sustainability

MDPI

Article

Sustainability Formation of Machine Cells in Group Technology Systems Using Modified Artificial Bee Colony Algorithm

Adinarayanan Arunagiri [1,*], Uthayakumar Marimuthu [2], Prabhakaran Gopalakrishnan [3], Adam Slota [4], Jerzy Zajac [4] and Maheandera Prabu Paulraj [5]

1 Department of Mechanical Engineering, Dhaanish Ahmed College of Engineering, Chennai 601301, India
2 Faculty of Mechanical Engineering, Kalasalingam University, Krishnankovil 626126, India; m.uthayakumar@klu.ac.in
3 Department of Mechanical Engineering, KCG College of Technology, Chennai 600097, India; principal@kcgcollege.com
4 Institute of Production Engineering, Cracow University of Technology, 31-155 Krakow, Poland; slota@mech.pk.edu.pl (A.S.); zajac@mech.pk.edu.pl (J.Z.)
5 Department of Mechanical Engineering, Indian Institute of Technology Indore, Simrol, Indore 453552, India; prabusita@iiti.ac.in
* Correspondence: adinarayanan.a@dhaanishcollege.in; Tel.: +91-097-9072-1425

Received: 24 November 2017; Accepted: 20 December 2017; Published: 28 December 2017

Abstract: The efficiency and sustainability of a cellular manufacturing system (CMS) in batch type manufacturing is highly valued. This is done using a systematic method of equipment into machine cells, and components into part families, based on the suitable similar criteria. The present work discusses the cell formation problem, with the objective of minimizing the cumulative cell load variation and cumulative intercellular moves. The quantity of parts, operation sequences, processing time, capacity of machines, and workload of machineries were considered as parameters. For the grouping of equipment, the modified artificial bee colony (MABC) algorithm is considered. The computational procedure of this approach is explained by using up to 40 machines and 100 part types. The result obtained from MABC is compared with the findings acquired from the genetic algorithm (GA) and ant colony system (ACS) in the literature.

Keywords: cellular manufacturing system; sustainability; modified artificial bee colony algorithm; intercellular moves; cell load variation

1. Introduction

Group technology (GT) is a production strategy where the similar components are picked out and grouped together to gain the benefits of their similarities in design and/or production character [1]. Cellular manufacturing (CM) is based on the principles of group technology. Cellular manufacturing system plays a major role in the application to industries. It is an important technique to cope with the rapid changing industrial demands and new innovations. In CM, the equipment is located in close proximity, and sacrificed to make all necessary operations into the particular part family, and provide smooth flow of materials within the cell which lead to high productivity. The equipment in CM system permits the equipment to be changed or relocated whenever new part designs are incorporated, and product demand changes with minimum effort, in terms of cost and time [2]. GT concept has been implemented in various factories like machineries, automobiles, defense, and electrical industries [3]. There are many advantages in cellular manufacturing system (CMS) which include minimizing the processing cost, material handling cost, machine duplication cost, cycle times, material handling times, work-in-process inventory levels, factory space requirements,

product defect rates, and machine idle times [4,5]. There have been many investigations done on part families and machine cell problems during the past three decades. Opitz [6] described a method of part classification and coding analysis for effective production planning and control. Burbidge [7] explained the material flow system with process organization and process to product organization in production flow analysis. The problem comprises of a group of equipment, group of components, and assigning of components and equipment into the part families. Rank order clustering method was proposed by King [8]. The main purpose is to devise a generalized procedure that would convert, in a finite number of iterations, any original machine–part matrix into a clustered diagonal form, if one exists. Srinivasan et al. [9] discussed the similarity coefficient method, which describes that this method is more flexible for incorporating production data, such as production volume, sequence of operation, and operational time, into the machine cell formation. Heuristic algorithms [10,11] have been implemented for many cell formation (CF) solutions. In the present days, most of the cell formation problems have been solved by using meta-heuristic algorithms. Meta-heuristic search approaches, such as simulated annealing [12–14] and a genetic algorithm [15], have also been used to solve various cell formation problems. Prabhakaran [16] has discussed the cell load variation and intercellular moves in cell formation problem using genetic algorithm (GA). Sarac and Ozcelik [17] formulated a genetic algorithm with proper parameters for manufacturing cell formation problems. In their paper, the cell formation in cellular manufacturing systems is considered with the objective of maximizing the grouping efficacy. Arkat et al. [18] employed the minimization of exceptional elements and voids in the cell formation problem using a multi-objective genetic algorithm, and presented a bi-objective mathematical model to simultaneously minimize the number of exceptional elements and the number of voids in the part machine incidence matrix. An e-constraint method is then applied to solve the model and to generate the efficient solutions. Other metaheuristics, like tabu search [19,20], ant colony optimization (ACO) [21,22], particle swarm optimization [23,24], and scatter search [25] have also been proposed for designing CMS. Chattopadhyayet et al. [26] discussed machine–part cell formation through visual decipherable clustering of self-organizing map. Their paper deals with the self-organizing map method, an unsupervised learning algorithm in artificial intelligence which has been used as a visually decipherable clustering tool of machine–part cell formation. Ghezavati and Saidi-Mehrabad [27] proposed an efficient hybrid self-learning method for stochastic cellular manufacturing problems. It addresses a new version of stochastic mixed-integer model to design cellular manufacturing systems under random parameters, described by continuous distributions. Ying-Chin and Ta-Wei [28] explained about the concurrent solution for intra-cell flow path layouts and I/O point locations of cells in a cellular manufacturing system. In this study, the authors propose a layout procedure that can solve these two problems at the same time, so that the sum of the inter-cell flow distance and the intra-cell flow distance can be minimized. Venkumar and Haq [29] discussed the complete and fractional cell formation using neural network methodology. The other recent approaches of CMS design include a new branch-and-bound algorithm [30], and the firefly-inspired algorithm [31]. Mohammad and Kamran [32] proposed an integrated bi-objective layout and cell formation problem. In this problem, the main aim is to minimize the total inter- and intra-cell material handling costs, and the second aim is to maximize the total similarity between machines. Behrang et al. [33] explained mathematical models in cellular manufacturing systems for clustering workers and machines in product mix variation case. Sakhaii et al. [34] developed a robust optimization approach for a new integrated mixed-integer linear programming model to solve a dynamic cellular manufacturing system with unreliable machines and a production planning problem simultaneously. Brown [35] proposed optimized manufacturing model to minimize costs associated with exceptional elements. Costs are minimized through intercellular transfer, machine duplication, and subcontracting. Kia et al. [36] explained a mixed-integer programming model for a multi-floor layout design of cellular manufacturing systems (CMS) in a dynamic environment. Yung Chin et al. [37] explained a design procedure for improving the effectiveness of fractal layouts formation aiming at minimization of routing distances. Zahra et al. [38] explained the task scheduling Non-polynomial (NP) hard problem

in grid computing system using particle swarm optimization–gravitational emulation local search (PSO–GELS). In this problem, the PSO–GELS combined algorithm has provided better results than other approaches. Mohammad et al. [39] discussed the job scheduling problem in cloud computing, using the methodology of fuzzy theory and genetic algorithm. In this paper, the main aim is to perform optimal load balancing considering execution time and cost. Mohammad et al. [40] described a fuzzy reputation-based model for trust management in semantic P2P grids. Javanmardi et al. [41] present a hybrid job scheduling approach, which considers the load balancing of the system, and reduces total execution time and execution cost using genetic algorithm and fuzzy theory.

In the recent years, there are so many population based meta heuristic evolutionary algorithms used to solve cell formation problems due to their ease of implementation. However, these intelligence algorithms are sensitive to value and precision. It inspired us to use modified artificial bee colony (MABC) for cell formation problem. In this paper, the MABC algorithm, a meta heuristic population based algorithms is used in cell formation problem with the aim of reducing cumulative cell load variation and cumulative intercellular moves. The outcome of this approach is compared with that of both GA and ACO.

2. Problem Formulation

In this problem, the most cited performance measures, such as cumulative intercellular moves and cumulative cell load variation, are to be calculated. The constraint used is a minimum of two machines in a cell.

2.1. Cumulative Intercellular Moves

The movement of parts between the cells is known as intercellular movement. It deals with the impact of processing sequence and cell layout, which is explained [42].

$$\text{Cumulative moves} = \sum_{i=0}^{P} \sum_{i=0}^{k-1} k = 1 | C_k - C_{k+1} | \tag{1}$$

where C_k = cell number and processing k is carried out on component i, considering the operation sequence; C_{k+1} = cell number in which and processing $k + 1$ is carried out on component i, considering the operation sequence; k_i = total number of processes carried out on component i to finish its necessary operations; C = cell counting; P = part counting.

