Expert Systems With Applications 42 (2015) 8947–8955

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A hybrid iterated local search and variable neighborhood descent heuristic applied to the cell formation problem

Ivan C. Martins [a], Rian G.S. Pinheiro [a,b], Fábio Protti [a,*], Luiz S. Ochi [a]

[a] Fluminense Federal University - Institute of Computing, Niterói, RJ, Brazil
[b] Federal Rural University of Pernambuco, Garanhuns, PE, Brazil

ARTICLE INFO

Keywords:
Metaheuristics
Cellular manufacturing
Group technology

ABSTRACT

The Cell Formation Problem is an NP-hard optimization problem that consists of grouping machines into cells dedicated to producing a family of product parts, so that each cell operates independently and inter-cellular movements are minimized. Due to its high computational complexity, several heuristic methods have been developed over the last decades. Hybrid methods based on adaptations of popular metaheuristic techniques have shown to provide good performance in terms of solution quality. This paper proposes a new approach for solving the Cell Formation Problem using the group efficacy objective function. Our method is based on the Iterated Local Search metaheuristic coupled with a variant of the Variable Neighborhood Descent method that uses a random ordering of neighborhoods in local search phase. We consider two types of constraints on the minimum cell size, comparing them with several well-known algorithms in the literature. Computational experiments have been performed on 35 widely used benchmark instances with up to 40 machines and 100 parts. The proposed algorithm, besides obtaining solutions at least as good as any reported results, was able to find several optimal solutions and improve the group efficacy for some instances with unknown optima.

1. Introduction

Cellular Manufacturing (CM) is an application of group technology on the manufacturing process. It states that identifying and grouping elements by some form of similarity can be taken as an advantage in this kind of process (Selim, Askin, & Vakharia, 1998). This concept was originally proposed by Flanders (1924), but only through the works by Mitrofanov (1966) and Burbidge (1975) it was formalized and widespread. Since then, CM has been used as an effective strategy that combines the flexibility of functional manufacturing and the high production rate of linear manufacturing. The design of a Cellular Manufacturing System (CMS) has been called Cell Formation Problem (CFP). The objective is to form machine-part groups called cells in which products do not have to move from one cell to the other for processing (Gonçalves & Resende, 2004). These groups aim at reducing costs and increasing the efficiency of CMS's.

Due to the NP-hard nature of the CFP (Dimopoulos & Zalzala, 2000), several heuristic methods to solve it have been developed in the last decades. A reference survey on these methods is presented by Ghosh, Sengupta, Chattopadhyay, and Dan (2011), which present a taxonomic framework of metaheuristics for the CFP. Another survey, by Papaioannou and Wilson (2010), briefly reviews the different techniques used to attack the problem: cluster analysis, graph partitioning, mathematical programming, and hybridization of these methodologies. Also, Selim et al. (1998) have done an extensive review of prior research. Comprehensive summaries and taxonomies of studies devoted to part-machine grouping problems are also presented by Wemmerlov and Hyer (1989).

Non-hierarchical clustering methods are explained in detail in the paper by Srinivasan and Narendran (1991), which unlike traditional hierarchical clustering methods do not necessarily require specifying the number of clusters beforehand—clusters naturally emerge based on a given data set. However, they require identification of seeds for each potential cluster. Also, graph theoretic methods usually represent machines as vertices in graphs and similarity coefficients as weights of arcs (Oliveira, Ribeiro, & Seok, 2009; Rajagopalan & Batra, 1975). In our approach, as we shall see, we apply these concepts for generating initial solutions in the constructive phase of an ILS-based algorithm for the CFP.

The first systematic attempt to solve exactly the CFP using the group efficacy objective function is presented in the work by Pinheiro et al. (2013), where the authors first develop an exact method for a special graph biclustering problem and then apply it to the CFP, optimally solving 27 of 35 instances from the literature. In order to refine the analysis of the computational experiments, they deal separately with two scenarios related to allowing or not *singleton cells* (or simply

* Corresponding author. Tel.: +55 21 2629-5669; fax +55 21 2629–5669.
E-mail addresses: imartins@ic.uff.br (I.C. Martins), rgpinheiro@ic.uff.br (R.G.S. Pinheiro), fabio@ic.uff.br (F. Protti), satoru@ic.uff.br (L.S. Ochi).

http://dx.doi.org/10.1016/j.eswa.2015.07.050
0957-4174/

singletons) in the solution. Singletons are cells with less than two parts/machines.

Approaches for the CFP can be divided into those that allow and those that forbid singletons. In the former group, the paper by Elbenani, Ferland, and Bellemare (2012) presents a local search procedure that sequentially applies an intensification strategy to locally improve a current solution, and a diversification strategy to destroy solutions in order to form new ones; Pailla, Trindade, Parada, and Ochi (2010) have proposed two different algorithms, based on evolutionary local search and simulated annealing, that produce good solutions for the instance set used. In the latter group, Wu, Chung, and Chang (2010) adopted WFA logic, which mimics the natural behavior of water flowing from higher to lower levels; Wu, Chang, and Chung (2008) presented a simple and fast simulated annealing-based approach, although not as effective as the previous ones; and Gonçalves and Resende (2004) presented a hybrid algorithm combining local search and genetic algorithms with very promising results.

A recent work by Noktehdan, Seyedhosseini, and Saidi-Mehrabad (2015) proposes a League Championship Algorithm using population-based stochastic local search techniques for general grouping problems. The algorithm has been successfully applied to the CFP and allows solutions with singleton cells.

The aforementioned works deal each with only one of the two singleton assumptions (allowing/forbidding singleton cells). Forbidding singletons clearly impacts solution quality; therefore computational results based on one assumption cannot be fairly compared with results based on the other. As in Wu, Chang, and Yeh (2009), we analyze separately the two cases; results obtained using one of the assumptions are properly compared with approaches based on the same assumption.

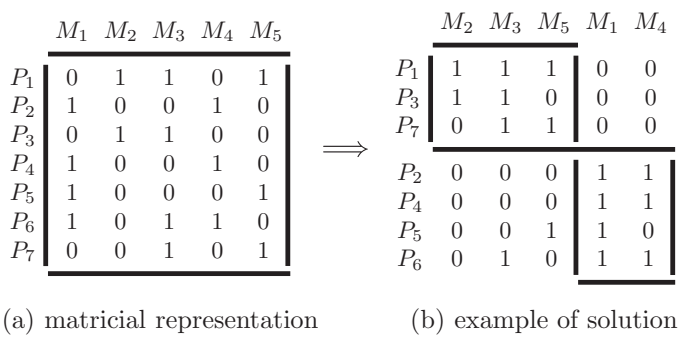
In this paper, we propose a novel hybrid algorithm to solve the CFP, based on the Iterated Local Search (ILS) metaheuristic coupled with a variant of the Variable Neighborhood Descent (VND) method that uses a random ordering of neighborhoods in local search phase. Our method uses a simple and efficient constructive method, a local search engine formed by three natural neighborhood structures, and three easy-to-implement perturbation procedures. Despite the simplicity of its modules, our method achieves expressive results when compared with the best existing algorithms for the CFP. Computational experiments conducted on a set of 35 benchmark instances show that our approach is fast and efficient, achieving optimum solutions in most instances, and better solutions than those found in the literature when the optimum is unknown.

