manuf Technol
28. King JR (1980) Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm. Int J Prod Res 213-232
29. King JR, Nakornchai V (1982) Machine-component group formation in group technology: review and extension. Int J Prod Res 20: 117–131
30. Krushinsky D, Goldengorin B (2012) An exact model for cell formation in group technology. Comput Manag Sci 9:323–338
31. Kumar C, Chandrasekharan MP (1990) Grouping efficacy: a quantitative criterion for goodness of block diagonal forms of binary matrix in group technology. Int J Prod Res 28:233–243
32. Kumar KR, Kusiak A, Vannelli A (1986) Grouping of parts and components in flexible manufacturing systems. Eur J Oper Res 24: 387–397
33. Kusiak A, Cho M (1992) Similarity coefficient algorithm for solving the group technology problem. Int J Prod Res 30:2633–263346
34. Kusiak A, Chow WS (1987) Efficient solving of the group technology problem. J Manuf Syst 6:117–124
35. Lei D, Wu Z (2005) Tabu search approach based on a similarity coefficient for cell formation in generalized group technology. Int J Prod Res 19:4035–4047
36. Lei D, Wu Z (2006) Tabu search for multiple-criteria manufacturing cell design. Int J Adv Manuf Technol 28:950–956
37. McCormick WT, Schweitzer PJ, White TW (1972) Problem decomposition and data reorganization by a clustering technique. Oper Res 20:993–1009
38. Mosier CT, Taube L (1985) The facets of group technology and their impact on implementation. OMEGA
39. Mosier CT, Taube L (1985) Weighted similarity measure heuristics for the group technology machine clustering problem. OMEGA
40. Noktehdan A, Karimi B, Husseinzadehkashan A (2010) A differential evolution algorithm for the manufacturing cell formation problem using group based operators. Expert Syst Appl 37:4822–4829
41. Noktehdan A, Seyedhosseini SM, Saidi-Mehrabad M (2015) A Metaheuristic algorithm for the manufacturing cell formation problem based on grouping efficacy. Int J Adv Manuf Technol
42. Pailla A, Trindade A, Parada V, Ochi L (2010) A numerical comparison between simulated annealing and evolutionary approaches to the cell formation problem. Expert Syst Appl 37:5476–5483
43. Prabhaharan G, Muruganandam A, Asokanm P, Girish BS (2005) Machine cell formation for cellular manufacturing systems using an ant colony system approach. Int J Adv Manuf Technol 25:1013–1019
44. Rao PK (2014) A multi stage heuristic for manufacturing cell formation. Int J Res EngTechnol 1163:2308–2321
45. Roy N, Komma VR (2014) Cellular manufacturing through composite part formation: a genetic algorithm approach. International Conference on Industrial Engineering and Operations Management, Bali, Indonesia, January 7 – 9
46. Sarker B (2001) Measures of grouping efficacy in cellular manufacturing systems. Eur J Oper Res 130:588–611
47. Sayadi MK, Hafezalkotob A, Jalali naini SG (2013) Firefly-inspired algorithm for discrete optimization problems: an application to manufacturing cell formation. J Manuf Syst 32: 78–84
48. Seifoddini H (1989) A note on the similarity coefficient method and the problem of improper machine assignment in group technology applications. Int J Prod Res 27: 1161–1165
49. Seifoddini H, Wolf PM (1986) Application of the similarity coefficient method in group technology. IIIE Transaction 18: 271–277
50. Selim HM, Askin RG, Vakharia AJ (1998) Cell formation in group technology: evaluation and directions for future research. Comput Ind Eng 34:3–20
51. Solimanpur M, Saeedi S, Mahdavi I (2010) Solving cell formation problem in cellular manufacturing using ant-colony-based optimization. Int J Adv Manuf Technol 50:1135–1144
52. Stanfel LE (1985) Machine clustering for economic production. Eng Costs Prod Econ 9:73–81
53. Stawowy A (2006) Evolutionary strategy for manufacturing cell design. Omega, Int J Manag Sci 34:1–18
54. Sun D, Lin L, Batta R (1995) Cell formation using tabu search. Comput Ind Eng 28:485–494
55. Vin E, DeLit P, Delchamber A (2005) A multiple-objective grouping genetic algorithm for the cell formation problem with alternative routings. J Intell Manuf 16:189–205
56. Waghodekar PH, Sahu S (1984) Machine-component cell formation in group technology MACE. Int J Prod Res 22: 937–948

57. Wemmerlov U, Hyer NL (1989) Cellular manufacturing in the US industry: a survey of users. Int J Prod Res 27:1511–1530
58. Wu T, Chang C, Chung S (2008) A simulated annealing algorithm for manufacturing cell formation problems. Expert Syst Appl
59. Wu T, Chang C, Yeh J (2009) A hybrid heuristic algorithm adopting both Boltzmann function and mutation operator for manufacturing cell formation problems. Int J Prod Econ 120:669–688
60. Wu T, Chung S, Chang C (2010) A water flow-like algorithm for manufacturing cell formation problems. Eur J Oper Res 205:346–360