2.2. Cumulative Cell Load Variation

The cell load variation is calculated by the difference between the equipment workload and the machine cell average load. A lower value of cell load variation is preferable, since it leads to efficient flow of parts and lowers the inventory level. The cumulative cell load variation is measured using the formula suggested [15].

$$\text{Cell load variation} = \sum_{i=1}^{m} \sum_{l=1}^{c} x_{il} \sum_{j=1}^{p} \left(w_{ij} - m_{ij} \right)^2 \tag{2}$$

where m = machine counting; c = cell counting; p = part counting.

$W = [w_{ij}]$ is an $m \times p$ workload matrix, where

$$w_{ij} = \frac{(t_{ij} \times N_j)}{T_i}, \tag{3}$$

where t_{ij} = manufacturing time (hours/piece) of job j on equipment i; T_i = available time on equipment i for the allotted time period; N_j = manufacturing demand of part j for the allotted time period; $X = [x_{il}]$

is an $m \times c$ cell membership matrix, where $x_{il} = 1$ if the i-th equipment is in cell 1, and 0 otherwise; $M = [m_{lj}]$ is a $c \times p$ matrix of the average machine cell load.

Where

$$m_{lj} = \frac{\sum_{i=1}^{m} (x_{il} \times w_{ij})}{\sum_{i=1}^{m} x_{il}}. \quad (4)$$

2.3. The Combined Objective Function

$$\text{Min } Z = W_1 \left\{ \sum_{i=0}^{P} \sum_{i=0}^{K-1} K = 1 | C_k - C_{K+1} | \right\} + W_2 \left\{ \sum_{i=1}^{m} \sum_{l=1}^{c} x_{il} \sum_{j=1}^{P} (w_{ij} - m_{ij})^2 \right\} \quad (5)$$

where W_1 and W_2 are weights. By varying the values of W_1 and W_2, the decision maker can decide which function should be given a high priority.

2.4. Existing ABC Algorithm

Karaboga [43–46] proposed that ABC algorithm has been generated by imitating the behavior of the honey bee swarm intelligence. This algorithm has been implemented for numerous engineering problems. Its development was instigated through the intelligent foraging character of bees in their colony, and their performance was measured by the benchmark optimization function. Kalayci et al. [47] clearly explained the behavior of bees to solve disassembled line balancing problems. Particularly, the number of control parameters in ABC is fewer than the other population-based algorithms. Moreover, the optimization characteristics of ABC are comparable, and in some cases, much better than the state-of-the-art meta-heuristics. Hence, ABC is applied to the many types of optimization problems.

In this problem, bees represent a solution. There are three types of bees available in the ABC colony. They are employed bees, onlookers, and scout bee. It is assumed that there is only one artificial employed bee for each food source. In other words, the number of employed bees in the colony is equal to the number of food sources around the hive. Employed bees go to their food source, and come back to the hive and dance in this area. The employed bee whose food source has been abandoned becomes a scout, and starts to search for a new food source. Onlookers watch the dances of employed bees, and choose food sources depending on dances. The main steps of the algorithm are given below:

Initial food sources are produced for all employed bees.

REPEAT

Each employed bee goes to a food source in her memory and determines a neighbor source, then evaluates its nectar amount and dances in the hive.

Each onlooker watches the dance of employed bees and chooses one of their sources depending on the dances, and then goes to that source. After choosing a neighbor around that, she evaluates its nectar amount.

Abandoned food sources are determined and are replaced with the new food sources discovered by scouts.

The best food source found so far is registered.

UNTIL (requirements are met)

2.5. Proposed Modified ABC Algorithm

The basic ABC algorithm was originally designed for continuous function optimization. But especially for the NP hard combinatorial optimization problems, some modifications are

required in the basic ABC approach. The modified ABC approach (MABC) is described in the following subsections.

2.6. Representation

The length of a solution string represents the number of equipment considered in the problem. Each place in the solution string indicates the machine number, and the position number indicates the equipment present in that position. The position of any equipment represents its location.

Example: 1 3 5 2 4 8 7 6 9

Here, the length of the solution string is 9, which indicates that the number of equipment considered in the problem is 9. The places in the solution string take the values 1 to 9, which indicate the equipment number. Position of equipment 1 is 1, 3 is 2, and so on.

2.7. Population Initialization

The size of the initial population is formulated randomly. An initial population of the desired size is generated randomly. In this proposed MABC, 40 solution strings are randomly generated and used as initial population. Each solution string is converted into a 3×3 matrix, and the objective value is calculated using the objective function equation.

Example: 1 3 5 2 4 8 7 6 9

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 8 \\ 7 & 6 & 9 \end{bmatrix} \tag{6}$$

2.8. Employed Bee Phase

The employed bees initialize food origin near their present locations. In this phase, insert and swap manipulators are used to produce nearby findings. The insert manipulator of a string, S_i, is explained by taking away a bit from its initial location j, and inserting it into another location, k, such that ($k \in \{j, j-1\}$), where as the swap manipulator generates a nearby of S_i by exchanging two bits of S_i in various positions. To improve the neighborhood structure and spread out the population, the following two neighboring approaches are utilized to generate neighboring food sources for the employed bees.

Approach 1. Perform one insert operator to a solution S_i.

Initially, the 3rd bit is selected randomly. Then, the selected bit was inserted into 5th bit.
Before insert operation: 1 3 **5** 2 4 8 7 6 9;
After insert operation: 1 3 2 4 5 8 7 6 9.

Approach 2. Perform one swap operator to a solution S_i.

Initially, the 3rd and 7th bits are selected randomly, and swapping is performed.
Before swap operation: 1 3 5 2 4 8 7 6 9;
After swap operation: 1 3 7 2 4 8 5 6 9.

One of the neighboring approaches is selected randomly to produce new food sources. If the new food source is better than the current food source, then the new food source is accepted.

2.9. Onlooker Bee Phase

In the basic ABC algorithm, an onlooker bee selects a food source, x_i, depending on its winning probability value p_i, which is similar to the wheel selection in genetic algorithms [16]. In this MABC, the tournament selection with the size of 2 is used, due to its simplicity and ability to escape from

local optima. In the tournament selection, an onlooker bee selects a food source, x_i, in such a way that two food sources are picked up randomly from the population, and compared to each other, and then the better one is chosen. In addition, the onlooker utilizes the same method as used by the employed bee to produce a new neighboring solution. If the new obtained food source is better than, or equal to the current one, the new food source will replace the current one, and become a new member in the population.

2.10. Scout Bee Phase

In the basic ABC algorithm, a scout produces a food source randomly in the predefined search scope. This will decrease the search efficacy, since the best food source in the population often carries better information than others during the evolution process, and the search space around it could be the most promising region. In the MABC algorithm, the scout generates a food source by utilizing the same method as used by the employed bee to produce a new neighboring solution on the best food source.

2.11. Steps for Computational Procedure of the Proposed MABC Algorithm

Step 1: Initialize the parameters SN, PS, Limit, and PL. (In MABC, SN is equal to PS and number of Onlookers.)

Where SN denotes the size of the population, PS indicates the food source count, limit specifies control parameter and PL denotes the cycles count.

Step 2: Initialize population $S = \{s_1, s_2, \ldots, s_{PS}\}$ and calculate every result in the population.

Step 3: Employed bees:

For $i = 1, 2, \ldots,$ PS, repeat the following sub-steps:

Produce a new solution S_i* for the i-th employed bee who is associated with solution S_i by using the strategy presented in the subsection employed bee phase, and evaluate the new solution.

If S_i* is better than or equal to S_i, let $S_i = S_i$*.

Step 4: Onlooker bee:

For each $i = 1, 2, \ldots,$ PS, repeat the following sub-steps:

Select a food source in the population for the onlooker bee, S_i, by using the tournament selection presented in the subsection onlooker bee phase.

Generate a new solution, S_i*, for the onlooker by using the strategy presented in the sub section employed bee phase and evaluate it.

If S_i* is better than or equal to S_i, let $S_i = S_i$*.

Step 5: Scout bee:

For each $i = 1, 2, \ldots,$ PS, repeat the following sub-steps:

If a solution S_i in the population has not been improved during the last limit number of trials, abandon it.

Generate a new solution onlooker by using the strategy presented in the subsection Employed bee phase on the best solution.

Step 6: Store the best result achieved so far.

Step 7: If the termination criterion is reached, return the best solution found so far; else go to step 3.

3. Results and Discussion

The proposed algorithm is coded in C++ language on a personal computer with a 1.3 GHz Pentium IV CPU processor (Intel, Bengaluru, India) and 16 GB RAM memory (Intel, Bengaluru, India). Different parameters, such as machine workload, operation sequences, unit processing times on every equipment, and manufacturing quantity, among others, are considered in this study. From the literature [16], 20 data sets have been taken, and the experimental works have been carried out using the proposed MABC algorithm, and the outcomes were compared with those of the other two approaches, GA [16] and ACO [21]. The parameter settings selected for the problem, its implementation, and the results obtained are discussed below:

MABC Parameter Setting: For the given problem structure, the parameter settings are shown in Table 1.