Iterated Local Search has been successfully employed in the development of real-time expert systems for solving several combinatorial optimization problems arising in production planning, e.g. Chen, kuan Huang, and Dong (2010) Derbel, Jarboui, Hanafi, and Chabchoub (2012) Masson et al. (2013) Morais, Mateus, and Noronha (2014) Pardalos, Sun, Pei, and Zhang (2015). However, to the best of our knowledge, ILS has not been applied yet to solve the CFP. Another reason for the choice of ILS is that it requires only a small number of calibrating parameters to be adjusted during experimental procedures, unlike other methods such as, for instance, genetic algorithms.

The remainder of the paper is organized as follows. The CFP is described in Section 2; additional constraints about minimum cell size are also considered. The proposed algorithm is described in detail in Section 3. Section 4 shows computational results for test problems from the literature, and includes thorough analyses and discussions. Conclusions are laid out in Section 5.

2. The cell formation problem

The CFP aims at forming cells in which grouped machines are used in the production of a family of parts. The goal is to minimize the number of parts moved from a cell to another cell, while the number of operations within a cell is maximized. Each family consists of parts, which preferably have a high degree of similarity, that is, a large number of machines that perform common operations in these parts.



(a) matricial representation (b) example of solution

Fig. 1. CFP representation and solution.

Basically, a CMS can be represented by a binary part-machine matrix $A_{|P| \times |M|}$, where P is the set of parts and M the set of machines. An element a_{ij} is equal to 1 if an operation is performed on the part $i \in P$ by machine $j \in M$, otherwise we set $a_{ij} = 0$. The cells can be formed by a rearrangement (permutation) of rows and columns of the matrix to form part-machine blocks along the diagonal of the matrix. In Fig. 1(a), we show an example of the matrix representation of a manufacturing system, with $|P| = 7$ and $|M| = 5$, and in Fig. 1(b), an example of solution with machine cells $\{M_2, M_3, M_5\}$ and $\{M_1, M_4\}$ associated with part families $\{P_1, P_3, P_7\}$ and $\{P_2, P_4, P_5, P_6\}$, respectively.

In a perfect cell formation, each machine operates all parts assigned with its cell, and only these; thus, in this case, a part does not need to be moved to other cells because no other machine is needed to operate it. Although this is an ideal formation, a solution with this pattern is difficult to be found in practice.

2.1. Solution quality

A solution of the CFP is formed by machine cells and part families grouped to allow the best performance of a manufacturing system. To characterize a solution, we define some structures present in part-machine matrices in order to evaluate the quality of a solution:

- **Block:** a sub-matrix of a part-machine matrix that represents a manufacturing cell.
- **Operation:** an element with value 1. Such an element represents an operation of a part by a machine.
- **Void:** an element with value 0 that appears inside a block. Such an element represents a machine in a cell that does not operate some part associated with this cell, counting negatively to the cell performance.
- **Exception element:** an element with value 1 that appears outside blocks. Such an element represents a part operated by machines of distinct cells, and thus need to be moved for processing.

Group efficacy was proposed by Kumar and Chandrasekharan (1990) as a measure (index) of solution quality, in order to avoid the difficulties in establishing a weighting factor from another measure called *grouping efficiency* (Chandrasekharan & Rajagopalan, 1986b). As remarked in the work by Srinivasan and Narendran (1991), several approaches have been developed considering the group efficacy as a standard in evaluating solutions. This index is defined as:

$$\mu = \frac{e_1 - e_1^{out}}{e_1 + e_0^{in}}. \quad (1)$$

In Eq. (1), e_1 is the number of operations, e_1^{out} the number of voids, and e_0^{in} the number of exception elements. Note that, when $\mu = 1$, the solution is a perfect cell formation. We choose the group efficacy as

	M_2	M_3	M_1	M_4	M_5
P_1	1	1	0	0	1
P_3	1	1	0	0	0
P_2	0	0	1	1	0
P_4	0	0	1	1	0
P_6	0	1	1	1	0
P_5	0	0	1	0	1
P_7	0	1	0	0	1

Fig. 2. A solution allowing singleton cells.

a measure of performance for the algorithms proposed in this study, for several reasons presented below and reported by the literature:

- it is considered as a standard measure to represent the quality of solutions, more accurately then grouping efficiency;
- evaluates both minimization of voids and exception elements;
- does not require a weighting factor to prioritize one of the above two aspects;
- has a high discrimination power, i.e., works well on both matrices with few or many exception elements.

To demonstrate the applicability of this measure, we calculate the group efficacy of the example shown in Fig. 1(b). Note that there are a total of 16 part operations, 2 exception elements, and 3 voids. Thus, using Eq. (1), we have a group efficacy of $\mu = \frac{16-2}{16+3} = 0.7368$.

2.2. Minimum cell size

The cell size is usually expressed by the number of machines that comprise it. The term *singleton* is defined as a cell that has fewer than two machines or parts (Gonçalves & Resende, 2004). There are approaches that consider the presence of singletons. The paper by Paydar, Mahdavi, Sharafuddin, and Solimanpur (2010) reviews several approaches, comparing the solution method employed and whether the singletons are allowed or not to solve the CFP. However, in the literature of the CFP, there is no consensus on the minimum cell size. Among those approaches that permit singletons, Pailla et al. (2010) and Elbenani et al. (2012) allow the existence of empty cells in a solution, i.e., machines or parts that do not belong to any cell, while some methods as in James, Brown, and Keeling (2007) and Wu et al. (2008) require cells with at least one machine and at least one part.

Restricting minimum cell size makes the problem more realistic, but can decrease the group efficacy of solutions. As many results in the literature show, approaches where singletons are not allowed often have worse solutions than those which allow them. For this reason, the comparison between these distinct approaches is not fair. As in Wu et al. (2009), in this paper we will analyze separately the two approaches. When the presence of singletons is allowed, solutions with empty cells will be accepted.

Fig. 2 represents another solution of the machine-part matrix of Fig. 1(a). Unlike the solution in Fig. 1(b), now there is a cell with only one machine (M_5); hence, this is a singleton cell. There are 16 part operations, but now we have 4 exception elements and no voids; therefore, using Eq. (1), the group efficacy is $\mu = \frac{16-4}{16+0} = 0.75$, higher than the solution value in Fig. 1(b).

3. The ILS-RVND algorithm

In this section we describe the method employed in this work, based on the ILS metaheuristic. The essence of ILS consists of focusing the search on a smaller subset of the solution space, defined by the local optimum of a heuristic (Lourenço, Martin, & Sttzle, 2002). We have employed a variant of the VND procedure as the local search; here, it is a random ordering of neighborhoods, known as Randomized Variable Neighborhood Descent (RVND). The ILS is performed iteratively until an acceptance criterion is satisfied. However, instead of restarting the same procedure from a completely new solution, it performs a perturbation in the current solution. This approach generates good solutions if the perturbation is neither too small nor too big. Very small perturbations can be undone after applying the local search, while excessively large perturbations lead to random solutions which eliminate the advantage of the sampling performed by the constructive method.

However, hybrid methods that are based on adaptations of popular metaheuristic techniques have shown to provide outstanding performance in terms of solution quality (Benlic & Hao, 2013). With this in mind, we coupled the ILS with the procedure to solve the CFP. This variant has been successfully employed in Penna, Subramanian, and Ochi (2013) Silva, Bahiense, Ochi, and Boaventura (2012) Subramanian, Uchoa, and Ochi (2013).

