Expert Systems With Applications 135 (2019) 327–350

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A novel hybrid algorithm for manufacturing cell formation problem

Milos Danilovic*, Oliver Ilic

Department of Operations Management, Faculty of Organizational Sciences, University of Belgrade, Jove Ilića 154, 11000 Beograd, Serbia

ARTICLE INFO

Article history:
Received 19 November 2018
Revised 30 May 2019
Accepted 9 June 2019
Available online 11 June 2019

Keywords:
Cell formation
Part-machine clustering
Grouping efficacy index
Cellular manufacturing
Feasible solution set

ABSTRACT

The cell formation problem is a crucial component of a cell production design in a manufacturing system. Problems related to the cell formation problem are complex NP-hard problems. The goal of the work is to design the algorithm for the cell formation problem that is more efficient then the best-known algorithms for the same problem. The strategy of the new approach is to use the specificities of the input instances to narrow down the feasible set, and thus increase the efficiency of the optimization process. In the dynamic production environment, efficacy is one of the most significant characteristics of the applied expert system. The result is, extensible hybrid algorithm that can be used to solve complex, multi-criteria optimization cell formation problems. The new algorithm produces solutions that are as good as, or better than, the best results previously reported in literature on all commonly used test instances. The time efficiency of the proposed algorithm is at least an order of magnitude better than the efficiency of the most efficient reported algorithms. The obtained experimental results, modularity and generality of the new algorithm imply the significant impact on the expert systems for cell formation problem since the proposed strategy can improve the efficiency of existing algorithms for the grouping problems.

1. Introduction

Group Technology (GT) is defined as an approach to work optimization in which the organizational production units are relatively independent. In other words, GT is a manufacturing philosophy that exploits similarities in product design and product processes. Products needing similar operations and a common set of resources are grouped into families, the resources being regrouped into production subsystems, cells. The major advantages of cellular manufacturing reported in the literature include: reduction in setup time, reduction in labor and overtime costs, reduction in work-in-process inventories, reduction in material handling costs, and faster response to internal and external changes such as: machine failures, product mix, and demand changes (Wu, Chang, & Yeh, 2009). In the globalized and interconnected market, a production system must have a high degree of flexibility and agility to deal with product changes. The Dynamic Cellular Manufacturing System (DCMS) is one of the well-known production systems that meet this requirement. Problems related to DCMS are complex and time-consuming NP-hard problems in their nature. The memory and computational time requirements are extremely high, and increase exponentially, as the problem size increases. In the dynamic production environment, a multi-period planning horizon is considered where each period has a different product mix and demand necessities. Therefore, the formed cells in a period may not be optimal for the next period. To address this production environment, two approaches could be implemented. One is the concept of virtual cells, initially introduced by (McLean, Bloom, & Hopp, 1982), which allows exploiting the processing of part families into machines grouped virtually. Rapid advances in information technologies have nowadays driven traditional manufacturing all the way to intelligent cloud-based design and manufacturing. The cloud-based design for cellular manufacturing can be referred to as a "*multiscale, uncertain*, and *dynamic* service-oriented network where a set of *CAD* parts, modeled by set of features, can be manufactured in intelligent virtual manufacturing cells under certain constraints" (Ostrosi & Fougères, 2018). Also, alternative approaches to manufacturing cell formation use such features as vectors of information transmission, and fuzziness to improve the problem formulation robustness (Mutel & Ostrosi, 2002).

Another approach is to develop extremely efficient algorithms that respond to a stable and robust demand and to implement them at any change of the production environment. The static Cell Formation Problem (CFP) is the first and foremost problem for this group of algorithms. In general, CFP requires the partitioning of parts and machines with the goal of establishing cells having maximal number of intra-cell operations and minimal number of inter-cell operations. The focus in our work is on this group of algorithms.

* Corresponding author.
E-mail addresses: milos.danilovic@fon.bg.ac.rs, danilovicm@fon.bg.ac.rs (M. Danilovic), oliver.ilic@fon.bg.ac.rs (O. Ilic).

https://doi.org/10.1016/j.eswa.2019.06.019
0957-4174/

The aim of this paper is to design a novel algorithm for CFP that meets the following requirements comparing the best-known algorithms from the aforementioned group of algorithms:

- Improved efficacy;
- Multi-criteria optimization;
- Extensibility;
- Modularity;
- Expert systems character of the approach.

1.1. Efficacy improvement

CFP is known as an NP-hard problem (Ballakur & Steudel, 1987), so the size of the feasible solution set (FSS) is the key parameter determining the efficacy of a procedure execution. Cardinality of the feasible solution set is given in (Wang, 1999):

$$\left(\left(\sum_{i=1}^{w}\frac{(-1)^{w-i}i^{r}}{i\,!\,(w-i)\,!}\right)*\left(\sum_{i=1}^{w}\frac{(-1)^{w-i}i^{q}}{i\,!\,(w-i)\,!}\right)\right),$$

where w is the number of cells, q is the number of machines and r is the number of parts. Considering that this number becomes computationally infeasible, even for small numbers of machines and parts, in recent years research has been more often directed toward heuristic search models.

Regarding some best-known combinatorial NP-hard problems, CFP shows significant differences in terms of possibilities for narrowing the FSS. For example, for the permutation flow shop problem, no reduction of FSS is possible based on the instance structure. One can always construct a schedule of a new job that would contradict previous conclusions. In contrast, the CFP implies a collection of various constraints that reduce FSS. If, for example, two machines are in a strong correlation, these machines will belong to the same cell independently of the operations on other machines. The common drawback of the significant number of published CFP algorithms is that the search is performed on the entire or insufficiently narrowed sollution set.

The strategy of the new approach is to take advantage of the FSS reductions and to propose an efficient method that directs the search towards the global optimum. The core of the approach is the definition of the reduced input metric, which replaces original input data.

1.2. Multi-criteria optimization

It is well known that the CFP is a multi-criteria problem (Ballakur & Steudel, 1987). In most of the papers that introduce algorithms from our target group of algorithms, a binary representation of CFP has been studied.

For the algorithm, presented in this paper, the binary matrix is one of possible different input parameters from the data store. This matrix defines the assignment of parts to machines. All other input parameters and constrains can be supplemented to the optimization process defining the multidimensional distances between entities. In each iteration, distances between entities are recalculated and used for grouping of machines into cells and parts into families. The reduction matrix, that we introduced can serve arbitrary complex multi-criteria problem models. It represents, the composition of cells in each iteration. Reduction matrix is recalculated in each iteration based on the updated values in cluster matrices. In the paper, two types of cluster matrices were used; the first defines the similarity, and the other the difference between entities. As a measure of similarity or difference, distance vectors containing all the multi-criteria requirements of the model can be used.

For the clarity of the presentation, without loss of generality, *grouping efficacy* was chosen as the objective function, and the binary incidence matrix is used in the examples. All operations are carried out in the same way as in cases of a complex problem model. Another important reason for choosing grouping efficacy is the possibility of an objective experimental evaluation of the proposed algorithm. The aim of the paper is to show, in the most objective way, the efficacy improvement of the proposed approach. All relevant CFP algorithms have grouping efficacy as an objective function.