Table 1. MABC parameter setting.

PS	PL	Limit
40 (no. of food sources)	500 (no. of cycles)	10 (no. of trials)

PS denotes the food source count, PL denotes the cycles count and Limit specifies control parameter.

The modified ABC algorithm is implemented, and the results are simulated for different problem sizes. The result obtained for the MABC problem, with respect to the problem sizes considered, are given below. The problem sizes considered for the present investigation is 8 × 20, 10 × 12, and 10 × 15.

Figure 1 shows the MABC convergence curve for intercellular moves of problem size 8 × 20. As can be seen in the figure, the increase of cycles minimizes the intercellular movement. Further, the result indicates that number of cells consisting of 2 indicates the minimum intercellular moves. Normally, the increase of cycles tries to minimize the intercellular move. Whereas for the 3 cell problem, the trend is almost linear up to 300 cycles, after which it tries to minimize, up to 400 cycles. The 400 to 500 cycles result does not indicate the possible variations. That is, variations obtained for cycles 400–500 are very minimal. For the cell number 4, drastic changes are observed in intercellular moves from the cycles 100 to 200, and minimum variation from the cycle of 200 to 500.

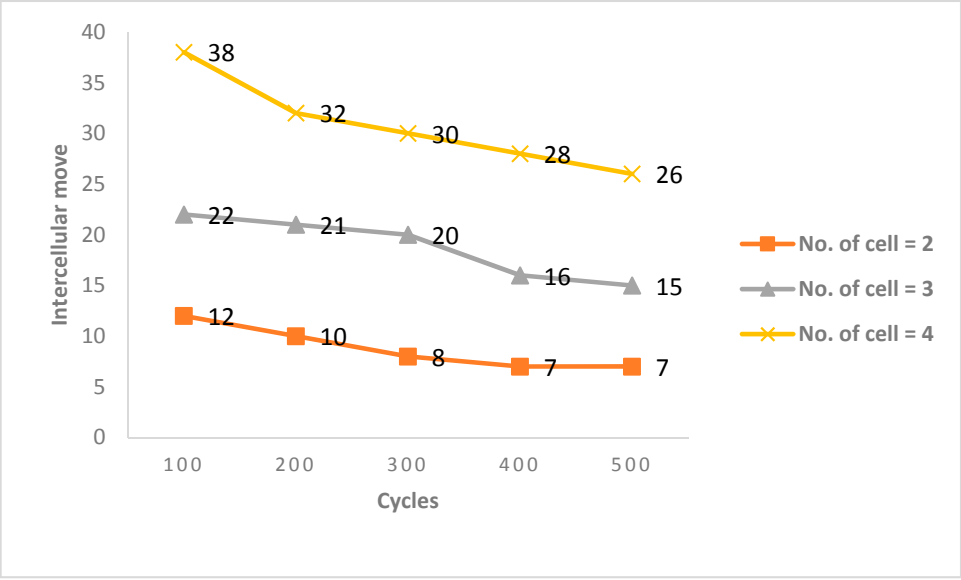

Figure 1. MABC convergence curve for intercellular move of problem size 8 × 20.

Figure 2 shows the MABC convergence curve for cell load variation of problem size 8 × 20. The result from the graph indicates that the increase in cycles minimizes the cell load variations. In the above figure, the result indicates that cell number 4 has minimum cell load variations. In the case of 3 cell problem, the curve is almost linear from the cycle of 100 to 300, after which it tries to minimize, up to 500 cycles. But in the 2 cells problem, there is a drastic change from 100 to 200 cycles, and then a linear decrease of cell load variation continues up to 500 cycles.

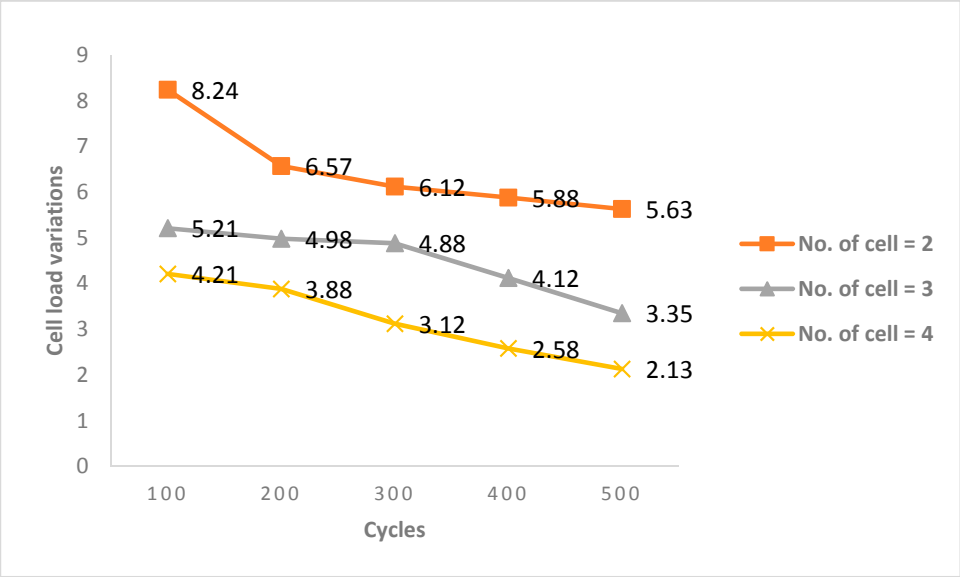

Figure 2. MABC convergence curve for cell load variation of problem size 8 × 20.

Figure 3 shows the MABC convergence curve for intercellular move for the problem size 10 × 12. In the above figure for the cell number 2, 3, and 4, the curve is almost linear from 100 to 300 cycles. But for the cell number 5, there is drastic change of curve from 400 to 500 cycles.

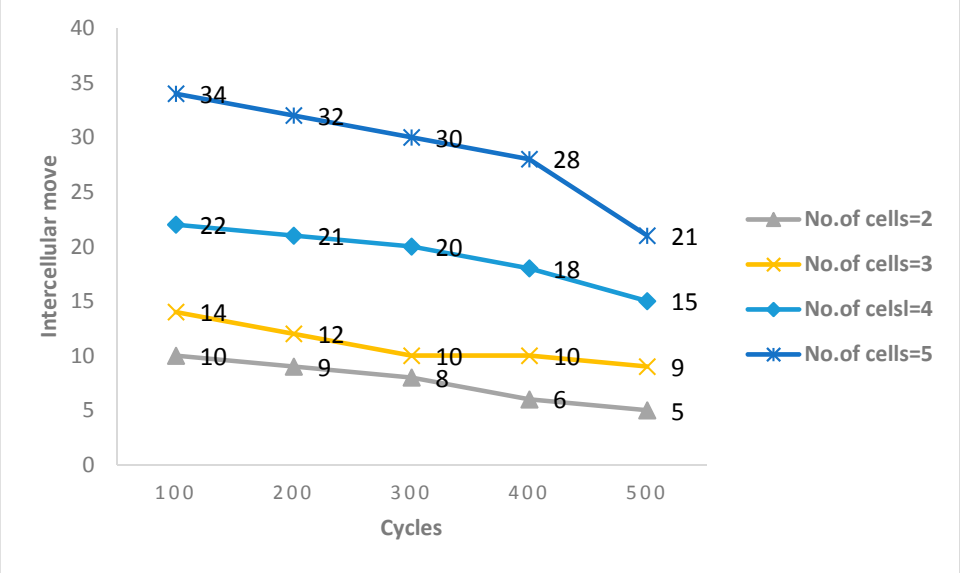

Figure 3. MABC convergence curve for intercellular move of problem size 10 × 12.

Figure 4 shows the MABC convergence curve for cell load variation of problem size 10 × 12. The above figure indicates that in cell 4, cell load variation is minimized from the cycle of 100 to 300,

and there is no significant improvement from the cycle of 300 to 500. For cell number 2, 3, and 5, the curve is linear from the cycle of 100 to 300.