To implement the ILS-RVND heuristic, summarized in Algorithm 1, four main methods are specified: (i) GenerateSeed, that creates seeds used to guide the constructive procedure (Fig. 3); (ii) Construct, where an solution is constructed (Section 3.2); (iii) RVND, which performs the local search procedure based on RVND method (Section 3.3); (iv) Perturb, where a new starting point is generated by perturbing the current solution (Section 3.4).

Algorithm 1 ILS-RVND.

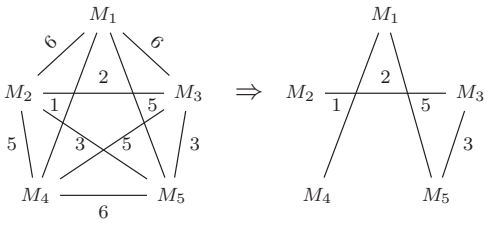
```
1:  procedure ILS-RVND(A, k)
2:      it, itILS ← 0
3:      T_m ← GenerateSeed(A)
4:      S ← Construct(T_m, 1)
5:      S* ← S
6:      repeat
7:          S' ← RVND(S)
8:          if μ(S') > μ(A*) then
9:              S* ← S'
10:             it ← 0
11:         end if
12:         if it = |M| then
13:             S ← Construct(T_m, k)
14:         else
15:             S ← Perturb(S')
16:         end if
17:         it ← it + 1
18:     until it = k × |M|
19:     return A*
20: end procedure
```

The input consists of a part-machine matrix A and a randomness factor k to be used by the constructive procedure. In matrix A, each entry a_{ij} represents the use of part $i \in P$ by a machine $j \in M$. First, we generate a seed (step (i) above), that is used by the constructive procedure (step (ii)) to create good initial solutions. A solution S is composed by machines grouped in cells and parts in families, defined as two collections MC and PF, respectively. These collections are then used in the local search phase (step (iii)) and in the perturbation phase (step (iv)) in order to improve the current solution.

In Algorithm 1, steps (iii) and (iv) are performed iteratively until a fixed number of iterations is reached; the number of iterations is based on the size of the input instance. A particular characteristic of our ILS method is constructive phase is performed in two distinct situations. First, before the ILS iterations begin, we run a greedy constructive procedure; preliminary tests show that this method often generates good solutions, and thus is a good starting point for the

	M_1	M_2	M_3	M_4	M_5
P_1	0	1	1	0	1
P_2	1	0	0	1	0
P_3	0	1	1	0	0
P_4	1	0	0	1	0
P_5	1	0	0	0	1
P_6	1	0	1	1	0
P_7	0	0	1	0	1

(a) input matrix A



(b) clique C_m (c) min. spanning tree T_m

Fig. 3. Generation of seeds.

subsequent iterations. Second, after a certain number of iterations, a constructive method is performed using a randomness factor k. In the next sections, each of these procedures is described in detail.

3.1. Seed formation

First, a machine clique C_m is created based on matrix A. C_m is a complete graph in which vertices correspond to machines and edge weights are given by the expression $d_{ij} = \sum_{k=1}^{|P|} |a_{ki} - a_{kj}|$, representing the "distance" between machines i and j, i.e., the number of parts operated by only one of the machines. Next, we obtain a minimum spanning tree T_m of C_m. In such a tree, each edge represents highly similar machines, and therefore these machines have a high probability of belonging to the same cell.

3.2. Constructive phase

In the constructive phase, seed T_m is used for creating initial solutions to the problem. We summarize in Algorithm 2 the proposed method.

Algorithm 2 Constructive procedure.

```
1:  procedure CONSTRUCT(T_m, k)
2:      while E(T_m) ≠ ∅ do
3:          for all i ∈ E(T_m) do
4:              T_m^i ← T_m − {i}
5:              get MC^i from T_m^i
6:              get PF^i from MC^i
7:              combine MC^i and PF^i to build a solution A^i
8:          end for
9:          randomly choose a solution A′ from the k best A^i solutions
10:         T_m ← T′_m
11:         if μ′ > μ* then
12:             A* ← A′
13:         end if
14:     end while
15:     return A*
16: end procedure
```

Given as input the minimum spanning tree T_m and a parameter k, an intermediary solution A^i is built (line 7) by removing an edge $i \in E(T_m)$, where the resulting forest T^i defines a collection of machine cells MC^i (line 5), as shown in Fig. 4. Given MC^i, each part $p \in P$ is associated with one cell in MC^i in order to produce the minimum number of voids and exception elements, forming a collection of part families PF^i (line 6). Finally, MC^i and PF^i are combined to form the intermediary solution A^i. After all edges have been tested, a solution A' is randomly chosen from the k best intermediary solutions A^i (line 9), and the edge whose removal from T_m forms A' is permanently removed by updating T_m as T'_m (line 10). We proceed by repeating new iterations

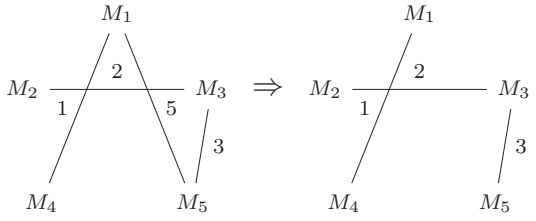


Fig. 4. Lines 4 and 5 of Algorithm 2.

	M_1	M_4	M_2	M_3	M_5
P_1	0	0	**1**	**1**	**1**
P_2	**1**	**1**	0	0	0
P_3	0	1	**1**	**1**	**0**
P_4	**1**	**1**	0	0	0
P_5	**1**	**0**	0	0	1
P_6	**1**	**1**	0	1	0
P_7	0	0	**0**	**1**	**1**

Fig. 5. Line 6 of Algorithm 2.

until no edge remains. The best solution found among all iterations is then returned.

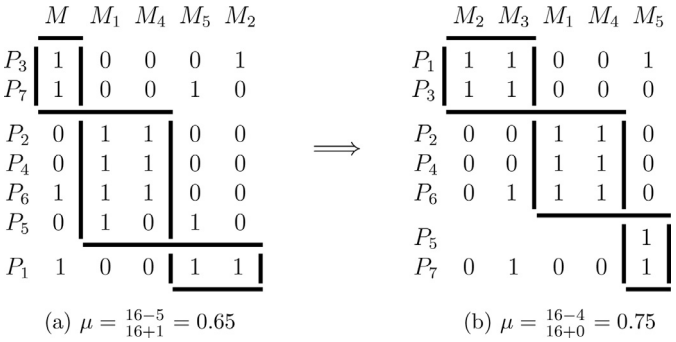
The following examples illustrate Algorithm 2. In Fig. 4, we have $MC^i = \{\{M_1, M_4\}; \{M_2, M_3, M_5\}\}$, obtained by removing edge (M_1, M_5) from T and taking each connected component as a cell in MC^i.

Next, in line 6, each part will be associated with exactly one cell from MC^i in order to produce the minimum number of voids and exceptions, forming a collection of part families PF^i. In Table 5 we illustrate this process.

Each part $p \in P$ may be associated either with cell $\{M_1, M_4\}$ or cell $\{M_2, M_3, M_5\}$, depending on which one gives a lower cost, considering the number of voids and exceptions formed. For example, associating P_5 with cell $\{M_1, M_4\}$ produces one void, but associating it with cell $\{M_2, M_3, M_5\}$ produces two voids; thus P_5 goes to cell $\{M_1, M_4\}$ (marked in bold). After associating all the parts with cells, we obtain the collection of part families $PF^i = \{\{P_2, P_3, P_5, P_6\}; \{P_1, P_4, P_7\}\}$.

