Optim Lett (2018) 12:335–346
https://doi.org/10.1007/s11590-017-1200-3

ORIGINAL PAPER

A matheuristic for the cell formation problem

Rian Gabriel S. Pinheiro[1,2] · **Ivan C. Martins**[2] · **Fábio Protti**[2] · **Luiz Satoru Ochi**[2]

Received: 29 November 2015 / Accepted: 13 September 2017 / Published online: 16 September 2017

Abstract In this paper we propose a GRASP matheuristic coupled with an Integer Programming refinement based on Set Partitioning to solve the Cell Formation Problem. We use the grouping efficacy measure to evaluate the solutions. As this measure is nonlinear, we propose a fractional Set Partitioning approach and its linearization. Our method is validated on a set of 35 instances from the literature. The experiments found four unknown solutions. For all instances with known optima, our method is able to determine the optimum solutions.

Keywords Combinatorial optimization · Matheuristics · Cellular manufacturing · Group technology · Hybrid heuristics

1 Introduction

The CELL FORMATION PROBLEM (CFP) consists of grouping machines into cells, dedicated to process a set of parts, so that machine operations on parts lying in different cells are minimized while operations on parts in a same cell are maximized. In the form of a 0–1 part-machine incidence matrix, the CFP can be viewed as a block diagonalization problem. The cells can be formed by a rearrangement (permutation) of rows and columns of the matrix to form part-machine blocks (submatrices) along

✉ Fábio Protti
fabio@ic.uff.br

Rian Gabriel S. Pinheiro
rian.gabriel@gmail.com

1 Federal Rural University of Pernambuco, Garanhuns, PE, Brazil

2 Instituto de Computação, Universidade Federal Fluminense, Rua Passo da Pátria 156, Bloco E - 3ºandar, São Domingos, Niterói, RJ 24210-240, Brazil

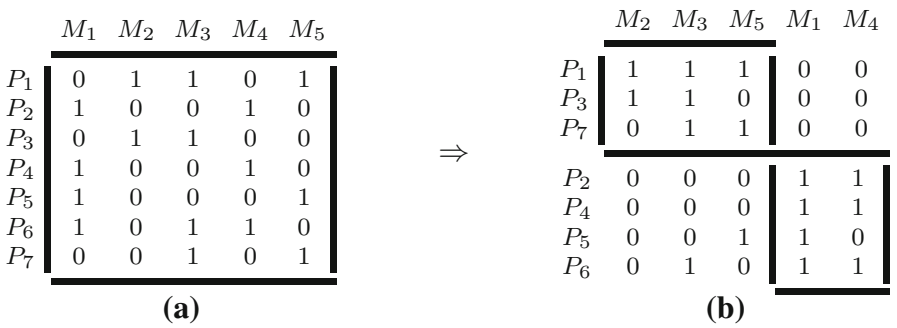

Fig. 1 The cell formation problem: representation and solution. **a** Matricial representation. **b** Example of a solution with two cells

the diagonal of the main matrix. The objective is then to find a rearrangement that minimizes the number of 0's inside and 1's outside the submatrices, as shown in Fig. 1.

Due to the NP-hard nature of the CFP [6], several heuristic methods have been developed over the last decades. A reference survey on these methods is presented by Ghosh et al. [9], and hybrid metaheuristics have been reported by several researchers [5, 7,13,16,21].

The term *matheuristics* describes the integration between mathematical programming and metaheuristics [4,12]. In this paper we develop a matheuristic which combines an exact procedure based on a fractional formulation for SET PARTITIONING (SP) [1,2] with a GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE (GRASP) heuristic. The idea is to store a pool of solutions generated during the heuristic execution and then solve a SP problem in order to extract the best combination of solutions.

Within the context of matheuristics, the SP approach has been successfully employed for solving several combinatorial optimization problems, e.g. the Vehicle Routing Problem [20] and the Cluster Editing Problem [3]. However, to the best of our knowledge, it has not been applied yet to solve the CFP. Indeed, this is the first matheuristic specifically developed to the CFP.

In order to evaluate our method, computational experiments have been performed on 35 widely used benchmark instances with up to 40 machines and 100 parts. The experiments found four unknown solutions. In addition, for all instances with known optima, our method is able to determine the optimum solutions.