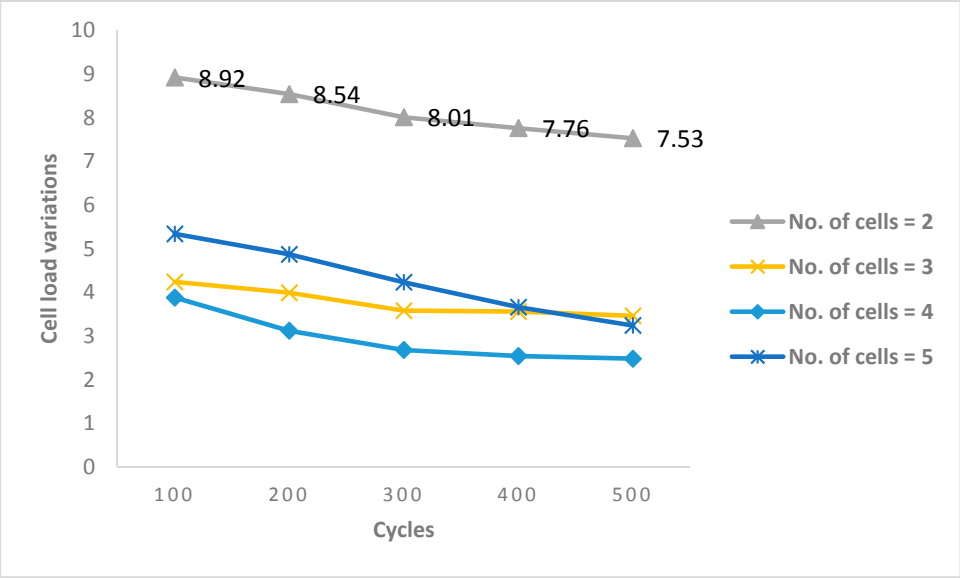

Figure 4. MABC convergence curve for cell load variation of problem size 10 × 12.

Figure 5 shows the MABC convergence curve for intercellular move for the problem size 10 × 15. In the above figure, all the curves are linear from the cycle of 100 to 300. In this figure, cell number 2 reaches zero intercellular moves at the cycle of 500.

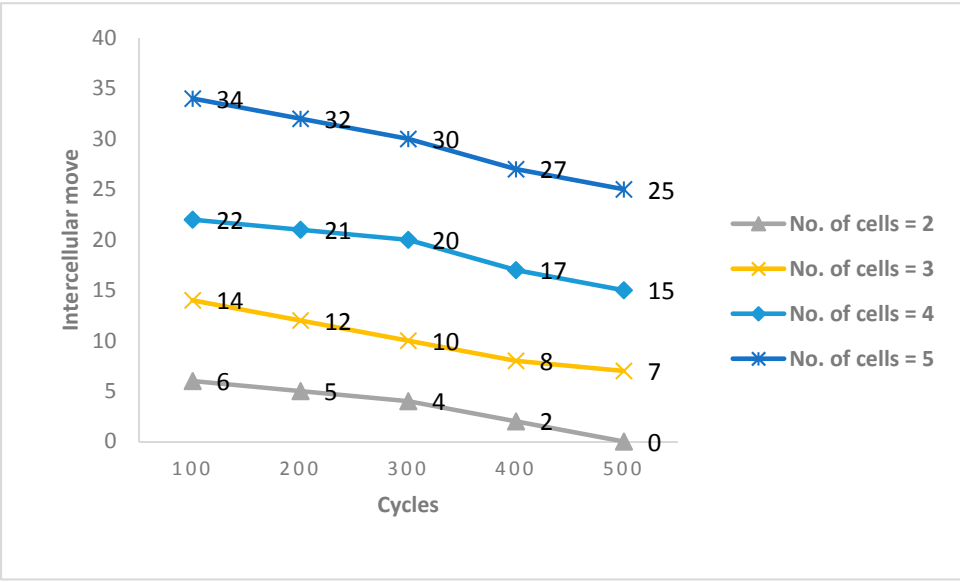

Figure 5. MABC convergence curve for intercellular move of problem size 10 × 15.

Figure 6 shows the MABC convergence curve for cell load variation of problem size 10 × 15. In the above figure, for the cell number 2, there are some improvements in minimizing cell load variation from cycle 100 to 300, and considerable improvement from cycle 300 to 500. For the cell numbers 3, 4, and 5, the curves are located very close to each other.

Comparison of results for minimum-sized problems, medium-sized problems, and bigger-sized problems from the literature [16] are mentioned in the following tables. Tables 2–4 show the results

that were obtained by the other methods in the literature using genetic algorithm (GA) [16], ant colony optimization (ACO) [21], results acquired by MABC, and MABC CPU time for intercellular move and cell load variations.

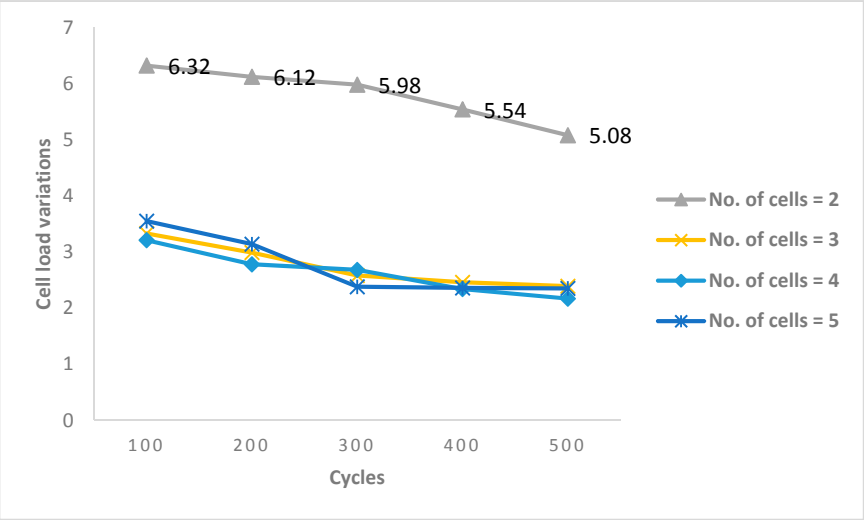

Figure 6. MABC convergence curve for cell load variation of problem size 10 × 15.

From Table 2, it has been inferred that there are 5 different matrix-sized problems and 21 cells. The cumulative intercellular moves provide better results in 16 cases (Serial number: 1, 2, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 18, 19, 20, 21 from Table 2) out of 21 cases (76%), than that of any of the other methods. It yields equal results in 5 cells (Sl. No.: 3, 8, 9, 16, 17) to the best one of other methods. The cumulative cell load variation provides better outcome in 11 cases (Sl. No.: 4, 5, 10, 11, 12, 13, 14, 15, 17, 19, 20 from Table 2) out of 21 cases (52%) than that of any of the other methods. It produces equal results in 9 cells (Sl. No.: 1, 2, 3, 6, 7, 8, 9, 16, 18) to the best of the other methods.

Table 2. Results comparison for minimum-sized problems.

Sl. No	Pbm.Size	No.of Cell	Intercellular Move			Cumulative Cell Load Variations			MABCCPU Time in Sec
			GA [16]	ACO [21]	MABC	GA [16]	ACO [21]	MABC	
1	8 × 20	2	9	9	7	5.63	5.63	5.63	0.593
2		3	17	17	15	3.35	3.35	3.35	0.516
3		4	26	26	26	2.13	2.13	2.13	0.578
4	10 × 12	2	7	7	5	9.09	9.09	7.53	0.344
5		3	13	19	9	8.24	3.46	3.46	0.454
6		4	20	20	15	2.48	2.48	2.48	0.905
7		5	22	22	21	3.24	3.24	3.24	0.686
8	10 × 15	2	0	0	0	5.08	5.08	5.08	0.343
9		3	7	7	7	2.39	2.39	2.39	0.453
10		4	17	13	15	2.18	2.17	2.17	1.129
11		5	26	24	25	3.14	2.35	2.35	0.676
12	11 × 22	2	13	8	7	7.27	6.22	6.22	0.406
13		3	18	17	15	4.30	5.11	4.92	0.468
14		4	29	25	25	2.83	3.81	3.52	1.895
15		5	45	26	36	2.33	3.76	2.55	1.609
16	14 × 24	2	0	0	0	11.37	11.37	11.37	0.593
17		3	5	0	0	10.67	8.91	8.91	0.625
18		4	3	3	2	6.51	6.51	6.51	1.765
19		5	10	9	9	5.94	6.33	4.59	0.954
20		6	17	14	12	4.41	4.30	4.3	1.327
21		7	22	22	18	3.89	3.89	4.14	23.625

Table 3. Result comparison for medium-sized problems.