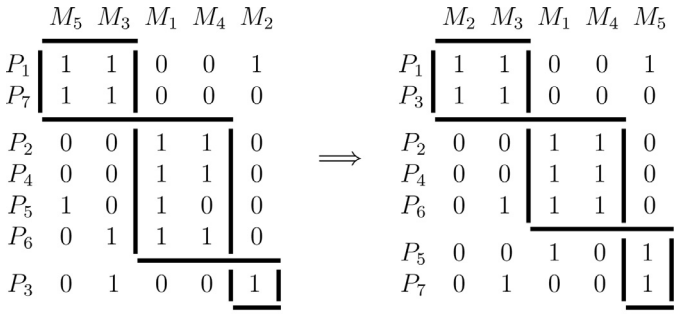
3.3. Solution improvement

After the construction phase, an initial solution is given. Then, a local search phase based on the VND method (Mladenovic & Hansen, 1997) is performed with the purpose of improving such a solution.



(a) $\mu = \frac{16-5}{16+1} = 0.65$ (b) $\mu = \frac{16-4}{16+0} = 0.75$

Fig. 6. Machine M_2 is moved from cell $\{M_5, M_2\}$ to cell $\{M_3\}$.



(a) $\mu = \frac{16-4}{16+1} = 0.71$ (b) $\mu = \frac{16-4}{16+0} = 0.75$

Fig. 7. Swap between machines M_2 and M_5.

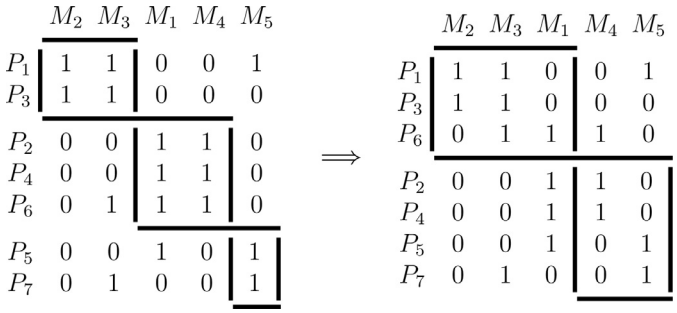


Fig. 8. Removal of cell $\{M_1, M_4\}$.

The VND procedure consists of exploring the solution space through a systematic switch between neighborhood structures, accepting only solutions that improve the current solution and returning to the first neighborhood when a better solution is found. We use a variant which uses a random ordering of the neighborhoods, as shown in Algorithm 3.

Algorithm 3 RVND.

```
1:  procedure RVND(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
```

Let $V = \{V^1, \ldots, V^r\}$ be the set of neighborhood structures. Whenever a given neighborhood fails to improve the current solution, Algorithm 3 switches to the next neighborhood. The visiting order is predefined randomly, and a subsequent call to Algorithm 3 will possibly define another visiting order.

In this work, three neighborhood structures have been employed: *Regroup, Machine Shift* and *Machine Swap*. These neighborhoods are based on modifying a machine cell and finding the associated part family in a similar way as in line 6 of Algorithm 2. The computational complexity of each of these moves is $O(n^2)$.

- **Regroup:** given the collection of part families *PF* of the current solution, regroup machines in a new collection of machine cells, depending on which one gives the minimum cost, considering the number of voids and exceptions formed.
- **Machine shift:** given the collection of machine cells *MC* of the current solution, move machines from one cell to another. Such moves perform the modification of a solution. Fig. 6 illustrates this move, where machine M_2 is moved from cell $\{M_5, M_2\}$ to cell $\{M_3\}$.
- **Machine swap:** Given the collection of cell machines *MC* of the current solution, swaps machines from different cells. Unlike Machine Shift, this neighborhood lacks the ability to empty a cell; thus it cannot decrease the number of cells. Fig. 7 illustrates this move, where a swap is made between machines M_2 and M_5.

3.4. Perturbation

As described in Algorithm 1, ILS repeatedly explores the neighborhood of solutions by perturbing locally optimal solutions previously visited. We present below three perturbation procedures proposed in this work. Once the procedure Perturb is called, one of those perturbations is randomly selected.

- **Multiple machine shift:** performs a random number of successive Machine Shifts between cells. To do this, both the machine and the machine cell where this machine will be moved to are randomly chosen. A point to note here is the number of moves performed. A low number of moves results in few changes in the solution, and then the subsequent local search can undo these changes. On the other hand, a high number of moves makes the neighborhood too large. Therefore, the number of moves ranges between the minimum cell size and the number of cells.
- **Cell removal:** randomly selects a cell and removes it, distributing all the machines belonging to it to other cells, also chosen at random. An interesting point here is that this perturbation can be viewed as Multiple Machine Shift when all machines are chosen in the same cell. The goal of Cell Removal is to decrease the number of machine cells. Fig. 8 shows an example, where cell $\{M_1, M_4\}$ is chosen to be removed; machines M_1 and M_5 are randomly moved to other cells.
- **Cell split:** randomly selects a cell and divides it into two new cells M' and M''. The probability of a machine being moved to M' (or M'') is 50%. Unlike other perturbations, Cell Split never moves a machine to an existing cell; its goal is precisely to increase the number of machine cells. Fig. 9 shows an example, where cell $\{M_2, M_3, M_5\}$ is splitted into two new cells: $\{M_2, M_3\}$ and $\{M_5\}$.

4. Computational results

In this section we report computational results for Algorithm 1. In Section 4.1 we describe the instances employed in our experiments.

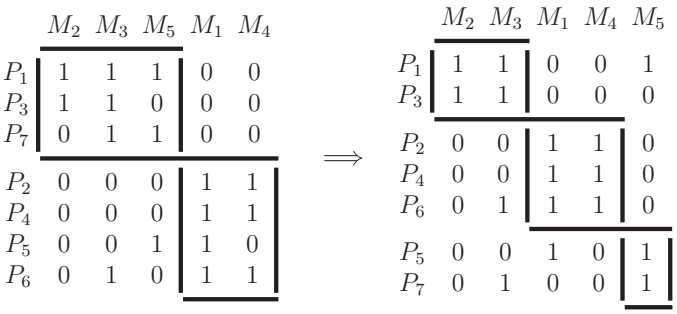


Fig. 9. Cell $\{M_2, M_3, M_5\}$ is splitted into two new cells.

Table 1
Instances of the CFP used in our experiments.