1.1 The cell formation problem

The CFP aims to form cells in which grouped machines are used for the production of a family of parts. Each family must preferably have a high degree of similarity, that is, it is desirable that a large number of machines can perform common operations on these parts. An instance of the CFP can be represented by a binary part-machine matrix $A = (a_{ij})_{|P|\times|M|}$, where P is the set of parts and M the set of machines. An entry a_{ij} with value 1 indicates that an operation is performed on part $i \in P$ by machine

$j \in M$; otherwise, $a_{ij} = 0$. Figure 1a shows an example of the matrix representation of a manufacturing system, with $|P| = 7$ and $|M| = 5$, and Fig. 1b an example of solution where the machine groups $\{M_2, M_3, M_5\}$ and $\{M_1, M_4\}$ are associated with the families $\{P_1, P_3, P_7\}$ and $\{P_2, P_4, P_5, P_6\}$, respectively. A cell is precisely an association of a machine group with a family of parts. Note that the number of cells is not fixed.

An ideal cell formation is obtained when each machine operates on all the parts assigned to its cell, and does not operate on parts in other cells. In this scenario, parts do not need to be moved from one cell to another because no other machine is needed to operate on it. However, a solution with this pattern is very difficult to be found in practice.

1.2 Solution quality

Different measures have been proposed to evaluate the quality of a solution of the CFP. Sarker and Khan [19] made a critical review and a comparative study among several possible metrics. The grouping efficacy measure [11] is widely used in the literature to evaluate the quality of solutions. This measure is defined as follows:

$$\mu = \frac{e_1 - e_1^{out}}{e_1 + e_0^{in}} \tag{1}$$

where e_1 is the number of entries with value 1, e_1^{out} the number of entries with value 1 that appear outside cells (called *exceptions*), and e_0^{in} the number of entries with value 0 that appear inside cells (called *voids*). Note that $\mu = 1$ implies a perfect cell formation.

1.3 Constraints

In the design of manufacturing cell formation, several production characteristics can be considered at the same time, e.g., cost/operation time, capacity of the cells or parts, machine capacity, demand for a product, and machine proximity constraints. Martins et al. [13] made a review on the number of machines that must comprise the cells (the most common constraint). The term *singleton* is used for a cell that contains fewer than two machines or parts [10]. There are some approaches that do not allow the presence of singletons in a solution. However, in the literature of the CFP, there is no consensus about the minimum cell size. In this paper, as in most works, the presence of singletons in a solution is allowed.

2 The GRASP-SP approach

In this section we describe the proposed algorithm, based on the GRASP metaheuristic [8] coupled with an exact method based on SP and a VARIABLE NEIGHBORHOOD SEARCH (VND) procedure that uses a random neighborhood ordering in the local search phase.

To implement the GRASP-SP method, summarized in Algorithm 1, three main methods are specified: (1) CONSTRUCT, that constructs an initial solution (see Sect. 2.1); (2) VND, that performs the local search procedure using three neighborhoods proposed by [13] (see Sect. 2.2); (3) SP- SOLVER, that executes a refinement method based on SP (see Sect. 2.3).

Algorithm 1 GRASP-SP

```
1: procedure GRASP- SP(input matrix A)
2:     while (stop condition) do
3:         sol = CONSTRUCT()
4:         sol = VND(sol)
5:         put the cells in the Pool
6:         if sol > best then
7:             best = sol
8:         end if
9:     end while
10:    SP-model = createModel(Pool)
11:    sol = SP- SOLVER(SP-model)
12:    sol = VND(sol)
13:    if sol > best then
14:        best = sol
15:    end if
16:    return best
17: end procedure
```

2.1 Constructive phase

In order to generate initial solutions, the algorithm randomly creates machine groups. Each group will form a new cell. Next, each part is greedily associated with one of the machine groups and is included in the corresponding cell. The greedy choice is simply based on the grouping efficacy measure resulting from an association.

2.2 Local search

Local search algorithms are based on the iterative exploration of the solution space. At each iteration, the algorithm moves from the current solution to a neighbor solution. A neighborhood of a solution S is a subset of solutions denoted by $N(S)$. Each solution $S' \in N(S)$ is obtained from S by applying a *movement*. Approaches such as the VND, proposed by [14], make use of multiple neighborhood structures (see Algorithm 2). Let $V = \{V^1, \ldots, V^r\}$ be a set of neighborhood structures. Whenever a given neighborhood fails to improve the current solution, the VND takes the next neighborhood (line 11).