Sl. No	Pbm. Size	No. of Cell	Intercellular Move			Cumulative Cell Load Variations			MABCCPU Time in Sec
			GA [16]	ACO [21]	MABC	GA [16]	ACO [21]	MABC	
1	Boctor 1	2	16	15	14	11.33	9.47	9.47	0.938
2		3	35	26	30	8.98	8.65	8.44	2.188
3		4	35	35	42	8.41	8.41	8.05	2.593
4		5	49	49	49	6.43	6.43	6.43	2.655
5		6	62	58	60	6.18	6.23	6.21	1.374
6		7	72	72	72	5.44	4.84	4.84	2.719
7		8	77	77	77	4.31	4.31	4.31	79.36
8	Boctor 2	2	4	4	3	15.45	15.45	15.45	0.876
9		3	9	9	8	14.30	12.46	12.46	1.031
10		4	30	14	13	12.78	10.63	11.01	1.093
11		5	33	23	23	8.56	9.40	9.4	1.372
12		6	40	38	37	9.18	7.92	8.6	1.312
13		7	54	52	50	6.75	7.02	7.93	2.563
14		8	62	61	62	6.28	6.42	6.28	78.782
15	Boctor 3	2	6	1	1	13.76	14.03	14.03	0.921
16		3	9	6	6	10.86	11.08	10.94	1.313
17		4	23	15	18	9.63	6.59	6.59	1.999
18		5	24	20	21	5.61	6.02	6.1	2.141
19		6	33	29	29	6.34	5.04	5.52	1.312
20		7	40	37	39	5.35	4.13	4.93	2.876
21		8	47	47	47	4.35	4.35	4.35	75.03
22	Boctor 4	2	13	13	13	15.88	15.88	15.75	0.937
23		3	25	25	16	14.51	14.51	14.38	1.328
24		4	53	36	37	10.66	12.59	11.28	1.486
25		5	68	50	50	10.75	10.42	10.39	1.656
26		6	68	70	72	8.65	8.37	8.87	1.374
27		7	82	76	95	6.96	7.53	7.81	2.859
28		8	88	86	86	6.15	6.50	6.5	75.422
29	Boctor 5	2	7	7	2	10.70	10.70	10.70	0.923
30		3	20	14	8	10.00	9.06	8.79	1.265
31		4	19	19	18	6.77	6.77	6.77	1.969
32		5	40	30	21	7.46	5.74	5.68	1.947
33		6	38	38	31	5.28	5.15	5.07	1.345
34		7	49	45	47	4.75	4.65	4.53	2.765
35		8	57	56	58	4.10	4.12	4.25	75.829

Table 3. *Cont.*

Sl. No	Pbm. Size	No. of Cell	Intercellular Move			Cumulative Cell Load Variations			MABCCPU Time in Sec
			GA [16]	ACO [21]	MABC	GA [16]	ACO [21]	MABC	
36	Boctor 6	2	8	2	2	12.18	11.50	11.55	0.906
37		3	10	4	5	9.64	9.20	9.24	1.25
38		4	20	12	12	8.93	6.51	6.56	1.656
39		5	17	17	18	5.94	5.94	5.98	1.656
40		6	34	27	31	5.16	5.52	4.96	1.297
41		7	44	39	45	4.74	5.53	4.69	2.765
42		8	55	55	55	4.59	4.59	4.59	119.50
43	Boctor 7	2	3	3	2	6.65	6.65	6.65	1.001
44		3	10	9	9	5.51	4.71	4.71	1.094
45		4	24	14	15	4.31	3.91	3.66	1.5
46		5	23	23	28	3.96	3.96	3.8	1.358
47		6	46	46	46	2.39	3.25	3.2	1.343
48		7	56	55	59	3.13	2.44	2.93	2.656
49		8	68	69	69	2.73	2.86	2.86	126.17
50	Boctor 8	2	23	12	11	14.26	12.95	12.95	1.001
51		3	22	22	20	10.87	10.87	10.87	1.249
52		4	33	28	32	10.09	9.95	10.08	1.485
53		5	39	38	39	9.17	8.87	9.17	1.485
54		6	54	49	49	7.55	8.09	8.09	1.391
55		7	63	65	65	7.06	6.81	6.76	2.828
56		8	74	74	74	5.41	5.51	5.51	119.33
57	Boctor 9	2	10	5	5	5.11	5.44	5.12	1.015
58		3	31	18	9	4.85	4.47	4.27	1.328
59		4	24	24	23	3.08	3.08	3.09	1.51
60		5	41	31	34	2.78	3.04	3.01	1.762
61		6	45	45	49	2.93	2.93	2.92	1.438
62		7	58	57	57	2.54	2.65	2.57	2.858
63		8	69	71	71	2.15	2.17	2.18	115.14
64	Boctor 10	2	13	6	5	4.83	4.08	4.08	0.891
65		3	18	18	18	3.85	3.30	3.19	1.297
66		4	29	23	28	3.38	2.66	2.66	1.623
67		5	37	33	41	2.75	2.57	2.45	1.909
68		6	46	41	49	2.28	2.39	2.25	2.347
69		7	58	58	58	2.03	2.08	2.08	2.765
70		8	65	65	65	1.85	1.85	1.85	78.829

Table 4. Result comparison for bigger-sized problems.

Sl. No	Pbm. Size	No. of Cell	Intercellular Move			Cumulative Cell Load Variations			MABCCPU Time in Sec
			GA [16]	ACO [21]	MABC	GA [16]	ACO [21]	MABC	
1	16 × 43	2	6	6	7	20.45	20.45	20.45	1.219
2		3	16	15	10	19.00	19.11	17.46	1.453
3		4	27	27	21	16.33	16.33	14.24	1.626
4		5	31	32	31	16.26	16.39	16.26	2.101
5		6	50	45	50	12.56	15.04	12.56	1.86
6		7	65	57	57	10.92	10.70	10.7	2.969
7		8	68	69	68	9.39	8.97	9.39	45.407
8	20 × 37	2	14	12	10	37.22	38.04	38.08	1.343
9		3	40	28	24	38.95	32.25	32.94	1.657
10		4	47	47	46	29.10	23.44	24.84	2.046
11		5	61	64	61	21.69	18.30	21.23	2.504
12		6	68	67	68	21.13	14.91	24.63	2.882
13		7	78	73	78	16.87	17.28	17.01	3.12
14		8	98	81	81	17.46	14.29	14.29	3.39
15		9	98	99	98	12.82	13.46	13.47	16.843
16		10	117	117	117	12.05	12.05	12.05	3015.17
17	20 × 55	2	41	30	30	59.25	61.35	60.28	1.751
18		3	74	48	50	51.35	57.81	57.50	2.281
19		4	101	93	85	47.72	47.22	47.19	3.281
20		5	114	110	100	43.46	45.47	43.64	3.507
21		6	133	130	135	40.51	36.27	37.92	3.784
22		7	149	150	150	38.20	34.81	34.81	3.944
23		8	175	164	170	31.71	31.98	29.82	4.344
24		9	198	180	198	25.39	27.96	25.39	12.797
25		10	213	203	200	24.72	25.27	25.13	393.483

Table 4. *Cont.*

Sl. No	Pbm. Size	No. of Cell	Intercellular Move			Cumulative Cell Load Variations			MABCCPU Time in Sec
			GA [16]	ACO [21]	MABC	GA [16]	ACO [21]	MABC	
26		2	18	0	26	57.76	58.53	56.44	2.031
27		3	51	12	26	53.66	51.87	51.38	2.064
28		4	48	30	32	51.26	49.05	45.98	2.751
29		5	78	53	52	33.80	37.71	36.36	3.889
30		6	102	63	54	33.17	35.44	39.96	3.964
31	25 × 50	7	78	93	81	32.40	22.68	29	4.021
32		8	114	99	107	28.95	21.44	25.03	4.245
33		9	113	100	100	25.66	26.07	26.07	5.741
34		10	128	132	125	20.13	17.50	20.77	9.361
35		11	131	128	129	20.69	18.73	19.28	63.24
36		12	138	136	138	18.24	16.41	18.24	3112.20
37		2	62	37	55	175.6	170.14	172.36	5.984
38		3	84	56	40	156.21	132.20	153.37	6.906
39		4	117	113	90	126.64	114.54	109.55	7.268
40		5	129	163	145	97.54	114.58	81.42	7.781
41		6	218	147	127	123.69	86.67	84.23	8.548
42		7	259	241	115	108.22	96.09	69.18	8.921
43		8	232	169	172	88.58	76.36	83.39	12.812
44		9	249	247	257	84.75	84.39	66.66	12.999
45		10	268	278	310	75.20	65.14	75.17	15.984
46	40 × 100	11	247	272	327	69.00	60.78	74.52	17.395
47		12	288	291	290	61.51	59.00	57.01	20.974
48		13	339	307	302	75.19	58.23	58.11	30.054
49		14	425	303	293	66.31	57.91	57.32	51.947
50		15	413	420	408	65.28	69.24	55.57	62.578
51		16	390	360	360	60.30	53.05	53.57	75.172
52		17	447	381	390	60.56	52.75	53.45	81.138
53		18	463	413	447	57.04	49.22	55.9	89.644
54		19		412	415		48.40	48.54	105.367
55		20		451	462		50.81	53.72	3568.459

From the analysis of Table 3, it has been known that there are 10 different matrix-sized problems and 70 cells are considered for results comparison. The cumulative intercellular moves provide better results in 40 cases out of 70 cases (57%) than that of any other method, and yield equal results in 18 cells compared to the best of the other methods. The cumulative cell load variation provides better outcome in 44 cases out of 70 cases (63%) than that of any of the method, and produces equal results in 18 cells to the best one of the other methods.