Instance	Size	Problem source
CFP01	05 × 07	King and Nakornchai (1982)
CFP02	05 × 07	Waghodekar and Sahu (1984)
CFP03	05 × 18	Seifoddini (1989)
CFP04	06 × 08	Kusiak and Cho (1992)
CFP05	07 × 11	Kusiak and Chow (1987)
CFP06	07 × 11	Boctor (1991)
CFP07	08 × 12	Seifoddini and Wolfe (1986)
CFP08	08 × 20	Chandrasekharan and Rajagopalan (1986b)
CFP09	08 × 20	Chandrasekharan and Rajagopalan (1986a)
CFP10	10 × 10	Mosier and Taube (1985a)
CFP11	10 × 15	Chan and Milner (1982)
CFP12	14 × 23	Askin and Subramanian (1987)
CFP13	14 × 24	Stanfel (1985)
CFP14	16 × 24	McCormick, Schweitzer, and White (1972)
CFP15	16 × 30	Srinivasan, Narendran, and Mahadevan (1990)
CFP16	16 × 43	King (1980)
CFP17	18 × 24	Carrie (1973)
CFP18	20 × 20	Mosier and Taube (1985b)
CFP19	20 × 23	Kumar, Kusiak, and Vannelli (1986)
CFP20	20 × 35	Carrie (1973)
CFP21	20 × 35	Boe and Cheng (1991)
CFP22	24 × 40	Chandrasekharan and Rajagopalan (1989)
CFP23	24 × 40	Chandrasekharan and Rajagopalan (1989)
CFP24	24 × 40	Chandrasekharan and Rajagopalan (1989)
CFP25	24 × 40	Chandrasekharan and Rajagopalan (1989)
CFP26	24 × 40	Chandrasekharan and Rajagopalan (1989)
CFP27	24 × 40	Chandrasekharan and Rajagopalan (1989)
CFP28	27 × 27	McCormick et al. (1972)
CFP29	28 × 46	Carrie (1973)
CFP30	30 × 51	Kumar and Vannelli (1987)
CFP31	30 × 50	Stanfel (1985)
CFP32	30 × 50	Stanfel (1985)
CFP33	30 × 90	King and Nakornchai (1982)
CFP34	37 × 53	McCormick et al. (1972)
CFP35	40 × 100	Chandrasekharan and Rajagopalan (1987)

Next, in Section 4.2, we present the results of the experiments, evaluating their performance with respect to the quality of the solutions. Finally, in Section 4.3, the best results found in the literature are compared with our results.

We present results on two variants of the CFP: the Cell Formation Problem Allowing Singletons (CFPAS) and the Cell Formation Problem Forbidding Singletons (CFPFS). To evaluate the quality of the solutions, we use the group efficacy measure, as explained in Section 2.

In all the tables, group efficacy values are presented as percentages with two decimal digits, and computational times are given in seconds. All the algorithms have been run on an Intel Core i7-2600 processor with 3.40 GHz and 32GB of RAM, using OS Arch Linux 3.3.4.

4.1. Instances

We use 35 benchmark instances from the literature, available in the work by Gonçalves and Resende (2004), widely used by many authors to evaluate their methods. The selected matrices have sizes ranging from 5 × 7 to 40 × 100, with different degrees of difficulty, admitting solutions with perfect cells to solutions with poorly structured cells, i.e., with a large number of exceptions and voids. Columns in Table 1 show, from left to right: instance name, instance size (recall that an instance is given as an input matrix $A_{m \times p}$, where m is the number of machines and p the number of parts) and reference source (i.e., where the instance is specified)

Table 2
Best solutions found in the literature.

Instance	CFPFS		CFPAS	
	Best	First approach	Best	First approach
CFP01*	73.68	(Gonçalves & Resende, 2004)	75.00	(Pailla et al., 2010)
CFP02*	62.50	(Gonçalves & Resende, 2004)	69.57	(Wu et al., 2009)
CFP03*	79.59	(Gonçalves & Resende, 2004)	80.85	(Pailla et al., 2010)
CFP04*	76.92	(Gonçalves & Resende, 2004)	79.17	(Pailla et al., 2010)
CFP05*	53.13	(Gonçalves & Resende, 2004)	60.87	(Wu et al., 2009)
CFP06*	70.37	(Gonçalves & Resende, 2004)	70.83	(Wu et al., 2009)
CFP07*	68.29	(Gonçalves & Resende, 2004)	69.44	(Pailla et al., 2010)
CFP08*	85.25	(Gonçalves & Resende, 2004)	85.25	(Wu et al., 2009)
CFP09*	58.72	(Gonçalves & Resende, 2004)	58.72	(Wu et al., 2009)
CFP10*	70.59	(Gonçalves & Resende, 2004)	75.00	(Wu et al., 2009)
CFP11*	92.00	(Gonçalves & Resende, 2004)	92.00	(Wu et al., 2009)
CFP12*	69.86	(Gonçalves & Resende, 2004)	74.24	(Pailla et al., 2010)
CFP13*	69.33	(Gonçalves & Resende, 2004)	72.86	(Pailla et al., 2010)
CFP14*	51.96	(Gonçalves & Resende, 2004)	53.33	(Pailla et al., 2010)
CFP15*	67.83	(Gonçalves & Resende, 2004)	69.92	(Pailla et al., 2010)
CFP16*	56.52	(Pinheiro et al., 2013)	58.04	(Pinheiro et al., 2013)
CFP17*	54.46	(Gonçalves & Resende, 2004)	57.73	(Pailla et al., 2010)
CFP18	42.96	(Gonçalves & Resende, 2004)	43.97	(Pailla et al., 2010)
CFP19*	49.65	(Gonçalves & Resende, 2004)	50.81	(Wu et al., 2009)
CFP20*	76.54	(Wu et al., 2010)	79.38	(Pailla et al., 2010)
CFP21*	58.15	(Wu et al., 2010)	58.79	(Pailla et al., 2010)
CFP22*	100.0	(Gonçalves & Resende, 2004)	100.00	(Wu et al., 2009)
CFP23*	85.11	(Gonçalves & Resende, 2004)	85.11	(Wu et al., 2009)
CFP24*	73.51	(Gonçalves & Resende, 2004)	73.51	(Wu et al., 2009)
CFP25*	51.97	(Gonçalves & Resende, 2004)	53.29	(Wu et al., 2009)
CFP26	47.37	(Wu et al., 2010)	48.95	(Elbenani et al., 2012)
CFP27	44.87	(Gonçalves & Resende, 2004)	46.58	(Elbenani et al., 2012)
CFP28	54.27	(Gonçalves & Resende, 2004)	54.82	(Pailla et al., 2010)
CFP29	46.06	(Wu et al., 2010)	47.68	(Pailla et al., 2010)
CFP30*	58.94	(Pinheiro et al., 2013)	63.04	(Pinheiro et al., 2013)
CFP31*	59.66	(Gonçalves & Resende, 2004)	59.77	(Pinheiro et al., 2013)
CFP32	50.51	(Gonçalves & Resende, 2004)	50.83	(Wu et al., 2009)
CFP33	42.64	(Gonçalves & Resende, 2004)	47.93	(Pailla et al., 2010)
CFP34	59.85	(Wu et al., 2010)	61.16	(Pailla et al., 2010)
CFP35*	84.03	(Gonçalves & Resende, 2004)	84.03	(Wu et al., 2009)

* instances with known optimal solutions, confirmed in Pinheiro et al. (2013).

In Table 2, we show the best group efficacy value found so far in the literature for the CFPFS and the CFPAS. For each efficacy value, we indicate the paper in which it appeared first. This set of 35 instances received an important contribution from Pinheiro et al. (2013), where the authors prove the optimum solution for 27 of these instances (marked with "∗").

4.2. Performance evaluation

In this section we evaluate the performance of our methods with respect to the quality of solutions. We perform two sets of tests, one for the CFPAS and other for the CFPFS. In each set, our ILS-RVND method is run ten times for each instance; the number of iterations in each run is based on the number of machines $|M|$ of the instance and a randomness factor k. As shown in Algorithm 1, every time an improved solution is found, the iteration counter is reset to zero; when the counter reaches $k \times |M|$, the algorithm stops – no solution improvement has been achieved. After some preliminary experiments, we have defined $k = 2$.

In Table 3, performance evaluation results of our ILS-RVND method are presented. For each instance, the table shows: the best

Table 3
Performance evaluation on 35 instances from the literature.