Here we use a variant of the VND that employs a random ordering of the neighborhoods in the local search phase. In Algorithm 2, the visiting order is prearranged randomly (line 2), and a subsequent call to this procedure will possibly try another visiting order. This variant has been successfully employed in Martins et al. [13], Penna et

al. [17], Subramanian et al. [20]. The effectiveness of a local search procedure depends on several aspects, such as the choice of neighborhood structures. In this work we use three neighborhoods to implement the VND: Regroup, Machine Shift, and Machine Swap.

Algorithm 2 VND

```
1: procedure VND(A)
2:     randomly initialize V
3:     k ← 1
4:     μ* ← μ(A)
5:     repeat
6:         A' ← V^k(A)
7:         if μ(A') > μ* then
8:             A* ← A'
9:             k ← 1
10:        else
11:            k ← k + 1
12:        end if
13:    until k = k_max
14:    return A*
15: end procedure
```

Regroup:	Similar to CONSTRUCT, but executes the inverse process, grouping parts into cells. Next, each machine is associated with one of the cells.
Machine Shift:	Moves a machine from one cell to another.
Machine Swap:	Swaps two machines in different cells.

2.3 A refinement method based on set partitioning

In this section, we propose a refinement method based on SP. Metaheuristic coupled with SP has been successfully applied to solve other problems [3,20]. First, consider all the feasible solutions $A_1, A_2, \ldots, A_k$ obtained by iterations of the GRASP heuristic in order to obtain A^*. Assume that A_j is partitioned into blocks (cells) $B_j^1, \ldots, B_j^{n_j}$, and consider the collection $\mathcal{B}$ of all blocks generated in the first phase of the GRASP, that is, $\mathcal{B} = \cup_{j=1}^{k} \{B_j^1, \ldots, B_j^{n_j}\}$. We remark that $\mathcal{B}$ is not considered as a multiset (possible duplicates of a block B_j^h are discarded). Rewrite $\mathcal{B} = \{B_1, B_2, \ldots, B_\ell\}$. Note that each subset $B_j \in \mathcal{B}$ is a candidate to be a block in an optimal solution.

The idea is to select some suitable blocks from the pool of blocks $\mathcal{B}$ such that the selected blocks form a partition of $M \cup P$. Let $\mathcal{B}_i \subseteq \mathcal{B}$ be the subset of blocks that contain the machine/part $i \in M \cup P$. Define y_j as the binary variable associated with a block $B_j \in \mathcal{B}$. Let a_j be the number of voids in block B_j, and r_j the number of exceptions occurring in entries of A whose rows (parts) belong to B_j.

We solve the following SP formulation for the CFP with input A:

$$\max \quad \sum_{i \in M \cup P} \frac{e_1 - \sum\limits_{i \in M \cup P} r_i y_i}{e_1 + \sum\limits_{i \in M \cup P} a_i y_i} \tag{2}$$

$$\text{s.t.} \quad \sum_{B_j \in \mathcal{B}_i} y_j = 1 \qquad \forall i \in M \cup P \tag{3}$$

$$y_j \in \{0, 1\} \qquad 1 \leq j \leq \ell \tag{4}$$

The objective function (2) maximizes the grouping efficacy as in Eq. (1). The SP constraints (3) state that every machine or part $i \in M \cup P$ must be exactly in a single block from the subset $\mathcal{B}_i$, and (4) define the domain of the decision variables.

Note that this is a linear-fractional formulation. We now describe a new linear formulation using x_{ra} as the binary variable associated with the fact that the final solution has $e_1^{out} = r$ and $e_0^{in} = a$. Let $c(r, a) = \frac{e_1 - r}{e_1 + a}$, and let q be the number of 1's in input matrix A. The linear formulation is as follows:

$$\max \quad \sum_{a \leq |P||M| - q} \sum_{r \leq q} c(r, a)\, x_{ra} \tag{5}$$

$$\text{s.t.} \quad \sum_{B_j \in \mathcal{B}_i} y_j = 1 \qquad \forall i \in M \cup P \tag{6}$$