Table 4 indicates that there are 5 different matrix-sized problems and 55 cases. The cumulative intercellular moves provide better results in 41 cases out of 55 cases (75%) than that of any one of the other methods, and yields equal result in 7 cells compared to the best of the other methods. The cumulative cell load variation provides better outcome in 42 cases out of 55 cases (76%) than that of any one of the other methods, and produces equal results in 4 cells compared to the best of the other methods. Based on the analysis and results, the bar charts are plotted for clarity of results for the intercellular moves and cell load variations for the problem sizes 8 × 20, 10 × 12, and 10 × 15, to compare the results of MABC with GA and ACO.

Figure 7 indicates that in cell numbers 2 and 3, MABC intercellular moves are considered to be at a minimum, as there is a better convergence compared to GA and ACO. In cell number 4, MABC produces results that are on par with GA and ACO, and there is no appreciable difference observed when using MABC.

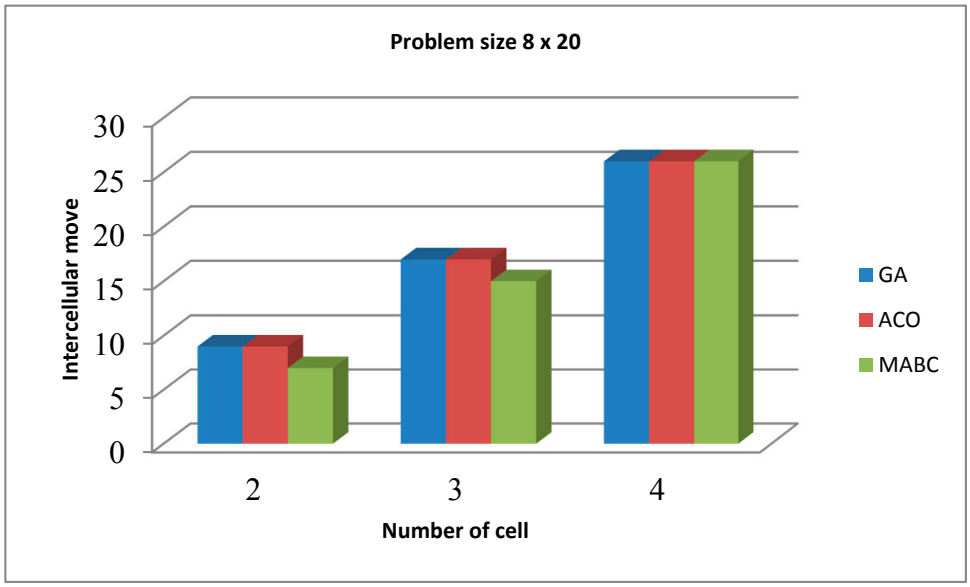

Figure 7. Results comparison of intercellular moves for the problem size 8 × 20.

Figure 8 indicates that in cell numbers 2, 3, and 4, MABC produces almost the same cell load variations as that of GA and ACO.

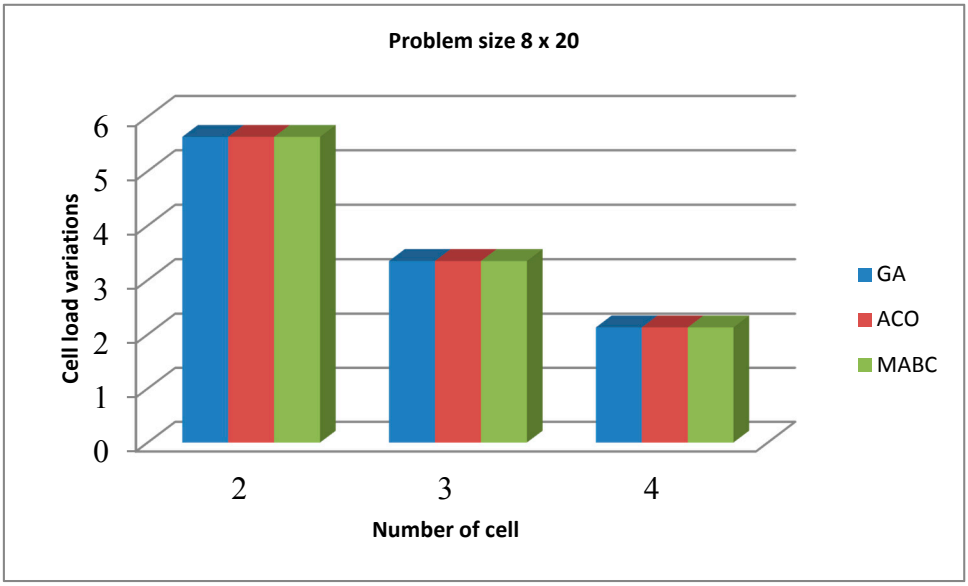

Figure 8. Results comparison of cell load variation for the problem size 8 × 20.

Figure 9 indicates that MABC intercellular moves are considered to be a minimum for cell number 2, 3, 4, and 5, as compared with GA and ACO.

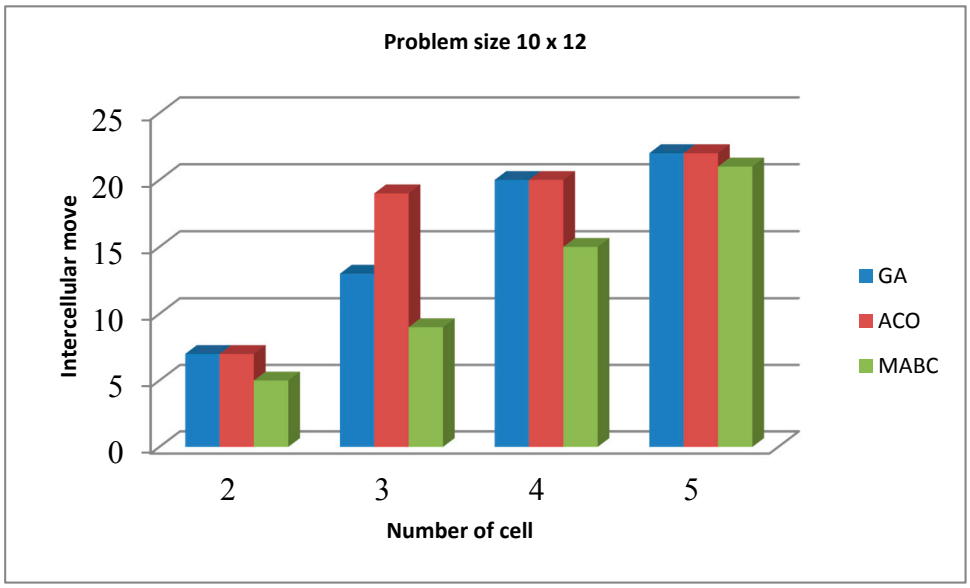

Figure 9. Results comparison of intercellular move for the problem size 10 × 12.

Figure 10 indicates that in cell number 2, MABC produces minimum cell load variation as compared with GA and ACO. In cell number 3, MABC exhibits minimum cell load variations as compared with GA, and produces the same result with ACO. For cell numbers 4 and 5, MABC yields the same cell load variations as that of GA and ACO.

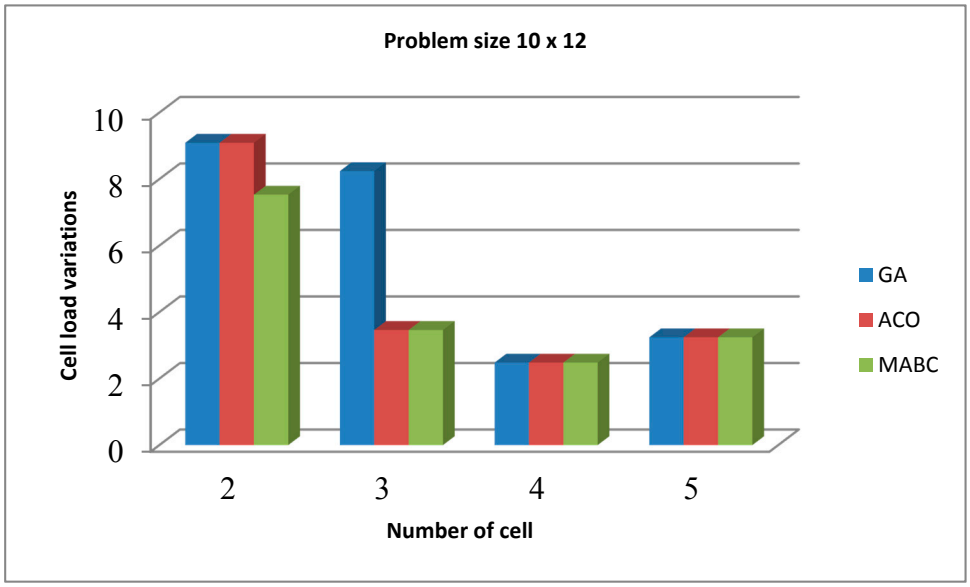

Figure 10. Results comparison of cell load variation for the problem size 10 × 12.