Instance	CFPAS				CFPFS			
	Best	Avg.	sd	Time (s)	Best	Avg.	sd	Time (s)
CFP1	75.00	75.00	0	0.026	73.68	73.68	0	0.024
CFP2	69.57	69.57	0	0.007	62.50	62.50	0	0.007
CFP3	80.85	80.85	0	0.012	79.59	79.59	0	0.011
CFP4	79.17	79.17	0	0.007	76.92	76.92	0	0.008
CFP5	60.87	60.87	0	0.044	53.13	53.13	0	0.041
CFP6	70.83	70.83	0	0.024	70.37	70.37	0	0.023
CFP7	69.44	69.44	0	0.034	68.29	68.29	0	0.038
CFP8	85.25	85.25	0	0.044	85.25	85.25	0	0.043
CFP9	58.72	58.72	0	0.018	58.72	58.72	0	0.020
CFP10	75.00	75.00	0	0.053	70.59	70.59	0	0.052
CFP11	92.00	92.00	0	0.088	92.00	92.00	0	0.089
CFP12	74.24	74.24	0	0.150	69.86	69.86	0	0.149
CFP13	72.86	72.86	0	0.145	69.33	69.33	0	0.151
CFP14	53.33	53.33	0	0.286	51.96	51.96	0	0.264
CFP15	69.92	69.92	0	0.302	67.83	67.83	0	0.305
CFP16	57.96	57.96	0	0.664	56.52	56.37	0.29	0.678
CFP17	57.73	57.73	0	0.243	54.46	54.46	0	0.237
CFP18	43.97	43.76	0.21	0.349	42.96	42.93	0.05	0.353
CFP19	50.81	50.73	0.08	0.839	49.65	49.65	0	0.690
CFP20	79.38	79.38	0	0.457	76.54	76.54	0	0.392
CFP21	58.70	58.70	0	0.430	58.15	58.15	0	0.406
CFP22	100.00	100.00	0	0.437	100.00	100.00	0	0.365
CFP23	85.11	85.11	0	0.970	85.11	85.11	0	0.960
CFP24	73.51	73.51	0	1.897	73.51	73.51	0	1.682
CFP25	53.29	53.29	0	1.274	51.97	51.95	0.04	1.347
CFP26	48.95	48.79	0.16	1.467	47.06	47.04	0.03	1.547
CFP27	46.26	46.26	0	2.186	44.87	44.84	0.09	2.139
CFP28	54.82	54.82	0	0.739	54.27	54.27	0	0.694
CFP29	47.68	47.53	0.11	3.302	45.57	45.57	0	2.756
CFP30	62.94	62.94	0	2.142	58.90	58.86	0.13	2.296
CFP31	59.77	59.77	0	3.777	59.66	59.66	0	3.556
CFP32	50.83	50.83	0	3.148	50.51	50.51	0	3.354
CFP33	47.93	47.93	0	4.606	43.37	43.22	0.31	4.735
CFP34	61.16	61.16	0	1.441	59.39	59.39	0	1.322
CFP35	84.03	84.03	0	6.955	84.03	84.03	0	5.810

solution value among all runs; the average solution value; the standard deviation; and the average computational time.

Results show the robustness of our method, as can be seen from the low standard deviations and computational times. Notice that the standard deviation is not zero only for four cases of the CFPAS and seven cases of the CFPFS. It can also be observed that when singletons are allowed the method needs more computational effort to find a solution; this happens because the CFPAS has a larger search space.

4.3. Comparison with the results in the literature

Now we compare the results presented in Section 4.2 with results from several methods in the literature, on the same instance set.

For each pair (I, A), where I is an instance and A is a method, we calculate the gap between the value of the solution obtained by A when applied to I and the value of the best known solution for I (shown in Table 1), as follows:

$$\%gap = \frac{\mu(s^*) - \mu(s)}{\mu(s^*)} \quad (2)$$

where $\mu(s)$ is the group efficacy of the solution s obtained by method A when applied to I, and $\mu(s^*)$ the group efficacy of the best known solution s^* for I.

In subsequent tables, instances marked with "$*$" have known optima. In such cases, if $\%gap = 0$ then the method was able to find the optimum. On the other hand, for instances with unknown optima, $\%gap = 0$ simply means that the method was able to find the best existing result.

A negative gap means that a solution better than the previously best (not necessarily optimum) was found by the method. We remark that if a negative gap occurs for a pair (I, A) and I is marked with "$*$" (have known optima) then an error (whose cause is unknown by the authors of this work) occurred when applying A to I. Such an event is marked with "$\times$".

We consider first the CFPFS. In Table 4 we present a comparison between our ILS-RVND and the following approaches for the CFPFS. found in the literature:

SACF: *Simulated Annealing for Cell Formation* (Wu et al., 2008)
WFACF: *Water Flow-like Algorithm for Cell Formation* (Wu et al., 2010)
EA: *Evolutionary Algorithm* (Gonçalves & Resende, 2004)

Table 4 shows the *%gap* for each pair (I, A). In addition, for each method A, the table shows the number of times A finds the optimum and the number of times A finds the best existing solution.

For the CFPFS, our ILS-RVND method outperforms the other approaches finding the optimal solution for 26 instances and the best solution (not necessarily optimum) for 31 of 35 instances. In particular, only ILS-RVND was able to find an optimal solution for instance CFP16; moreover, it has found a new solution for instance CFP33.

Now, we consider the CFPAS. Results are presented in Table 5, in the same way as in Table 4. Our ILS-RVND method is compared with the following ones:

SCM–BMCF: *Similarity Coefficients Method with Boltzmann function and Mutation operator for Cell Formation* (Wu et al., 2009)
SA: *Simulated Annealing* (Pailla et al., 2010)
HMGA: *Hybrid Method Genetic Algorithm* (Elbenani et al., 2012)

Table 4
CFPFS: Comparison between ILS-RVND and other approaches from the literature.

Instance	SACF	WFACF	EA	ILS-RVND	Instance	SACF	WFACF	EA	ILS-RVND
CFP01∗	0.00	0.00	0.00	0.00	CFP19∗	0.00	0.08	0.00	0.00
CFP02∗	0.00	0.00	0.00	0.00	CFP20∗	0.52	0.00	0.42	0.00
CFP03∗	0.00	0.00	0.00	0.00	CFP21∗	3.63	0.00	0.14	0.00
CFP04∗	0.00	0.00	0.00	0.00	CFP22∗	0.00	0.00	0.00	0.00
CFP05∗	0.00	0.00	0.00	0.00	CFP23∗	0.00	0.00	0.00	0.00
CFP06∗	0.00	0.00	0.00	0.00	CFP24∗	0.00	0.00	0.00	0.00
CFP07∗	0.00	0.00	0.00	0.00	CFP25∗	0.17	0.00	0.00	0.00
CFP08∗	0.00	0.00	0.00	0.00	CFP26	6.19	0.00	0.65	0.65
CFP09∗	0.53	0.00	0.00	0.00	CFP27	1.40	0.00	0.00	0.00
CFP10∗	0.00	0.00	0.00	0.00	CFP28	1.99	0.00	0.00	0.00
CFP11∗	0.00	0.00	0.00	0.00	CFP29	11.95	0.00	3.13	1.06
CFP12∗	3.22	0.00	0.00	0.00	CFP30∗	0.61	×	0.78	0.07
CFP13∗	0.00	0.00	0.00	0.00	CFP31∗	×	×	0.00	0.00
CFP14∗	0.00	0.00	0.00	0.00	CFP32	0.00	0.00	0.00	0.00
CFP15∗	0.00	0.00	0.00	0.00	CFP33	6.31	0.00	0.00	−1.71
CFP16∗	1.34	1.11	2.94	0.00	CFP34	2.71	0.00	5.73	0.77
CFP17∗	1.51	0.00	0.00	0.00	CFP35∗	0.00	0.00	0.00	0.00
CFP18	1.44	0.00	0.00	0.00					
					# Opt	18	23	23	26
					# Best	19	30	27	31

Table 5
CFPAS: comparison between ILS-RVND and other approaches from the literature.