$$\sum_{a \leq |P||M| - q} \sum_{r \leq q} x_{ra} = 1 \tag{7}$$

$$\sum_{a \leq |P||M| - q} \sum_{r \leq q} r\, x_{ra} = \sum_{B_j \in \mathcal{B}_i} r_j y_j \qquad \forall i \in M \cup P \tag{8}$$

$$\sum_{a \leq |P||M| - q} \sum_{r \leq q} a\, x_{ra} = \sum_{B_j \in \mathcal{B}_i} a_j y_j \qquad \forall i \in M \cup P \tag{9}$$

$$x_{ra} \in \{0, 1\} \qquad \forall\, a \leq |P||M| - q\,, r \leq q \tag{10}$$

$$y_j \in \{0, 1\} \qquad 1 \leq j \leq \ell \tag{11}$$

The objective function (5) minimizes the sum of the costs by choosing the best combination of blocks. There are no changes in SP constraints (6). Due to the fractional objective function, we add constraints (7)-(9) in order to associate the costs r_j and a_j with the variable x_{ra}. Finally, constraints (10) and (11) define the domain of the decision variables.

3 Computational results

In this section we present the results of our computational experiments. To evaluate the quality of the solutions, the performance measure used is the grouping efficacy, as explained in Sect. 1.1. In Tables 1, 2 and 3, the grouping efficacy of a solution is multiplied by 100 in order to present the results in percentage terms. Computational times are presented in seconds. All the algorithms were run on an Intel Core i7-4790 processor with 3.60 GHz and 16 GB of RAM, using OS Linux 4.2.0.

3.1 Instances

Our tests were performed on 35 well known instances from the literature that can be found in the appendix of the paper by [10]. In each row of Table 1, the first column contains the name of the instance, the second shows the size $|P| \times |M|$ of the instance (represented by input matrix $A_{|P|\times|M|}$), and the third reports the best grouping efficacy in the literature together with the corresponding reference. All the rows referenced by the work of [18] contain optimal solution values. In each of the remaining rows, the best value found in the literature is always obtained via one of the following heuristic approaches:

SCM-BMCF:	*Similarity Coefficients Method with Boltzmann Function and Mutation Operator for Cell Formation* [21]
SA:	*Simulated Annealing* [15]
HMGA:	*Hybrid Method Genetic Algorithm* [7]

3.2 Literature comparison

In this section we compare our methods with the existing ones in the literature.

Our GRASP-SP algorithm is run ten times for each instance in Table 1 with a fixed number of iterations (based on instance size), and stops after $k \times |M|$ iterations with no solution improvement. After some preliminary tests, we defined $k = 2$. We marked with "$*$" the instances that are solved to optimality.

Table 2 shows, for each instance: the best heuristic value in the literature; the value found by the exact approach developed by [18]. The GRASP and GRASP-SP approaches are compared using the following criteria: the best solution obtained among all the 10 runs, the gap and the average computational time. The corresponding gaps are computed as follows:

$$\%gap = 100\,\frac{\mu(s^*) - \mu(s)}{\mu(s^*)}, \tag{12}$$

where $\mu(s)$ is the grouping efficacy of solution s (obtained by GRASP or GRASP-SP) and s^* is the best solution in the literature.

We observe that our GRASP-SP method finds the best solution values, for all the instances. In particular, for CFP16 and CFP30, it was able to find optimal solutions