1.3. Extensibility

The proposed set of reductions in the novel algorithm is not finite; one can define a new criterion for a more effective reduction. This criterion could be easily implemented as a separate procedure that is processed in the iterations of the hybrid algorithm.

1.4. Modularity

Each sub-procedure in the hybrid algorithm could be replaced. This is an important advantage that enables the improvement of the existing algorithms. For example, one of the best published algorithms (Brusco, 2015) uses Iterated Local Search (ILS) as the improvement heuristic. The obvious improvement of this algorithm is to implement ILS on the feasible set reduced by our hybrid algorithm.

1.5. Expert systems character of the approach

The structure of the novel hybrid algorithm emulates the decision-making ability of a human expert. Complex and important problems in cellular design are solved based on the rules for the narrowing of the feasible solution set. These rules represent the knowledge base which is extensible and modular. The hybrid algorithm as an inference engine applies the rules to the input data in the iterations to deduce new, reduced data as the input for the following iterations.

1.6. Results

The novel algorithm is compared with the best algorithms from the target group of algorithms on the referent set of test instances. All relevant CFP algorithms from that group have been tested on 35 instances which can be downloaded from http://mauricio.resende.info/data/cell-formation/. The size of the instances is specified as the number of machines and the number of parts. The instances range from dimension $5\times7-40\times100$ and comprise well-structured, as well as unstructured matrices. Due to the three-decade period of testing on these instances, the best-known results are also the optimal results for most of these instances. All examples in this paper refer to the instances from this list. It will be shown that on all relevant test instances from literature, the presented algorithm provides best-known results with significantly lower consumption of CPU time.

The remainder of this article is organized as follows. The Literature review of the related algorithms is given in Section 2. The definitions of the problem are presented in Section 3. The new hybrid algorithm is presented in Section 4. In Section 5, the sub-algorithms of the hybrid algorithm are described in detail. Section 6 shows the computational results, and Section 7 concludes the paper. In Appendix 1, a detailed example of the algorithm execution is presented.

2. State of the art

Many approaches have been developed to deal with the CFP. Most of them provide heuristic solutions and only a few exact

methods have been suggested. Since a novel algorithm is compared with the algorithms for the binary representation of CFP, researches addressing the CFP as a binary part-machine incidence matrix are reviewed in the following.

2.1. Exact methods

Elbenani & Ferland (2012) presented a mixed-integer linear programming model which maximizes the grouping efficacy. The procedure use, as the input parameter the predefined number of production cells. Therefore, the suggested approach cannot guarantee global optimality of the obtained solutions with respect to a variable number of production cells. For certain tested instances the computational times are long, or memory limitations are exceeded, and the optimal solutions cannot be found. (Brusco, 2015a) presented two approaches for solving the CFP with the grouping efficacy objective. The first is a mixed-integer linear programming model which is based on a general two-mode clustering formulation. The simplifying assumption is that the numbers of clusters by rows and columns are equal. In the second, branch-and-bound algorithm, a relocation heuristic is used to obtain an initial solution. The branch and bound approach could solve more problem instances regarding the first approach and the computational times are improved as well. Pinheiro et al. (2016) reduced the CFP to a bi-cluster graph editing problem and suggested an exact method and a linear programming model which provides good computational results for the grouping efficacy objective. Bychkov & Batsyn (2018) presented the mixed-integer linear programming model. They use machine-machine and part-machine assignments instead of the widely used machine-part-cell assignment. This leads to a formulation considering only constraints which ensure a block-diagonal structure of solutions. It allows them to reduce the number of variables and constraints in their programming model and obtain optimal solutions for some large-sized problem instances.

Exact algorithms for CFP are important as the reference for the optimality of the heuristic algorithms for the same problem. In the experimental evaluation of the new algorithm, the CPU times for algorithms Elbenani & Ferland (2012) and Pinheiro et al., (2016) are included, as a reference. Exact algorithms cannot be used in the dynamic environment, due to the enormous processing times for even middle-sized input instances. For example, for the most efficient among these algorithms, Bychkov & Batsyn (2018), reported processing time on the instance No18 is 32,243.10 s, while for the instances No27, No29, No32 and No33 no results have been obtained. Another interesting observation regarding these algorithms is that on some of the small sized instances, reported CPU times are greater than the times spent when the total search is performed.

2.2. Heuristic algorithms

Meta-heuristics algorithms have been adopted to solve the CFP, producing the best results obtained so far. Since the list of these algorithms is long, here we briefly review the algorithms that have been reported the best-known results.

Onwubolu & Mutingi (2001) developed a genetic algorithm to simultaneously group machines and part families into cells. The designer can specify the number of cells, lower and upper bounds on cell size in advance. Goncalves & Resende (2004) used a local search algorithm coupled with the genetic algorithm to improve the CFP's solution in several instances. An algorithm based on evolution strategy has been developed by Stawowy (2006) which uses a modified permutation with a separator encoding scheme and a unique concept of separators' movement during the mutation process. James, Brown, & Keeling (2007) utilized a hybrid grouping genetic algorithm that combines a local search with a standard grouping genetic algorithm. The local search first analyzes the assignment of one part to a group of machines, then analyzes the assignment of a machine to a family of parts. At the time of publishing, reported results were among the best results so far. Wu, Chang, & Chung (2008) considered a simple effective simulated annealing-based approach. They proposed a local optimization, like the generation of neighboring solutions used by the tabu-search method. They use two kinds of moves; single-move and exchange-move to revise the neighborhood of the current solution. Unler & Gungo (2009) developed a K-harmonic means clustering algorithm to solve the CFP. Tariq, Hussain, & Ghafoor (2009) developed an approach that combines a local search heuristic with genetic algorithm. Mahdavi, Paydar, Solimanpur, & Heidarzade (2009) developed a mathematical model for the CFP based on cell utilization concept. Algorithm based on genetic algorithm was designed to solve developed mathematical model. A hybrid methodology based on using Boltzmann function of simulated annealing and mutation operator of GA was proposed by Wu et al. (2009) to optimize the initial clustering obtained from similarity coefficient method. Tunnukij & Hicks (2009) presents the Enhanced Grouping Genetic Algorithm that replaces the replacement heuristic in a standard Grouping Genetic Algorithm with a Greedy Heuristic and employs a rank-based roulette-elitist strategy, as a new mechanism for creating successive generations. Wu, Chung, & Chang (2010) adopted the water flow-like algorithm and designed a heuristic algorithm for solving the CFP. Noktehdan, Karimi, & Husseinzadeh Kashan (2010) used a grouping and hybridized version of differential evolution algorithm together with a local search algorithm to solve the CFP Li, Baki, & Aneja (2010) used a Max-Min ant colony optimization meta-heuristic to solve the machine-part CFP, implemented in the hyper-cube framework. Pailla, Trindade, Parada, & Ochi (2010) used an evolutionary algorithm and a local search around some of the solutions it visits. In addition, an approach based on simulated annealing was implemented that uses the same representation scheme of a feasible solution. Durań, Rodriguez, & Consalter (2010) used a modified particle swarm optimization algorithm for clustering problems. Arkat, Hosseini, & Farahani (2011) presented a bi-objective programming model to simultaneously minimize the number of exceptional elements and the number of voids in the part machine incidence matrix. Because of the NP-hardness of the model, they have also developed a bi-objective genetic algorithm for large-scale problems. Feng & Pheng (2011) proposed an exact schema theorem that can predict the expected number of copies of schemas in the next GA generation which applied for machine cell formation. Ying, Lin, & Lu (2011) developed a simulated annealing algorithm with variable neighborhood search for CFP empirically evaluated in terms of grouping efficacy. Dıaz, Luna, & Luna (2012) proposed a GRASP heuristic to obtain lower bounds for the optimal solution of the problem. They used a greedy randomized adaptive search procedure that consists of two phases. In the first phase an initial partition of machines into machine-cells or parts into part families is obtained, while in the second phase the assignment of parts to machine cell or machines to part-families is considered. Brusco (2015), applies a generally known repetition of a two-steps process: (i) perturbing an incumbent locally and (ii) applying a local – search heuristic to return the perturbed solution to a local optimum. In the source code of the algorithm there is a limitation that the number of clusters can't be above 20. Parameter k defines the number of iterations, regardless of the specificity of the instance proceeded. The tests were performed with three values for k: 10^3, 10^4 and 10^5. Experimental results for these three values show an unfavorable trend: larger instances require larger values of k. Best results for 35 instances were obtained with $k = 10^4$ and