Instance	SCM-BMCF	SA	HMGA	ILS-RVND	Instance	SCM-BMCF	SA	HMGA	ILS-RVND
CFP01∗	–	0.00	×	0.00	CFP19∗	0.00	0.00	0.00	0.00
CFP02∗	0.00	0.00	0.00	0.00	CFP20∗	1.23	0.00	1.85	0.00
CFP03∗	1.56	0.00	1.56	0.00	CFP21∗	2.01	0.00	1.38	0.00
CFP04∗	2.84	0.00	2.84	0.00	CFP22∗	0.00	0.00	0.00	0.00
CFP05∗	0.00	1.43	0.00	0.00	CFP23∗	0.00	0.00	0.00	0.00
CFP06∗	0.00	0.00	0.00	0.00	CFP24∗	0.00	0.00	0.00	0.00
CFP07∗	–	0.00	0.00	0.00	CFP25∗	0.00	0.00	0.00	0.00
CFP08∗	0.00	0.00	0.00	0.00	CFP26	0.65	0.78	0.00	0.00
CFP09∗	0.00	0.00	0.00	0.00	CFP27	0.92	1.25	0.00	0.69
CFP10∗	0.00	0.00	0.00	0.00	CFP28	–	0.00	0.00	0.00
CFP11∗	0.00	0.00	0.00	0.00	CFP29	–	0.00	1.30	0.00
CFP12∗	–	0.00	2.94	0.00	CFP30∗	0.71	0.29	×	0.16
CFP13∗	1.41	0.00	1.41	0.00	CFP31∗	0.59	0.18	×	0.00
CFP14∗	–	0.00	0.13	0.00	CFP32	0.00	0.55	0.00	0.00
CFP15∗	–	0.00	0.56	0.00	CFP33	–	0.00	0.38	0.00
CFP16∗	2.86	0.14	0.88	0.14	CFP34	–	0.00	0.87	0.00
CFP17∗	–	0.00	0.00	0.00	CFP35∗	0.00	0.00	0.00	0.00
CFP18	1.61	0.00	1.18	0.00					
					# Opt	13	23	14	25
					# Best	14	28	19	32

From Table 5, ILS-RVND finds the optimal solution for 25 of 27 instances, and the best solution for 32 of 35 instances. For instance CFP31 it succeeds in finding optimal solutions, differently from the other methods.

5. Concluding remarks

The main contribution of this paper is a simple and efficient ILS method to solve an important problem in the field of production planning, the Cell Formation Problem (CFP). Unlike several metaheuristics proposed to this problem, our algorithm is based on a general, flexible framework, and is easy adaptable for dealing with minimum cell size constraints, i.e., it can be used to solve the two variants of the CFP (allowing/forbidding singleton cells). Up to the authors' knowledge, the only existing study that considers both variants is (Wu et al., 2009); however, its results are not competitive in relation to other proposals.

When singleton cells are allowed, the best heuristic method found in previous literature is WFACF (Wu et al., 2010). Our method finds three more optimal solutions than WFACF (26 versus 23); and when we look to best solutions found (not necessarily optimal), our approach finds 31 (versus 30) best solutions. Similarly, when forbidding singleton cells in the solution, the best heuristic method in literature is SA (Pailla et al., 2010). Our method finds two more optimal solutions than SA (25 versus 23) and 32 (versus 28) best solutions.

We remark that a limitation of most metaheuristics for the CFP is concerned with the construction of good initial solutions. First grouping machines and then using this information to group parts can limit the quality of initial solutions obtained. Our constructive phase presented in Section 3.2 attempts to settle such a limitation by associating a part to a subset of machines according to a minimization criterion.

Future research includes the development of a hybrid approach combining the advantages of ILS-RVND with an exact method based on Set Partitioning, as succesfully done in Subramanian et al. (2013). The constructive procedure can be improved by considering simultaneous formation of part-machine assignments. Finally, research efforts can be directed to adapt our method to extensions of the CFP, described in Defersha and Chen (2006), where several production factors can be considered at the same time: machine reliability, scheduling, cell load balance, cell layout, and intracellular machine layout.

Acknowledgments

The authors thank the financial support of CAPES, CNPq, and FAPERJ, Brazilian research agencies.

References

Askin, R. G., & Subramanian, S. P. (1987). A cost-based heuristic for group technology configuration. *International Journal of Production Research, 25*, 101.

Benlic, U., & Hao, J.-K. (2013). Hybrid metaheuristics for the graph partitioning problem. In E.-G. Talbi (Ed.), *Hybrid metaheuristics*. In Studies in computational intelligence: *434* (pp. 157–185). Springer.

Boctor, F. (1991). A linear formulation of the machine-part cell formation problem. *International Journal of Production Research, 29*(2), 343–356.

Boe, W. J., & Cheng, C. H. (1991). A close neighbour algorithm for designing cellular manufacturing systems. *International Journal of Production Research, 29*(2), 2097–2116.

Burbidge, J. (1975). *The introduction of group technology*. New York: John Wiley & Sons.

Carrie, A. S. (1973). Numerical taxonomy applied to group technology and plant layout. *International Journal of Production Research, 11*(2), 399–416.

Chan, H. M., & Milner, D. A. (1982). Direct clustering algorithm for group formation in cellular manufacture. *Journal of Manufacturing Systems, 1*, 63.

Chandrasekharan, M. P., & Rajagopalan, R. (1986a). An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. *International Journal of Production Research, 24*(2), 451–464.

Chandrasekharan, M. P., & Rajagopalan, R. (1986b). Modroc: an extension of rank order clustering for group technology. *International Journal of Production Research, 24*(2), 1221–1233.

Chandrasekharan, M. P., & Rajagopalan, R. (1987). Zodiac - an algorithm for concurrent formation of part-families and machine-cells. *International Journal of Production Research, 25*(2), 835–850.

Chandrasekharan, M. P., & Rajagopalan, R. (1989). Groupability: An analysis of the properties of binary data matrices for group technology. *International Journal of Production Research, 27*, 1035.

Chen, P., kuan Huang, H., & Dong, X.-Y. (2010). Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. *Expert Systems with Applications, 37*(2), 1620–1627.

Defersha, F. M., & Chen, M. (2006). A comprehensive mathematical model for the design of cellular manufacturing systems. *International Journal of Production Economics, 103*(2), 767–783.

Derbel, H., Jarboui, B., Hanafi, S., & Chabchoub, H. (2012). Genetic algorithm with iterated local search for solving a location-routing problem. *Expert Systems with Applications, 39*(2), 2865–2871.

Dimopoulos, C., & Zalzala, A. (2000). Recent developments in evolutionary computations for manufacturing optimization: problem, solutions, and comparisons. *IEEE Transactions on Evolutionary Computation, 4*(2), 93–113.

Elbenani, B., Ferland, J. A., & Bellemare, J. (2012). Genetic algorithm and large neighbourhood search to solve the cell formation problem. *Expert Systems with Applications, 39*(2), 2408–2414.