Table 1 Instances of the CFP

Instance	Name of instance	Size	Best efficacy in the literature
CFP01*	King1982	07 × 05	75.00 [18]
CFP02*	Waghodekar1984	07 × 05	69.57 [18]
CFP03*	Seiffodini1989	18 × 05	80.85 [18]
CFP04*	Kusiak1992	08 × 06	79.17 [18]
CFP05*	Kusiak1987	11 × 07	60.87 [18]
CFP06*	Boctor1991	11 × 07	70.83 [18]
CFP07*	Seiffodini1986	12 × 08	69.44 [18]
CFP08*	Chandrasekaran1986a	20 × 08	85.25 [18]
CFP09*	Chandrasekaran1986b	20 × 08	58.72 [18]
CFP10*	Mosier1985a	10 × 10	75.00 [18]
CFP11*	Chan1982	15 × 10	92.00 [18]
CFP12*	Askin1987	23 × 14	74.24 [18]
CFP13*	Stanfel1985	24 × 14	72.86 [18]
CFP14*	McCornick1972a	24 × 16	53.33 [18]
CFP15*	Srinivasan1990	30 × 16	69.92 [18]
CFP16*	King1980	43 × 16	58.04 [18]
CFP17*	Carrie1973a	24 × 18	57.73 [18]
CFP18	Mosier1985b	20 × 20	43.97 [15]
CFP19*	Kumar1986	23 × 20	50.81 [18]
CFP20*	Carrie1973b	35 × 20	79.38 [18]
CFP21*	Boe1991	35 × 20	58.79 [18]
CFP22*	Chandrasekharan1989_1	40 × 24	100.00 [18]
CFP23*	Chandrasekharan1989_2	40 × 24	85.11 [18]
CFP24*	Chandrasekharan1989_3-4	40 × 24	73.51 [18]
CFP25*	Chandrasekharan1989_5	40 × 24	53.29 [21]
CFP26	Chandrasekharan1989_6	40 × 24	48.95 [7]
CFP27	Chandrasekharan1989_7	40 × 24	46.58 [7]
CFP28	McCormick1972b	27 × 27	54.82 [15]
CFP29	Carrie1973c	46 × 28	47.68 [15]
CFP30*	Kumar1987	51 × 30	63.04 [18]
CFP31*	Stanfel1985_1	50 × 30	59.77 [18]
CFP32	Stanfel1985_2	50 × 30	50.83 [21]
CFP33	King1982	90 × 30	47.93 [15]
CFP34	McCormick1972c	53 × 37	61.16 [15]
CFP35*	Chandrasekharan1987	100 × 40	84.03 [18]

(note that other heuristic approaches were not able to find such solutions). Also, for CFP29 and CFP33, it was able to improve the best solutions in the literature (such instances still have unknown optima).

Table 2 Results of GRASP and GRASP-SP

Instance	Lit.		GRASP			GRASP-SP		
	Heuristic	Exact	Best	Gap (%)	time (s)	Best	Gap (%)	time (s)
CFP01	75	75	75	0.0	0.01	75	0.0	0.01
CFP02	69.57	69.57	69.57	0.0	0.01	69.57	0.0	0.01
CFP03	80.85	80.85	80.85	0.0	0.02	80.85	0.0	0.02
CFP04	79.17	79.17	79.17	0.0	0.01	79.17	0.0	0.01
CFP05	60.87	60.87	60.87	0.0	0.01	60.87	0.0	0.01
CFP06	70.83	70.83	70.83	0.0	0.01	70.83	0.0	0.01
CFP07	69.44	69.44	69.44	0.0	0.02	69.44	0.0	0.02
CFP08	85.25	85.25	85.25	0.0	0.03	85.25	0.0	0.04
CFP09	58.72	58.72	58.72	0.0	0.29	58.72	0.0	0.32
CFP10	75	75	75	0.0	0.03	75	0.0	0.03
CFP11	92	92	92	0.0	0.03	92	0.0	0.03
CFP12	74.24	74.24	74.24	0.0	0.30	74.24	0.0	0.33
CFP13	72.86	72.86	72.86	0.0	0.33	72.86	0.0	0.37
CFP14	53.33	53.33	53.33	0.0	0.71	53.33	0.0	0.78
CFP15	69.92	69.92	69.92	0.0	0.81	69.92	0.0	0.89
CFP16	57.96	58.04	**58.04**	0.0	1.94	**58.04**	0.0	2.14
CFP17	57.73	57.73	57.73	0.0	1.22	57.73	0.0	1.34
CFP18	43.97		43.97	0.0	2.54	43.97	0.0	2.79
CFP19	50.81	50.81	50.81	0.0	2.25	50.81	0.0	2.48
CFP20	79.38	79.38	79.38	0.0	4.01	79.38	0.0	4.41
CFP21	58.79	58.79	58.79	0.0	2.53	58.79	0.0	2.78
CFP22	100	100	100	0.0	4.81	100	0.0	5.29
CFP23	85.11	85.11	85.11	0.0	6.43	85.11	0.0	7.07
CFP24	73.51	73.51	73.51	0.0	7.01	73.51	0.0	7.71
CFP25	53.29	53.29	53.29	0.0	9.37	53.29	0.0	10.31
CFP26	48.95		48.95	0.0	9.97	48.95	0.0	10.97
CFP27	46.58		46.26	0.7	10.55	46.58	0.0	11.61
CFP28	54.82		54.82	0.0	19.11	54.82	0.0	21.02
CFP29	47.68		47.83	−0.3	27.57	**47.84**	−0.3	30.33
CFP30	62.94	63.04	**63.04**	0.0	27.17	**63.04**	0.0	29.89
CFP31	59.77	59.77	59.77	0.0	35.37	59.77	0.0	38.91
CFP32	50.83		50.83	0.0	38.53	50.83	0.0	42.38
CFP33	47.93		48.15	−0.5	73.27	**48.29**	−0.8	80.60
CFP34	61.16		61.16	0.0	83.55	61.16	0.0	91.91
CFP35	84.03	84.03	84.03	0.0	248.09	84.03	0.0	272.90