Figure 11 indicates that for cell numbers 2 and 3, MABC produces the same intercellular moves as that of GA and ACO. For cell numbers 4 and 5, MABC produces minimum intercellular moves as compared with GA.

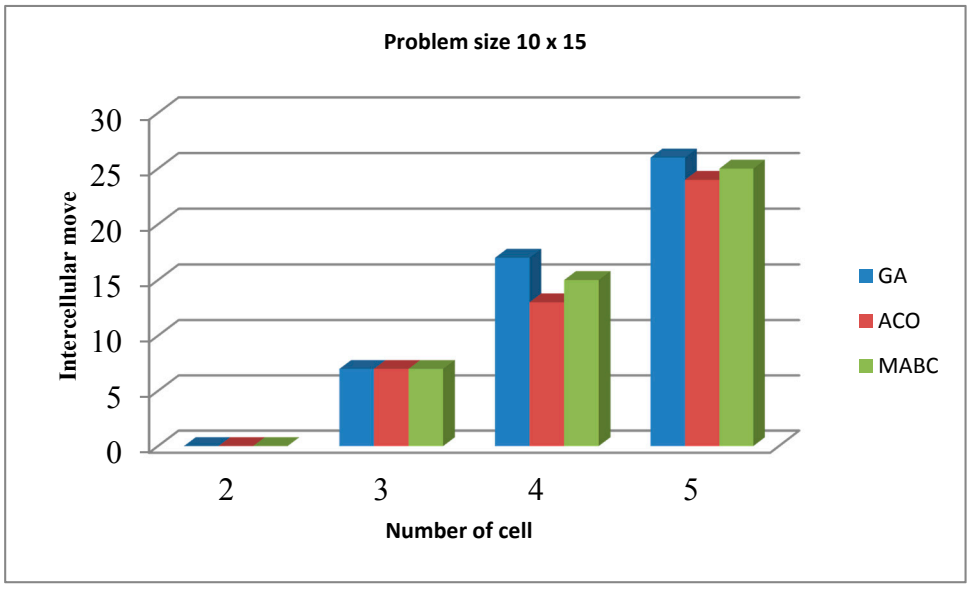

Figure 11. Results comparison of intercellular moves for the problem size 10 × 15.

Figure 12 indicates that in cell numbers 2 and 3, MABC produces same cell load variation as that of GA and ACO. In cell numbers 4 and 5, MABC produces minimum cell load variations as compared with GA, and the same result as that of ACO.

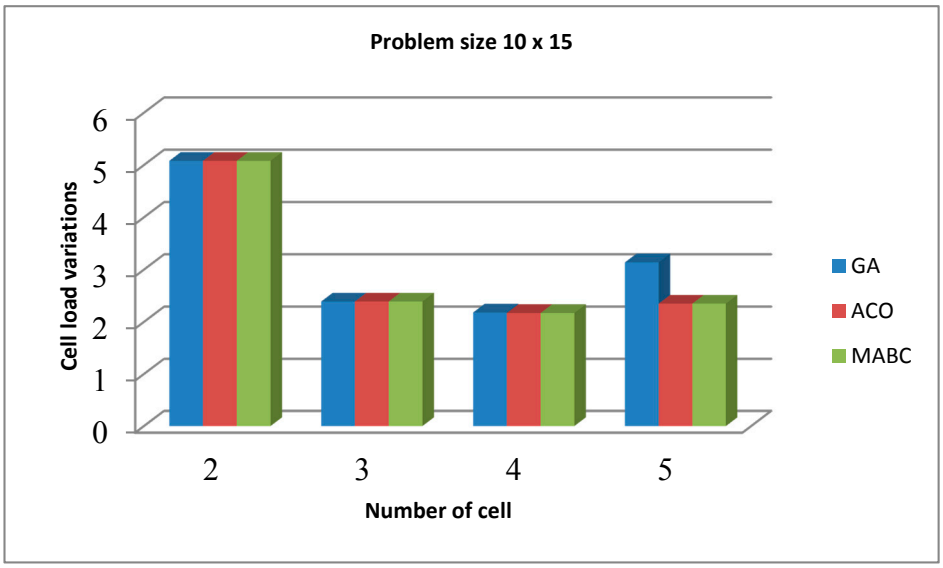

Figure 12. Results comparison of cell load variations for the problem size 10 × 15.

Figure 13 represents the MABC CPU time variation for cell 4 for the minimum-sized problems of sizes 8 × 20, 10 × 12, 10 × 15, and 11 × 22. The figure depicts that the increase of problem size (search space) leads to increase of CPU time.

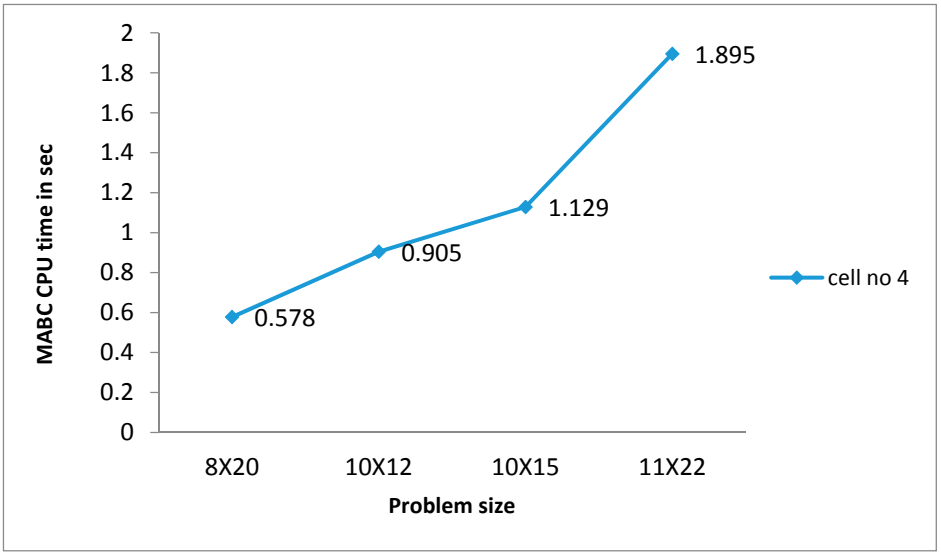

Figure 13. MABC CPU time variation for minimum-sized problems.

4. Conclusions

The main purpose of this paper is to compare the performance of the MABC approach with the other two heuristic algorithms in the literature GA [16] and ACO [21] to minimize the cumulative intercellular moves and the cumulative cell load variation. From the results obtained, it has been indicated that, in most of the problems discussed in the literature, cumulative intercellular moves and cumulative cell load variations obtained by the proposed method, MABC, is either better (minimized) than the other methods, or it is equal to the best one. The outcome obtained in the computational experiences carried out show that the proposed algorithm MABC deserves to be well placed among the three approaches in studies on machine cell formation. The work can be further extended in the future,

incorporating parameters like demand of parts, cost factor, etc., enhancing it to a more generalized sustainability in manufacturing environments.