Flanders, R. E. (1924). Design, manufacture and production control of a standard machine. *Transactions of the American Society of Mechanical Engineers, 46*, 691–738.

Ghosh, T., Sengupta, S., Chattopadhyay, M., & Dan, K. P. (2011). Meta-heuristics in cellular manufacturing: A state-of-art review. *International Journal of Industrial Engineering Computations, 2*, 87–122.

Gonçalves, J., & Resende, M. (2004). An evolutionary algorithm for manufacturing cell formation. *Computers & Industrial Engineering, 47*(2–3), 247–273.

James, T. L., Brown, E. C., & Keeling, K. B. (2007). A hybrid grouping genetic algorithm for the cell formation problem. Computers & Operations Research, *34*(2), 2059–2079.

King, J. (1980). Machine-component grouping in production flow analysis: An approach using a rank order clustering algorithm. *International Journal of Production Research, 18*(2), 213–232.

King, J. R., & Nakornchai, V. (1982). Machine-component group formation in group technology: review and extension. *International Journal of Production Research, 20*(2), 117–133.

Kumar, C. S., & Chandrasekharan, M. P. (1990). Group efficacy: a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. *International Journal of Production Research, 28*(2), 233–243.

Kumar, K. R., Kusiak, A., & Vannelli, A. (1986). Grouping of parts and components in flexible manufacturing systems. *European Journal of Operational Research, 24*(2), 387–397.

Kumar, K. R., & Vannelli, A. (1987). Strategic subcontracting for efficient disaggregated manufacturing. *International Journal of Production Research, 25*(2), 1715–1728.

Kusiak, A., & Cho, M. (1992). Similarity coefficient algorithms for solving the group technology problem. *International Journal of Production Research, 30*(2), 2633–2646.

Kusiak, A., & Chow, W. (1987). Efficient solving of the group technology problem. *Journal of Manufacturing Systems, 6*(2), 117–124.

Lourenço, H. R., Martin, O. C., & Sttzle, T. E. (2002). Iterated local search. *Handbook of metaheuristics* (pp. 321–353). Kluwer Academic Publishers.

Masson, R., Vidal, T., Michallet, J., Penna, P. H. V., Petrucci, V., Subramanian, A., & Dubedout, H. (2013). An iterated local search heuristic for multi-capacity bin packing and machine reassignment problems. *Expert Systems with Applications, 40*(2), 5266–5275.

McCormick, W. T., Schweitzer, P. J., & White, T. W. (1972). Problem decomposition and data reorganization by a clustering technique. *Operations Research, 20*(2), 993–1009.

Mitrofanov, S. P. (1966). *The scientific principles of group technology*. National Lending Library for Science and Technology.

Mladenovic, N., & Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research, 24*(2), 1097–1100.

Morais, V. W. C., Mateus, G. R., & Noronha, T. F. (2014). Iterated local search heuristics for the vehicle routing problem with cross-docking. *Expert Systems with Applications, 41*(2), 7495–7506.

Mosier, C., & Taube, L. (1985a). The facets of group technology and their impacts on implementation—a state-of-the-art survey. *Omega, 13*(2), 381–391.

Mosier, C., & Taube, L. (1985b). Weighted similarity measure heuristics for the group technology machine clustering problem. *Omega, 13*(2), 577–579.

Noktehdan, A., Seyedhosseini, S., & Saidi-Mehrabad, M. (2015). A metaheuristic algorithm for the manufacturing cell formation problem based on grouping efficacy. *The International Journal of Advanced Manufacturing Technology*. Available online:13p.

Oliveira, S., Ribeiro, J. F. F., & Seok, S. C. (2009). A spectral clustering algorithm for manufacturing cell formation. *Computers &* Industrial Engineering, *57*, 1008–1014.

Pailla, A., Trindade, A. R., Parada, V., & Ochi, L. S. (2010). A numerical comparison between simulated annealing and evolutionary approaches to the cell formation problem. *Expert Systems with Applications, 37*, 5476–5483.

Papaioannou, G., & Wilson, J. M. (2010). The evolution of cell formation problem methodologies based on recent studies (1997–2008): review and direction for future search. *European Journal of Operational Research, 206*, 509–521.

Pardalos, P. M., Sun, H., Pei, J., & Zhang, Y. (2015). Iterated local search embedded adaptive neighborhood selection approach for the multi-depot vehicle routing problem with simultaneous deliveries and pickups. *Expert Systems with Applications, 42*(2), 3551–3561.

Paydar, M. M., Mahdavi, I., Sharafuddin, I., & Solimanpur, M. (2010). Applying simulated annealing for designing cellular manufacturing systems using MDmTSP. *Computers & Industrial Engineering, 59*(4), 929–936.

Penna, P. H. V., Subramanian, A., & Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics, 19*(2), 201–232.

Pinheiro, R. G. S., Martins, I. C., Protti, F., Ochi, L. S., Simonetti, L. G., & Subramanian, A. (2013). On solving manufacturing cell formation via bicluster editing. *Technical report*. Cornell University Library *arXiv:1312.3288*. [math.OC].

Rajagopalan, R., & Batra, J. L. (1975). Design of cellular production systems a graph-theoretic approach. *International Journal of Production Research, 13*(2), 567–579.

Seifoddini, H. (1989). Single linkage versus average linkage clustering in machine cells formation applications. *Computers & Industrial Engineering, 16*(2), 419–426.

Seifoddini, H., & Wolfe, P. M. (1986). Application of the similarity coefficient method in group technology. *IIE Transactions, 18*(2), 271–277.

Selim, H. M., Askin, R. G., & Vakharia, A. J. (1998). Cell formation in group technology: Review, evaluation and directions for future research. *Computers & Industrial Engineering, 34*(1), 3–20.

Silva, G. C., Bahiense, L., Ochi, L. S., & Boaventura, P. O. (2012). The dynamic space allocation problem: applying a hybrid grasp and tabu search metaheuristics. *Computers & Operations Research, 39*, 671–677.

Srinivasan, G., & Narendran, T. T. (1991). Grafics - a nonhierarchical clustering-algorithm for group technology. *International Journal of Production Research, 29*(2), 463–478.

Srinivasan, G., Narendran, T. T., & Mahadevan, B. (1990). Assignment model for the part-families problem in group technology. *International Journal of Production Research, 28*(2), 145–152.

Stanfel, L. E. (1985). Machine clustering for economic production. *Engineering Costs and Production Economics, 9*(2), 73–81.

Subramanian, A., Uchoa, E., & Ochi, L. S. (2013). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research, 40*, 2519–2531.

Waghodekar, P. H., & Sahu, S. (1984). Machine-component cell formation in group technology: MACE. *International Journal of Production Research, 22*(2), 937–948.

Wemmerlov, U., & Hyer, N. L. (1989). Cellular manufacturing in the US industry: a survey of users. *International Journal of Production Research, 27*, 1511–1530.

Wu, T. H., Chang, C. C., & Chung, S. H. (2008). A simulated annealing algorithm for manufacturing cell formation problems. *Expert Systems with Applications, 34*(2), 1609–1617.

Wu, T. H., Chang, C. C., & Yeh, J. Y. (2009). A hybrid heuristic algorithm adopting both Boltzmann function and mutation operator for manufacturing cell formation problems. *International Journal of Production Economics, 120*(2), 669–688.

Wu, T. H., Chung, S. H., & Chang, C. C. (2010). A water flow-like algorithm for manufacturing cell formation problems. *European Journal of Operational Research, 205*(2), 346–360.