Table 1
A brief review of the algorithms in the related literature.

No.	Method	Algorithm	Source	Measure	Test
1	GA	Genetic	Onwubolu & Mutingi (2001)	GE	*
2	EA	Evolutionary	Goncalves & Resende (2004)	GE	*
3	ES	Evolutionary strategy	Stawowy (2006)	GE	*
4	HGGA	Hybrid grouping genetic	James et al. (2007)	GE	*
5	SACF	Simulated annealing	Wu et al. (2008)	GE	*
6	KHC	K-harmonic means clustering	Unler & Gungo (2009)	GE	
7	HGA	Hybrid genetic	Tariq et al. (2009)	GE	*
8	GAA	Genetic approach	Mahdavi et al. (2009)	EX and V	*
9	HHA	Hybrid heuristic	Wu et al. (2009)	GE	*
11	EnGGA	Enhanced grouping genetic	Tunnukij & Hicks (2009)	GE	*
12	WFA	Water flow-like	Wu et al. (2010)	GE	*
13	HGDE	Differential evolution	Noktehdan et al. (2010)	GE	*
14	ACO	Ant colony optimization	Li et al. (2010)	GE	*
15	SA	Simulated annealing	Pailla et al. (2010)	GE	*
16	PSO	Particle swarm opt.	Durań et al. (2010)	GE	
17	BGA	Bi-objective genetic alg.	Arkat et al. (2011)	EX and V	
18	EST	Genetic	Feng & Pheng (2011)	–	
19	SAYLL	Simulated annealing with variable neighborhood	Ying et al. (2011)	GE	*
20	GRASP	GRASP heuristic	Dıaz et al. (2012)	GE	*
21	GAVNS	Genetic with variable neighborhood	Paydar & Saidi-Mehrabad (2013)	GE	*
22	ILS	Iterated Local Search	Brusco (2015)	GE	*
23	CFPAS	Hybrid VN Descent	Martinsa et al. (2015)	GE	*
24	GLCA	League Championship	Noktehdan et al. (2016)	GE	*

$k = 10^5$. Obtained results categorize this algorithm as the best algorithm so far. Martinsa, Pinheiroa, Prottia, & Ochi (2015) developed an algorithm 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 a local search phase. Noktehdan, Seyedhosseini, & Saidi-Mehrabad (2016) proposed a grouping version of the league championship algorithm. Their algorithm can reach the best-known solution for 29 of the 35 referent instances.

In Table 1, a brief review of the most important heuristics in the related literature are presented. This table shows the solution method and the applied measure (GE denotes grouping efficacy, EXV denotes optimization of the number of exceptions and voids). Asterix in the column Test denotes that the results of heuristic are presented and compared in the experimental evaluation of the new algorithm.

2.3. Discussion

Regarding the modularity and extensibility, all algorithms have their logic drowned in the iteration process. For example, one of the best algorithms so far (Brusco, 2015), applies a generally known iterated local search heuristic. Calculations of the values of the objective function are an integral part of the iteration process and cannot be replaced or extracted. In the similar way, crossover operators are the core of the evolutionary algorithms and any change of these operators implies the change of the overall algorithm as well.

Multi-criteria optimization in the reviewed heuristics can be performed only for the concrete model that is defined in advance. An algorithm is designed for the specific set of criteria in advance and this set cannot be changed. A novel approach presented in this paper can implement any criteria for any arbitrary model. The values of the objective function through the iterations are obtained using the reduction matrix, the distances between entities are tracked through the clustering matrices and the constrains are handled through the definition of the forbidden assignments.

A general conclusion can be drawn that in all reviewed algorithms, the specifies of the input instances are not sufficiently used. The logical consequence is, that all these algorithms would be more efficient when processed on the reduced instance.

3. Problem formulation

The task of the CFP is to design the cellular manufacturing system to process similar parts within the same production cell, minimize parts movement from one cell to another, and balance machines workload during the production process. In almost all studies reviewed in the State-of-the-art Section, *grouping efficacy* has been accepted as the measure of the objective function:

$$\eta = \frac{n-e}{n+v} = \frac{r}{n+v}, \tag{1}$$

where n is the total number of operations (ones in the machine-part matrix), e is the number of *exceptions* (ones outside the cells), v is the number of *voids* (zeros within cells), and r is the number of operations within cells (ones within cells).

The CFP is a multi-criteria problem and the binary incidence matrix is a simplified presentation of the model. Eq. (1) is chosen as the objective function for two reasons:

- Clarity of the explanation of procedures and examples: without loss of generality; in the next Section, the multi-criteria character of the approach is elaborated;
- Objectivity of the evaluation, since all the referent algorithms use the same objective function.

Regarding the minimal size of a cell, two different structures are: cells containing only machines or parts (*residuals*) and cells with one machine and several parts or vice versa (*singletons*). Although some studies don't allow the existence of residuals, considerations in this paper include residuals for two reasons: (1) restriction on residuals artificially narrows FSS, so it represents a simpler version of the problem, discussed in this paper; all presented statements and procedures concerning FSS reduction are also valid for this case; (2) from the point of practical applications, it is unjustified to insert a machine or a part into the cell if this insertion decreases the value of the objective function.