Author Contributions: Adinarayanan Arunagiri developed the algorithm and wrote the manuscript. Uthayakumar Marimuthu analyzed the results. Prabhakaran Gopalakrishnan provided the various input datas to solve the problem. Adam Slota and Jerzy Zajac collected materials for literature review. Mahendra Prabhu Paulraj reviewed the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ham, I.; Hitomi, K. *Group Technology Applications to Production Management*; Kluwer-Nijhoff: Boston, MA, USA, 1985.
2. Black, J. Cellular manufacturing systems reduce setup time, make small lot production economical. *Ind. Eng.* **1983**, *15*, 36–48.
3. Wemmerlov, U.; Johnson, D.J. Cellular manufacturing at 46 user plants: Implementation and performance improvements. *Int. J. Prod. Res.* **1997**, *35*, 29–49. [CrossRef]
4. Kazerooni, M.L.; Luong, H.S.; Abhary, K. A genetic algorithm based cell design considering alternative routing. *Int. J. Comput. Integr. Manuf.* **1997**, *10*, 93–107.
5. Taboun, S.M.; Merchawi, N.S.; Ulger, T. Part family and machine cell formation in multi-period planning horizons of cellular manufacturing systems. *Prod. Plan. Control* **1998**, *9*, 561–571. [CrossRef]
6. Opitz, H. *A Classification System to Describe Work Pieces*; Pergamon: New York, NY, USA, 1970.
7. Burbidge, L. Production flow analysis. *Prod. Eng.* **1963**, *42*, 742–752. [CrossRef]
8. King, J.R. Machine-component grouping in production flow analysis: An approach using a rank order-clustering algorithm. *Int. J. Prod. Res.* **1980**, *18*, 213–232. [CrossRef]
9. Srinivasan, G.; Narendran, T.T.; Mahadevan, B. An assignment model for the part families in group technology. *Int. J. Prod. Res.* **1990**, *29*, 463–478. [CrossRef]
10. Mukattash, A.M.; Adil, M.B.; Tahboub, K.K. Heuristic approaches for part assignment in cell formation. *Comput. Ind. Eng.* **2002**, *42*, 329–341. [CrossRef]
11. Kim, C.O.; Baek, J.G.; Baek, J.K. A two-phase heuristic algorithm for cell formation problems considering alternative part routes and machine sequences. *Int. J. Prod. Res.* **2004**, *42*, 3911–3927.
12. Chen, W.H.; Srivastava, B. Simulated annealing procedures for forming machine cell in-group technology. *Eur. J. Oper. Res.* **1994**, *71*, 100–111. [CrossRef]
13. Venugopal, V.; Narendran, T.T. Cell formation in manufacturing systems through simulated annealing: An experimental evaluation. *Eur. J. Oper. Res.* **1992**, *63*, 409–422. [CrossRef]
14. Tavakkoli-Moghaddam, R.; Aryanezhad, M.B.; Safaei, N.; Azaron, A. Solving a dynamic cell formation problem using metaheuristics. *Appl. Math. Comput.* **2005**, *170*, 761–780. [CrossRef]
15. Venugopal, V.; Narendran, T.T. A genetic algorithm approach to the machine-component grouping problems with multiple objectives. *Comput. Ind. Eng.* **1992**, *22*, 469–480. [CrossRef]
16. Prabhakaran, G. Clustering and Machine Cell Formation for Cellular Manufacturing Systems. Ph.D. Thesis, Bharathidasan University, Tiruchirappalli, Tamil Nadu, India, 2001.
17. Sarac, T.; Ozcelik, F. A genetic algorithm with proper parameters for manufacturing cell formation problems. *J. Intell. Manuf.* **2012**, *23*, 1047–1061. [CrossRef]
18. Arkat, J.; Hosseini, L.; Farahani, M.H. Minimization of exceptional elements and voids in the cell formation problem using a multi-objective genetic algorithm. *Expert Syst. Appl.* **2011**, *38*, 9597–9602. [CrossRef]
19. Vakharia, A.J.; Chang, Y.L. Cell formation in group technology: A combinatorial search approach. *Int. J. Prod. Res.* **1997**, *35*, 2025–2043. [CrossRef]
20. Chung, S.H.; Wu, T.H.; Chang, C.C. An efficient tabu search algorithm to the cell formation problem with alternative routings and machine reliability considerations. *Comput. Ind. Eng.* **2011**, *60*, 7–15. [CrossRef]
21. Prabhakaran, G.; Muruganandam, A.; Asokan, P.; Girish, B.S. Machine cell formation for cellular manufacturing systems using an ant colony system approach. *Int. J. Adv. Manuf. Technol.* **2005**, *25*, 1013–1019. [CrossRef]
22. Spiliopoulos, K.; Sofianopoulou, S. An efficient ant colony optimization system for the manufacturing cells formation system. *Int. J. Adv. Manuf. Technol.* **2008**, *36*, 589–597. [CrossRef]

23. Andres, C.; Lozano, S. A particle swarm optimization algorithm for part machine grouping. *Robot. Comput. Integr. Manuf.* **2006**, *22*, 468–474. [CrossRef]
24. Duran, O.; Rodriguez, N.; Consalter, L.A. A PSO-based clustering algorithm for manufacturing cell design. In Proceedings of the 2008 Workshop on Knowledge Discovery and Data Mining, Adelaide, Australia, 23–24 January 2008. [CrossRef]
25. Bajestani, A.M.; Rabbani, M.; RahimiVahed, A.R.; Khoshkhou, G.B. A multi objective scatter search for a dynamic cell formation problem. *Comput. Oper. Res.* **2009**, *36*, 777–794. [CrossRef]
26. Chattopadhyay, M.; Chattopadhyay, S.; Dan, P.K. Machine-part cell formation through visual decipherable clustering of self-organizing map. *Int. J. Adv. Manuf. Technol.* **2011**, *52*, 1019–1030. [CrossRef]
27. Ghezavati, V.R.; SaidiMehrabad, M. An efficient hybrid self-learning method for stochastic cellular manufacturing problem: A queuing-based analysis. *Expert Syst. Appl.* **2011**, *38*, 1326–1335. [CrossRef]
28. Ying Chin, H.; Ta-Wei, L. A concurrent solution for intra-cell flow path layouts and I/O point locations of cells in a cellular manufacturing system. *Comput. Ind. Eng.* **2011**, *60*, 614–634.
29. Venkumar, P.; Haq, A.N. Complete and fractional cell formation using Kohonen self-organizing map networks in cellular manufacturing system. *Int. J. Prod. Res.* **2006**, *20*, 4257–4271. [CrossRef]
30. Arkat, J.; Abdollahzadeh, H.; Ghahve, H.A. New branch and bound algorithm for cell formation problem. *Appl. Math. Model.* **2012**, *36*, 5091–5100. [CrossRef]
31. Sayadi, M.K.; Hafezalkotob, A.; Naini, S.G.J. Firefly-inspired algorithm for discrete optimization problems: An application to manufacturing cell formation. *J. Manuf. Syst.* **2013**, *32*, 78–84. [CrossRef]
32. Mohammad, M.; Kamran, F. Designing cellular manufacturing systems considering S-shaped layout. *Comput. Ind. Eng.* **2016**, *98*, 221–236. [CrossRef]
33. Behrang, B.; Iraj, M.; Mohammad, M.P. New criteria for configuration of cellular manufacturing considering product mix variation. *Comput. Ind. Eng.* **2016**, *98*, 413–426.
34. Sakhaii, M.; Tavakkoli-Moghaddam, R.; Bagheri, M.; Vatani, B. A robust optimization approach for an integrated dynamic cellular manufacturing system and production planning with unreliable machines. *Appl. Math. Model.* **2016**, *40*, 169–191. [CrossRef]
35. Brown, J.R. A capacity constrained mathematical programming model for cellular manufacturing with exceptional elements. *J. Manuf. Syst.* **2015**, *37*, 227–232. [CrossRef]
36. Kia, R.; KhaksarHaghani, F.; Javadian, N.; Tavakkoli-Moghaddam, R. Solving a multi-floor layout design model of a dynamic cellular manufacturing system by an efficient genetic algorithm. *J. Manuf. Syst.* **2014**, *33*, 218–232. [CrossRef]
37. Shih, Y.C.; Filho, E.V.G. A design procedure for improving the effectiveness of fractal layouts formation. *Artif. Intell. Eng. Des. Anal. Manuf.* **2014**, *28*, 1–26. [CrossRef]
38. Pooranian, Z.; Shojafar, M.; Abawajy, J.H.; Abraham, A. An efficient meta-heuristic algorithm for grid computing. *J. Comb. Optim.* **2015**, *30*, 413–434. [CrossRef]
39. Mohammad, S.; Saeed, J.; Saeid, A.; Nicola, C. FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. *Clust. Comput.* **2015**, *18*, 829–844.
40. Mohammad, S.; Saeed, J.; Shahdad, S.; Sima, S. FR trust: A fuzzy reputationbased model for trust management in semantic P2P grids. *Int. J. Grid Util. Comput.* **2015**, *6*, 57–66.
41. Javanmardi, S.; Shojafar, M.; Amendola, D.; Cordeschi, N.; Liu, H.; Abraham, A. Hybrid Job Scheduling Algorithm for Cloud Computing Environment. In Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA), Ostrava, Czech Republic, 23–25 June 2014; pp. 43–52.
42. Logendran, R. A workload based model for minimizing total intercell and intracell moves in cellular manufacturing. *Int. J. Prod. Res.* **1990**, *28*, 913–925. [CrossRef]
43. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization*; Technical Report TR06; Computer Engineering Department, Erciyes University: Kayseri, Turkiye, 2005.
44. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
45. Karaboga, D.; Basturk, B. On the performance of artificial bee colony (ABC) algorithm. *Appl. Soft Comput.* **2008**, *8*, 687–697. [CrossRef]

46. Karaboga, D.; Akay, B. A comparative study of artificial bee colony algorithm. *Appl. Math. Comput.* **2009**, *214*, 108–132. [CrossRef]
47. Kalayci, C.B.; Gupta, S.M.; Nakashima, K. Bees colony intelligence in solving disassembly line balancing problem. In Proceedings of the 2011 Asian Conference of Management Science and Applications, Sanya, China, 21–23 December 2011; pp. 34–41.