Table 3 Results of GRASP and GRASP-SP for 30 random instances

Instance	GRASP			GRASP-SP		
	Best	Average	Time (s)	Best	Average	Time (s)
RAND01	48.03	47.95	5.06	**48.43**	47.97	10.56
RAND02	48.33	48.33	6.17	48.33	48.33	11.95
RAND03	47.31	47.31	6.25	47.31	47.31	11.30
RAND04	46.26	45.69	4.40	46.26	45.71	14.15
RAND05	48.11	48.11	0.80	**48.24**	48.12	10.50
RAND06	47.51	47.46	5.18	47.51	47.46	10.08
RAND07	48.35	48.16	2.46	48.35	48.16	12.16
RAND08	50.85	50.63	8.24	50.85	50.65	8.44
RAND09	46.85	46.37	8.26	46.86	46.37	12.55
RAND10	45.49	45.44	4.40	**45.58**	45.48	11.05
RAND11	48.92	48.79	7.31	48.92	48.79	12.16
RAND12	48.30	48.09	8.80	48.30	48.09	12.51
RAND13	50.00	49.31	9.43	50.00	49.36	10.37
RAND14	47.48	47.48	3.65	47.48	47.48	8.55
RAND15	47.36	47.35	6.41	**47.98**	47.38	11.71
RAND16	50.92	50.92	4.55	50.92	50.92	12.37
RAND17	47.68	47.26	5.73	**48.30**	47.35	12.15
RAND18	50.39	50.33	2.17	50.39	50.33	8.37
RAND19	51.09	51.08	19.02	51.09	51.08	24.79
RAND20	49.68	49.68	8.38	**50.00**	49.85	10.32
RAND21	48.30	47.69	1.42	48.30	47.69	11.25
RAND22	46.01	45.91	4.89	46.01	45.91	14.05
RAND23	50.62	50.36	7.93	50.62	50.36	9.62
RAND24	45.40	45.26	4.98	45.40	45.26	14.80
RAND25	48.56	48.02	5.63	**48.64**	48.18	11.51
RAND26	49.84	49.83	6.20	49.84	49.83	11.07
RAND27	48.50	48.10	7.04	48.50	48.10	10.34
RAND28	53.74	53.74	4.44	53.74	53.74	7.51
RAND29	46.97	46.77	4.04	46.97	46.77	13.80
RAND30	49.47	49.47	5.82	**50.00**	49.50	6.45

In order to show the effectiveness of the proposed method, we also generate 30 random instances[1] with $|P| = 40$ and $|M| = 20$. In addition, each element a_{ij} has value 1 with probability 0.5, independently of the other elements. Table 3 shows the results. For each instance, the methods are compared using the following criteria: the best solution obtained, average solution and the average CPU time.

[1] The instances are available at http://www.ic.uff.br/~fabio/Instances_CFP_OPTL.zip.

Table 3 indicates GRASP-SP improves the best solution values in 8 out of the 30 instances. GRASP-SP is better not only with respect to the best solutions, but it also shows a superior performance in terms of average solutions.

4 Concluding remarks

In this work we presented a GRASP algorithm coupled with a SP refinement to solve the CFP. The SP refinement consists of a linear-fractional formulation for the CFP. We validate our algorithm on a set of 35 instances found in the literature and 30 randomly generated instances. Our experiments found optimal solutions for most instances, and solutions better than the existing ones in the literature when the optimal value is unknown. Ongoing research efforts are directed to the application of such a matheuristic approach to solve more complicated cellular manufacturing models for real-sized problems.