Due to the duality of features, for the clarity of explanation, machines and parts are denoted as *entities, en*. Number of entities (machines or parts) is denoted as N. When referenced separately, machines are denoted as m, while parts are denoted as p. For each entity, define the *neighbors* as the set of entities of another kind, having common operations with this entity (for example, machine neighbors are all parts processed on that machine…). Superscript M refers to machines, while P refers to parts.

One of the main tasks in the process of designing algorithm for CFP is to define a measure that quantifies the similarity (or dissimilarity) between entities. There are only two ways to do that. First is to explicitly define the measure and second is to implicitly implement this measure into the processing logic. The advantage of explicitly defined measure is that it can be replaced and adapted to fit the considered model. In this paper, for the clarity of explanation, Jaccard's similarity measure, $s_{i,j} = s(i, j)$ is used:

$$S = \|s(i,j)\|\ ;\ 1 \le i \le N;\ 1 \le j \le N,\ \text{where}\, s(i,j) = \frac{q(i,j)}{q(i,j)+g(i,j)};\ 0 \le s(i,j) \le 1. \quad (2)$$

The variable $q(i, j)$represents the number of common neighbors for entities i and j, while $g(i, j)$ is the number of distinct neighbors for these entities. Since this measure is used in the iterations of the new algorithm only to produce current best options for grouping, this measure can be replaced with any other, even multidimensional measure.

The matrix $T = \|t(i, j)\|$, $1 \le i \le N$; $1 \le j \le N$is additionally used as a measure of the dissimilarity between entities, where:

$$t(i,j) = q(i,j) - g(i,j). \quad (3)$$

The rest of the notation is:

$\rho = \frac{n}{N^M N^P}$ – *density* of operations,
d_l– *degree of* l; the number of neighbors of entity l,
$\bar{d}_l = \frac{d_l}{N}$ – *average degree* of entity l,
$C = \{c_1, \ldots, c_k, \ldots, c_w\}$– set of cells; w is the number of cells,
n_k - number of entities in cell k,
d^C – total dimension of cells, $d^C = d_1^C + \ldots + d_k^C + \ldots + d_w^C = \sum_{k=1}^{w} (n_k^M \cdot n_k^P)$,
e_l – number of exceptions formed by entity l,
v_l – number of voids formed by entity l.

Therefore, the grouping efficacy can be expressed as:

$$\eta = \frac{r}{n + d^C - r}. \quad (4)$$

4. The novel hybrid algorithm

The new algorithm, Cell Formation Optimization algorithm, CFOPT, is presented as the block diagram on Fig. 1.

At the beginning of the execution, the possible exact reductions are performed, and the global parameters of the obtained reduced

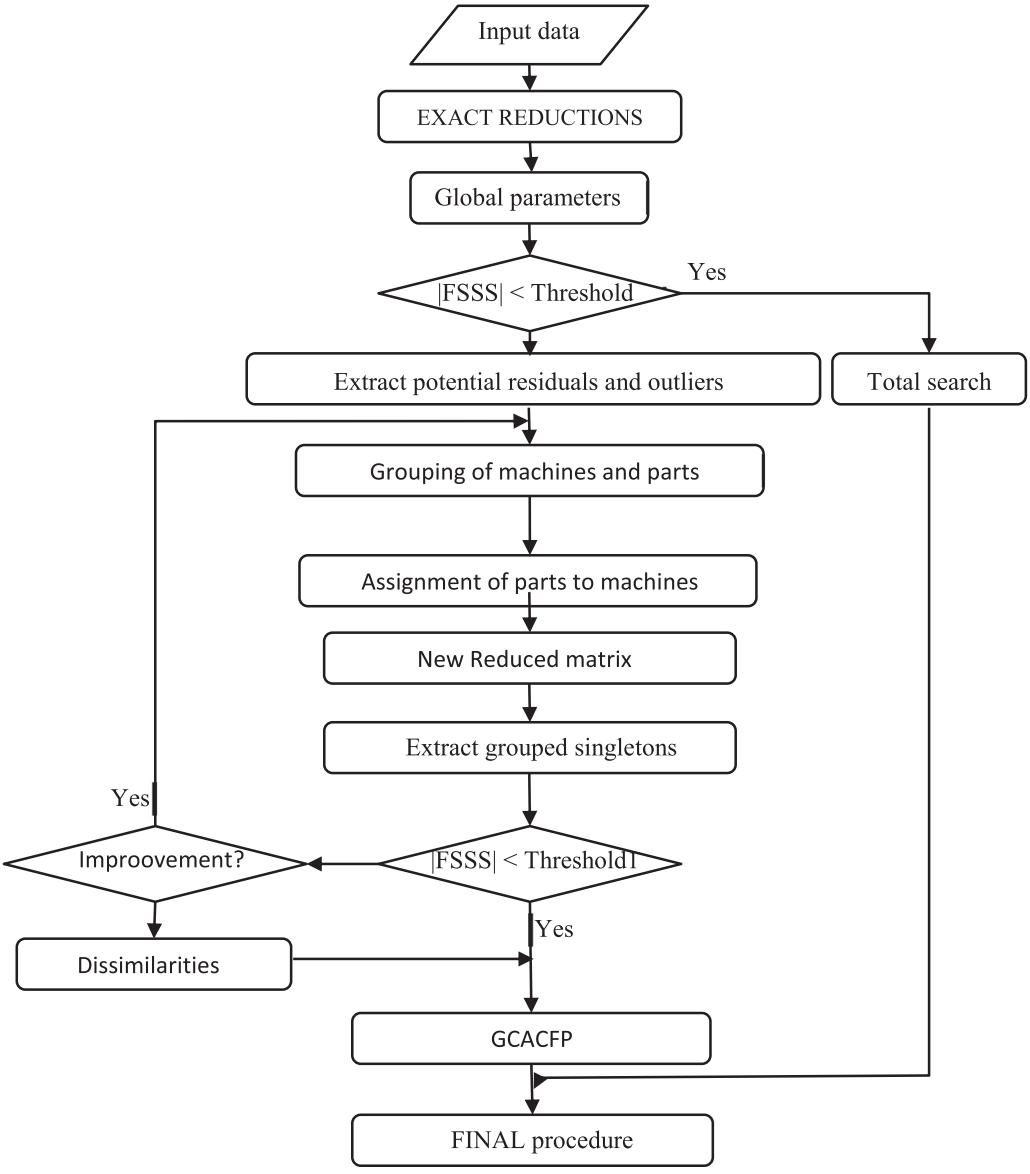


Fig. 1. Block diagram of CFOPT.

set are calculated. Depending on the values of the global parameters, the total search is performed, or the procedure continues. Prior to the iteration loop of the algorithm, potential residuals and outliers are removed from the process.

In the main loop, similarity matrices for machines and parts are calculated. The entities having maximal similarity value form the group of machines and the family of parts. Assignment of such grouped parts to machines is obtained and new cells are formed. If the cardinality of the reduced feasible set is below the defined threshold, the improvement heuristic is applied, and the final grouping of the original machines and parts are obtained using the procedure FINAL.

In the case that the cardinality of the FSS is above the predefined threshold, the iteration is repeated. If no improvement in the value of the objective function is achieved, restricted pairs for grouping are obtained using dissimilarity matrices. Using these restricted groupings, the improvement heuristic GCACFP obtains the final formation of cells.

5. Sub-algorithms in CFOPT

In this Section the definitions of the sub-algorithms in CFOPT are presented. Examples on the instances from the referent set of the test instances are used to clarify the procedures. In the Appendix 1, the example of all CFOPT steps is attached.

5.1. Handling input data

One of the main characteristics of the novel algorithm is that it can be applied to solve the multi-criteria CFP problems. The required part of the input data is the data that represents operations of parts on machines. Any form of this information can be handled; for the clarity of explanation, without loss of generality, hereinafter the incidence matrix will be used. Optionally, additional data, representing other parameters and relations for the specific model, could be supplemented. This data is used in the calculations of the distances between entities, through the iterations of the algorithm. The definition of the similarity matrices, Eq. (2), could be arbitrary changed in the way that it represents a multidimensional distance between entities. In each iteration, the entities having minimal mutual distance are grouped. In that way, grouping is performed based on the current model of the production environment. Constraints from the model could be handled through the part of the algorithm that tracks the forbidden groupings.

The main issue in the proposed approach is to find the effective way to calculate the value of the objective function on the reduced problem through the iterations of the algorithm. For this purpose, the *reduction matrix*, denoted as B, is introduced, together with two lists of joined entities, E (E^M and E^P). The list E consists of y sets of joined entities, where i th set has x_i entities:

$E=\{en_1=\{en_1(1),..., en_1(x_1)\},..., en_y=\{en_y(1),..., en_y(x_y)\}\}$. Thus, $x_1+x_2+...+x_y=n$ and the dimension of B is $y^M \times y^P$. The entries of matrix $B=\|b(i, j)\|$ are equal to the sum of the corresponding joined entries of matrix A.

As an example, Table 2 shows the matrix A, relevant to the problem **No.4** (Kusiak & Cho, 1992) from the reference list of 35 instances.

Let $E^M=\{\{4,6\},\{1\},\{2\},\{3\},\{5\}\}$ and $E^P=\{\{1,5\},\{3,6,8\},\{2\},\{4\},\{7\}\}$. Matrix B for this example is presented in Table 3. The first row represents joined machines 4 and 6, so, for example, number 2 in the first row and column 7 is obtained as a sum $a_{4,7}+a_{6,7}$ from the matrix A.

Additionally, the matrix $H=\|h(i, j)\|$ is introduced, having same dimensions as B, where $h(i, j)=x_i^M \cdot x_j^P$. In other words, the entry of H represents the dimension of the corresponding joined cell.

Table 2
Matrix A for the instance No.4.

M/P	1	2	3	4	5	6	7	8
1	0	1	0	1	0	0	1	0
2	1	1	1	0	1	1	1	1
3	0	0	1	0	0	1	0	1
4	0	0	0	1	0	0	1	0
5	1	0	1	0	1	1	0	1
6	0	0	0	1	0	0	1	0

Table 3
Matrix B for the instance No.4.

	1,5	3,6,8	2	4	7
4,6	0	0	0	2	2
1	0	0	1	1	1
2	2	3	1	0	1
3	0	3	0	0	0
5	2	3	0	0	0

Statement 1. If $b(i, j)$ belongs to any block, it participates to the overall value of v with $h(i, j)-b(i, j)$, otherwise it participates to the overall value of e with $b(i, j)$.

Proof. The entry $b(i, j)$ is obtained as the sum:

$$a(en_i^M(1), en_j^P(1))+a(en_i^M(1), en_j^P(2))+...+a(en_i^M(1), en_j^P(x_j^P))+$$
$$a(en_i^M(2), en_j^P(1))+a(en_i^M(2), en_j^P(2))+...+a(en_i^M(2), en_j^P(x_j^P))+$$
$$-----------------------------$$
$$a(en_i^M(i), en_j^P(1))+a(en_i^M(i), en_j^P(2))+...+a(en_i^M(i), en_j^P(x_j^P))$$

□

Each summand has value one or value zero, so $b(i, j)$ is equal to the number of ones in the corresponding cell. As the number of summands is $x_i^M \cdot x_j^P = h(i, j)$, the number of zeros in the corresponding cell is $h(i, j)-b(i, j)$.

Corollary. *The grouping efficacy for the original instance can be directly obtained from B and H.*

Therefore, complete optimization of the original problem can be carried out as the optimization of the reduced problem; the information concerning reduction, embedded in E, should be used only at the end of the process to obtain a final partition.

Entries in similarity matrices S and T, defined in (2) and (3) are functions of common neighbors, $q(i, j)$ and distinct neighbors, $g(i, j)$. In the reduced problem, determination of common and distinct neighbors for any pair of reduced entities i and j is simple. Let l be an entity of another kind, concerning i and j. Denote as b_1 the entry from B that corresponds to i and l, and denote as b_2 the entry from B that corresponds to j and l. If entity l participates in the value $q(i, j)$with Q and l participates in the value $g(i, j)$ with G, two cases can occur: When $h_1+h_2 \geq 2(b_1+b_2)$ then $Q=0$ and $G=b_1+b_2$. Otherwise, $Q=\min(b_1,b_2)$ and $G=h_1+h_2-b_1-b_2$.

Due to the above relations, complete optimization could be implemented using B and H instead of A. Reduction is a polynomial procedure consuming negligible CPU time, so overall time efficacy is directly increased by the degree of reduction.

5.2. Exact reductions

First, exact reduction removes *outliers* and residuals from consideration. The outlier is an entity having all entities of the other kind as neighbors. For machines, the condition is $d_i^M = N^P$ and for parts $d_j^P = N^M$. Next, *identities*, i.e. entities having $s(i, j)=1$ are grouped together. Finally, exact reduction removes well-structured

Table 4
Matrix *B* for the instance No.22.

	1	2	3	4	5	6	7
1	20	0	0	0	0	0	0
2	0	0	0	0	0	0	24
3	0	16	0	0	0	0	0
4	0	0	0	0	0	14	0
5	0	0	0	30	0	0	0
6	0	0	12	0	0	0	0
7	0	0	0	0	15	0	0

parts of the input instance, denoted as *exact singletons*. In the matrices *A* or *B*, exact singleton is an entry greater than zero having zeros in all other entries of the corresponding row and column. It is clear, without proving, that exact singleton is a cell in the optimal solution of the original problem.

Exact reductions are straightforward and represents no novelty in the CFP algorithms. However, missing to implement them in the real production problems is impractical. This conclusion can be presented on the instance **No.22** (Chandrasekharan & Rajagopalan, 1989). This is a 24×40 instance, having density 13.65%. The instance has no exact outliers or residuals. Reduction of identities results in a 7×7 matrix *B*, presented in Table 4. This is a well-structured matrix with seven exact singletons, and after the reduction of these singletons, the exact final solution is obtained. Thus, the optimal solution is obtained in a few microseconds of CPU consumption. By comparison, the exact algorithm CPLEX (Elbenani & Ferland, 2012) obtained the same result, in 108.58 s, while the best reported heuristics spent few seconds for the same result. It must be noted that these heuristics have no confirmation that the obtained result is optimal.