References

1. Agarwal, Y., Mathur, K., Salkin, H.M.: A set-partitioning-based exact algorithm for the vehicle routing problem. Networks **19**(7), 731–749 (1989). doi:10.1002/net.3230190702
2. Balas, E., Padberg, M.W.: Set partitioning: a survey. SIAM Rev. **18**(4), 710–760 (1976). doi:10.1137/1018115
3. Bastos, L., Ochi, L.S., Protti, F., Subramanian, A., Martins, I.C., Pinheiro, R.G.S.: Efficient algorithms for cluster editing. J. Comb. Optim. **31**(1), 347–371 (2016)
4. Boschetti, M.A., Maniezzo, V., Roffilli, M., Bolufé Röhler, A.: Matheuristics: Optimization, Simulation and Control, pp. 171–177. Springer, Berlin (2009). doi:10.1007/978-3-642-04918-7_13
5. Daz, J., Luna, D., Zetina, C.: A hybrid algorithm for the manufacturing cell formation problem. J. Heuristics **19**(1), 77–96 (2013)
6. Dimopoulos, C., Zalzala, A.: Recent developments in evolutionary computations for manufacturing optimization: problem, solutions, and comparisons. IEEE Trans. Evol. Comput. **4**(2), 93–113 (2000)
7. Elbenani, B., Ferland, J.A., Bellemare, J.: Genetic algorithm and large neighbourhood search to solve the cell formation problem. Expert Syst. Appl. **39**(3), 2408–2414 (2012)
8. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. J. Glob. Optim. **6**, 109–133 (1995)
9. Ghosh, T., Sengupta, S., Chattopadhyay, M., Dan, K.P.: Meta-heuristics in cellular manufacturing: a state-of-art review. Int. J. Ind. Eng. Comput. **2**, 87–122 (2011)
10. Gonçalves, J., Resende, M.: An evolutionary algorithm for manufacturing cell formation. Comput. Ind. Eng. **47**(2–3), 247–273 (2004)
11. Kumar, C.S., Chandrasekharan, M.P.: Group efficacy: a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. Int. J. Prod. Res. **28**(2), 233–243 (1990)
12. Maniezzo, V., Stützle, T., Voß, S.: Matheuristics: Hybridizing Metaheuristics and Mathematical Programming, 1st edn. Springer, Berlin (2009)
13. Martins, I.C., Pinheiro, R.G.S., Protti, F., Ochi, L.S.: A hybrid iterated local search and variable neighborhood descent heuristic applied to the cell formation problem. Expert Syst. Appl. **42**(22), 8947–8955 (2015)
14. Mladenovic, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
15. Pailla, A., Trindade, A.R., Parada, V., Ochi, L.S.: A numerical comparison between simulated annealing and evolutionary approaches to the cell formation problem. Expert Syst. Appl. **37**, 5476–5483 (2010)
16. Paydar, M.M., Saidi-Mehrabad, M.: A hybrid genetic-variable neighborhood search algorithm for the cell formation problem based on grouping efficacy. Comput. Oper. Res. **40**(4), 980–990 (2013)
17. Penna, P.H.V., Subramanian, A., Ochi, L.S.: An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. J. Heuristics **19**(2), 201–232 (2013)

18. Pinheiro, R.G., Martins, I.C., Protti, F., Ochi, L.S., Simonetti, L.G., Subramanian, A.: Eur. J. Oper. Res. **254**(3), 769–779 (2016). doi:10.1016/j.ejor.2016.05.010
19. Sarker, B., Khan, M.: A comparison of existing grouping efficiency measures and a new weighted grouping efficiency measure. IIE Trans. **33**, 11–27 (2001)
20. Subramanian, A., Uchoa, E., Ochi, L.S.: A hybrid algorithm for a class of vehicle routing problems. Comput. Oper. Res. **40**, 2519–2531 (2013)
21. Wu, T.H., Chang, C.C., Yeh, J.Y.: A hybrid heuristic algorithm adopting both boltzmann function and mutation operator for manufacturing cell formation problems. Int. J. Prod. Econ. **120**(2), 669–688 (2009)