Exact reductions have the main purpose to prepare the matrix *B* for heuristic reductions, presented in next Subsection. In some examples with ill-structured instances, matrix *B* can be identical to the initial matrix *A*, but the heuristic reductions just may benefit from those specificities of this ill-structure.

Exact reductions assume the existence of special procedures that transforms the solution of the reduced problem to a solution of the original problem. These procedures must satisfy two important requirements:

1. If the reduced solution is optimal for the reduced problem, then the final solution, obtained by this procedure, is the optimal solution for the original problem;
2. If two entities belong to the same cell in the reduced problem, corresponding joined entities from the original problem must be in the same cell in the optimal solution of the original problem, no matter whether the reduced solution is optimal or not. Hereinafter these procedures are referred as *final procedures* and are performed by procedure FINAL.

5.3. Global parameters

Global parameters define the criteria for the directions of the algorithm execution. According to the first set of global parameters and the value of the Threshold, defined in advance, the decision is made if the Total search should be performed. Through the execution of the iterations Threshold1 defines the criteria when to implement the improvement heuristic.

5.3.1. Total search

The most important information, in the process of constructing a combinatorial optimization algorithm, is the dependence between the size of the FSS and the instance size. The lack of this information has as the consequence that, in some step of a program execution, longer processing is performed instead of performing the total search on the unexplored part of the FSS. In almost all papers related to the CFP, regarding certain instances, the proposed algorithms obtain worse results with a longer working time in comparison with a simple total search algorithm.

The formation of machine cells or part families is a well-known problem of set partitioning. The *partitions of a set* are the ways to regard that set as a union of nonempty, disjoint subsets called *blocks* (Knuth, 2011). For example, the five essentially different partitions of $S_3 = \{1, 2, 3\}$ can be written in the form: $\Pi(S_3) = 123$, 12|3, 13|2, 1|23, 1|2|3, using a vertical line to separate one block from another. In this list the elements of each block could have been written in any order, and so could the blocks themselves, because 13|2 and 31|2 and 2|13 and 2|31 all represent the same partition. In the previous example we have standardized the representation by agreeing to list the elements of each block in increasing order, and to arrange the blocks in increasing order of their smallest elements. With this convention, a simple generator of all partitions $\Pi(S_k)$ from a given partitions of $\Pi(S_{k-1})$ can be formulated as PARTGEN (Algorithm 1):

Algorithm 1 PARTGEN.

```
1: procedure PARTGEN(Π(S_{k-1}))
2:   For each of the partitions par in Π(S_{k-1})
3:     Add k as the last element of the blocks, once at a time;
4:     Generate a new partition "par|k"
5:   Next
6: end
```

Using PARTGEN we can iteratively construct the partition of any given set:

$\Pi(S_1) : -1$

$\Pi(S_2) : -12\ 1|2$

$\Pi(S_3) : -123\ 12|3\ 13|2\ 1|23\ 1|2|3$

$\Pi(S_4) : -1234\ 123|4\ 124|3\ 12|34\ 12|3|4\ 134|2\ 13|24\ 13|2|4$
$14|23\ 1|234\ 1|23|4\ 14|2|3\ 1|24|3\ 1|2|34\ 1|2|3|4$

According to this simple generator, a triangle matrix $K = \|k_{i,j}\|$ can be constructed, where $k_{i,j}$ stores the number of partitions of the set with i elements having exactly j blocks. Entries $k_{i,j}$ are obtained using simple relations: $k_{i,1} = k_{i,i} = 1$; $k_{i,j} = k_{i-1,j-1} + j \cdot k_{i-1,j}$. Matrix K, for the sets, having up to 15 elements is presented in Table 5.

Any limitation on the partition composition lowers the table entries. For example, if the minimal number of elements in a block is two, $k_{6,3}$becomes 15 instead 90. Data from Table 3, as well as the corresponding partitions, can be stored in a database, so their evaluation does not overload the execution time of the algorithm. At a first glance, the total number of combinations is calculated by multiplying the corresponding element of the matrix for machines with the corresponding element of the matrix for parts. For the example with 6 machines, 10 parts and 3 cells, 90 is multiplied with 9330. However, due to the correlation between the partitioning of parts and machines, the considered number of combinations is only 90 (the number of machines is lower than the number of parts). Nevertheless, as 90 * 9330 = 839,700; the total search can be processed, almost instantly for this example.

5.3.2. Threshold 1

Allowed number of cells is calculated based on simple following considerations regarding dimensions of cells:

When all entities are in one cell, there are no exceptions, $r = n$ and the grouping efficacy is:

$$\eta = \frac{n}{n+v} = \frac{n}{n + N^M N^P - n} = \rho.$$

Table 5
Number of partitions for a given cardinality and number of blocks.

	1	2	3	4	5	6	7	8	9	10
1	1									
2	1	1								
3	1	3	1							
4	1	7	6	1						
5	1	15	25	10	1					
6	1	31	90	65	15	1				
7	1	63	301	350	140	21	1			
8	1	127	966	1701	1050	266	28	1		
9	1	255	3025	7770	6951	2646	462	36	1	
10	1	511	9330	34,105	42,525	22,827	5880	750	45	1
11	1	1023	28,501	145,750	246,730	179,487	63,987	11,880	1155	55
12	1	2047	86,526	611,501	1379,400	1323,652	627,396	159,027	22,275	1705
13	1	4095	261,625	2532,530	7508,501	9321,312	5715,424	1899,612	359,502	39,325
14	1	8191	788,970	10,391,745	40,075,035	63,436,373	49,329,280	20,912,320	5135,130	752,752
15	1	16,383	2375,101	42,355,950	210,766,920	420,693,273	408,741,333	216,627,840	67,128,490	12,662,650

Construction of cells makes sense only if $\eta > \rho$. The density of operations determines allowed values for d^C. If the lower bound for efficacy is $\hat{\eta}$, Eq. (4) implies that the bounds for d^C are:

$$\rho < \hat{\eta} \le \frac{n-e}{d^C+e} \le 1 \quad \Rightarrow \quad n-2e \le d^C \le \frac{n-(\hat{\eta}+1)\cdot e}{\hat{\eta}}. \tag{5}$$

Thus, for example, when $\hat{\eta}=0.5$, then d^C can have only $n-e$ different values. This limitation further reduces the set of possible values for the dimensions of cells. If singletons and residuals are removed from calculations, the smallest dimension of a cell is 2×2. In this case, the maximum number of cells is $w=\lfloor\frac{min(N^M,N^P)}{2}\rfloor$. As, for a given number w of cells, maximal d^C implies (w-1) cells having dimension 2×2, and minimal d^C implies cells having equal dimensions, then:

$$\frac{N^M N^P}{w} \le d^c \le 4(w-1)+\left[N^M-2(w-1)\right]\left[N^P-2(w-1)\right]. \tag{6}$$

The similar conclusions can be derived in the situations when singletons are considered. In the paper Chandrasekharan & Rajagopalan (1986a), the upper limit for the number of independent cells, when singletons are considered is:

$$w \le 1+\left\lfloor \frac{(N^M+N^P-1)-\sqrt{\left[(N^M+N^P-1)^2-4(N^MN^P-n)\right]}}{2} \right\rfloor$$

For example, when $\rho > 0.5$ and when lower degree residuals are removed, a cell formation procedure may produce maximally three cells and the most probable number of cells is two. This further implies simple, efficient CFP procedure for instances having $\rho > 0.5$. Instances **No.2, No.3, No.4, No.9** and **No.34** have $\rho > 0.5$ and optimal partitions into two cells can be easily determined for each of them.

5.4. Extraction of potential residuals and outliers

Heuristic reductions use similar parameters like exact reductions, with *potential* as the prefix: *potential outliers* and *potential residuals*.

The existence of outlier machines is a signal to a manager to consider a duplication of such machines. Potential outlier is entity l having density of operations ρ_l close to one. In the experimental evaluation it is considered that the potential outliers are entities having highest value of density greater than 0.85. Such entity is removed and after the completion of algorithm processing, potential outlier is added to the most suitable cell using procedure FINAL.

Potential residuals are entities having weak correlations to other entities. The idea is to remove them from processing, thus reducing the size of the problem, and to insert them into the most suitable cell, or leave them outside of cells after the completion of the algorithm. Selection of potential residuals at the beginning of the processing is not critical because their informational value in the cell formation procedure is small. Smaller values $s_l=\sum_{j=1}^{ne} s(l,j)$ and $\bar{d}_l$ imply greater possibility that l is residual. Selection of potential residuals is performed through the simple procedure: Sort s_l in ascending order; Define the first position *pos* in this sorted list when $d_l > \lfloor\frac{\bar{d}_l}{2}\rfloor$. Potential residuals are entities that correspond to the first (*pos* – 1) elements in the sorted list.

Additionally, all entities having $d_l=1$ are selected as potential residuals. The reason for this selection is the measure of contribution of e and v to the efficacy value, defined in Statement 2. This statement defines the sufficient condition for efficacy improvement when interiors r and voids v are changed by Δr and Δv.

Statement 2. The efficacy η is improved when:

1. $\frac{\Delta r}{\Delta v} \ge \eta$ when $\Delta r > 0 \wedge \Delta v > 0$
2. $\frac{\Delta r}{\Delta v} < \eta$ when $\Delta r \le 0 \wedge \Delta v < 0$

Proof: $\frac{r+\Delta r}{n+v+\Delta v} \ge \frac{r}{n+v} = \eta \quad \Rightarrow \quad \Delta r\cdot(n+v) \ge \Delta v\cdot o \quad \Rightarrow \quad \Delta r \ge \eta\cdot\Delta v.$ □

In the trivial case when $\Delta v=0$, any $\Delta r > 0$ improves the efficacy. This completes the proof. An important conclusion is that the ratio $\frac{\Delta r}{\Delta v}$ establishes the measure of improvement quality.

Suppose, without loss of generality, that the considered entity, having degree one, is a machine. If this machine belongs to a cell having x parts, it participates to the overall efficacy value with one interior and (x – 1) voids. If this machine is residual, it participates to the overall efficacy value with one exception. Thus, the first option has $\Delta r=1$ and $\Delta v=x-1$ comparing the option when the considered machine is residual. The conclusion is that, when $\eta > 0.5 \Rightarrow x \le 2$ or when $\eta > 0.34 \Rightarrow x \le 3$. Thus, if the considered machine is not a residual, it could belong to a cell with maximum three parts for any reasonable efficacy value. Also, after the exact reductions are performed, two machines having degree one must process different parts. These conclusions allow a simple inclusion of these potential residuals to a final partition.

The proposed selections of residuals can be clarified on the examples with instances. **No.33** and **No.34**. The high-density instance **No.34** has 37 machines and 53 parts, and density 49.82%. Exact reductions reduce the problem size to 30×40. Five exact outliers (11, 14, 17, 21, 26) are removed. The sorted list s for machines is presented in Table 6.

Row *lr* shows indices of the reduced problem, while row l contains indices of the original problem. From the condition $\lfloor\frac{\bar{d}_l}{2}\rfloor = \lfloor\frac{n}{2\cdot N}\rfloor = \lfloor\frac{977}{2\cdot 37}\rfloor = 13$, first 8 entities are selected as potential

Table 6
Sorted list *s* for the instance No.34.

lr	13	30	28	4	31	7	23	20	19	11	29	8	14	26
l	12	36	34	2	37	5	29	25	24	9	35	6	13	32
s	1.38	2.45	4.16	5.48	5.93	5.95	6.66	7.51	8.00	8.17	8.38	8.75	8.77	8.79
d	4	4	5	8	10	11	9	11	19	17	20	15	19	22

Table 7a
Matrix *A* for the instance No.11.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		1								1	1	1			
2			1		1			1					1		1
3	1					1			1					1	
4	1			1					1					1	
5			1		1			1					1		1
6	1			1		1			1					1	
7		1					1			1	1	1			
8			1		1			1					1		1
9				1		1			1					1	
10		1					1			1	1	1			

residuals {12, 36, 34, 2, 37, 5, 29, 25}. All these potential residuals are also residuals in the optimal solution of the original problem.

The low-density instance **No.33** has 30 machines and 90 parts, and density 11.19%. Exact reductions reduce the problem size to 29×72. One exact singleton (18, 26) is a final cell. According to the sorted list, seven potential residual machines (1, 2, 6, 8, 10, 13, 23) are selected. Additionally, machine 18, having degree one, is selected. In the final optimal partition, machines 2, 6, 13 and 23 are the only machine residuals. Machines 8, 10 and 18 are in the one-machine - one-part cells, while machine 1 forms a singleton with three parts. For parts, potential residuals are obtained from 19 parts having degree one (1, 3, 6, 7, 10, 12, 17, 18, 20, 23, 24, 26, 32, 34, 38, 39, 44, 45, 53). Their removal did not affect the optimal partitioning of the reduced instance.

5.5. Grouping of machines and parts

The next reduction step is an iterative grouping of high ranked similar entities. Through the iterations, entities having current maximal $s_{i,j}$ are grouped. The procedure halts when current maximal $s_{i,j}$ falls below the limits, defined in Threshold1. Table 7a shows matrix *A* for **No.11** while Table 7b presents reduced *B* after exact reductions.

$E^M = \{\{2,5,8\},\{7,10\},\{1\},\{3\},\{4\},\{6\},\{9\}\}$ and
$E^P = \{\{2,10,11,12\},\{3,5,8,13,15\},\{9,14\},\{1\},\{4\},\{6\},\{7\}\}$.
Matrix *B*', obtained after the first iteration of grouping, is presented in Table 7c.
$E^M = \{\{3,4,6,9\},\{7,10,1\},\{2,5,8\}\}$ and
$E^P = \{\{1,4,6,9,14\},\{2,10,11,12\},\{3,5,8,13,15\},\{7\}\}$.
E^M is the optimal partitioning. Procedure ASSIGNMENT, presented in the next Subsection transforms, in one step, E^P to the optimal partition assigning {7} to {2,10,11,12}.

Table 7b
Matrix *B* for No.11.

	1	2	3	4	5	6	7
1		15					
2	8						2
3	4						
4			2	1		1	
5			2	1	1		
6			2	1	1	1	
7			2		1	1	

Table 7c
Matrix *B*' for No.11.

	1	2	3	4
1	17			
2		12		2
3			15	

5.6. Machine-part assignment

Exact reductions produce matrix *B* and Heuristic reductions produce matrix $\hat{B}$. Each row or column of $\hat{B}$ corresponds to one or more rows and columns in *B*. In that way, the initial assignment of entities to cells is obtained: *i*th row and *i*th column of $\hat{B}$ represents *i*th cell, so all entities from *B* that correspond to that row and that column are in the *i*th cell. In a total search for CF, as noted in Section 5.3.1, the number of combinations is calculated as a product of the corresponding elements of the matrix *K*, for machines and parts. This non-linearity reduces the size of instances that can be processed with a total search within a reasonable time. Fortunately, the specificity of the CFP makes it possible to partition machines based on the partition of parts and vice versa. In this case the total search should be processed through the smaller entity (machines or parts). In the following, three kinds of assignment are introduced: (i) machines to parts; (ii) cell successors for non-assigned entities and (iii) forbidden assignments. It can be assumed that, without loss of generality, the number of machines is lower than the number of parts.

5.6.1. Machine-part correlation

The correlation between machines and parts is one among few CFP specificities, which are discussed in literature. In the paper Li et al. (2010), an important Theorem is proved that, for a given machine partitions, each part *j* is optimally assigned to cell c_k if:

$$k = \arg\min\{e^P_{j,t} + \eta' v^P_{j,t}\},\ t = 1, \ldots, w. \tag{7}$$

where $e^P_{j,t}$ is the number of exceptions formed by assigning part *j* to cell *t*, and $v^P_{j,t}$ is the number of voids produced by assigning part *j* to cell *t*. In the papers dealing with a machine-part correlation, the rules for part assignments are defined for non-optimal machine assignments. Thus, η' is a current, not optimal value, which implies that the assignment of a part depends on the assignment of other parts.

The following Statement introduces a stronger condition for parts assignment.

Statement 3. For a given partition of machines, the sufficient condition that p_j is optimally assigned to cell c_k is:

$$r^P_{j,k} - v^P_{j,k} - e^P_{j,k} > 0. \quad (8)$$

This assignment does not depend on the assignment of other parts.

Proof. The worst case for assigning p_j to c_k is when an alternative cell can be formed that contains only the machines outside c_k that proccesses p_j. The efficacy in this case is $\eta = \frac{r+\Delta e}{n+v}$. Thus, the sufficient condition to assign p_j to c_k is $\frac{r+\Delta r}{n+v+\Delta v} \geq \frac{r+\Delta e}{n+v} = \eta \leq 1 \Rightarrow \Delta r \geq \Delta v + \Delta e$. This condition does not depend on η, so the assignment p_j to c_k does not depend on the assignment of the other parts. This completes the proof. The main advantage of the proposed condition is that the assignment does not depend on the current efficacy value. Thus, the assignment of a part to a cell does not change any of the previous part assignments. □

Statement 3 can be used to extend the proposed parts assignment to reassignment of machines and parts. For this purpose, the definition of *cell successors* is introduced.

Definition 1. A cell successor is a machine, or a part, satisfying $sw_{i,k} = r_{i,k} - v_{i,k} - e_{i,k} > 0$. The value $sw_{i,k}$ is a *successor's weight*. Cell c_k is a *dominator* for its successors.

5.6.2. Restrictions in cell assignment

The limit values of similarity are $s_{i,j} = 1$ (two identical rows or columns in matrix A) and $s_{i,j} = 0$ (logical operation OR between the binary representation of these two rows or columns returns the value FALSE). If two entities having $s_{i,j} = 0$ are in the same cell, all operations within cell of one entity are voids for another. Unlike large values of s, value $s_{i,j} = 0$ hides degrees of dissimilarities. Matrix T, defined in (3), represents an appropriate measure of dissimilarities. This measure can effectively be used to reduce the FSS in the following manner. Declare two entities, en_i and en_j, as *opposites*, denoted as $opos(en_i, en_j)$ if $t_{i,j}$ is below the defined limit. In that way, all opposites form a list of pairs of entities that belong to different cells which can't be merged before the final procedures. In our experimental research, the threshold for opposites is defined as an average value of all entries in T. This threshold was the right choice for all tested instances.

One can notice the convenience of opposites in the reduction process on the example with the largest instance **No.35** (40 × 100). After the exact reductions, the reduced instance has dimensions 39 × 59. After iterative heuristic groupings the reduced instance has dimensions 14 × 17 and matrix B' is presented in Table 8.

avg. : -42, Treshold: -42, number of pairs: 33

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
▶ 01	0	-88	-118	-52	-140	-58	-40	-49	-59	-66	-22	-23	-23	-22
02	-88		-120	-59	-126	-65	-45	-54	-66	-71	-24	-25	-28	-27
03	-118	-120		-80	-167	-83	-61	-70	-87	-94	-36	-34	-40	-36
04	-52	-59	-80		-78	-40	-26	-38	-41	-48	-2	-15	-15	-2
05	-140	-126	-167	-78		-84	-61	-70	-85	-95	-35	-36	-39	-35
06	-58	-65	-83	-40	-84		-31	-40	-47	-51	-16	-17	-2	-19
07	-40	-45	-61	-26	-61	-31		-28	-32	-35	-13	-2	-17	-16
08	-49	-54	-70	-38	-70	-40	-28		-41	-41	-15	-16	-22	-21
09	-59	-66	-87	-41	-85	-47	-32	-41		-55	-15	-16	-16	-18
10	-66	-71	-94	-48	-95	-51	-35	-41	-55		-16	-17	-20	-19
11	-22	-24	-36	-2	-35	-16	-13	-15	-15	-16		-11	-11	-4
12	-23	-25	-34	-15	-36	-17	-2	-16	-16	-17	-11		-15	-14
13	-23	-28	-40	-15	-39	-2	-17	-22	-16	-20	-11	-15		-17
14	-22	-27	-36	-2	-35	-19	-16	-21	-18	-19	-4	-14	-17	

Fig. 2. Matrix T^M for the instance No.35.

Corresponding partitioning is:
E^M = {{2,10,16,21,31},{1,3,7,32},{6,12,26,38,40},{18,33,34},
{5,8,22,23,37,39},{19,25,28},{14,35},{11,13},{4,9,20},
{24,27,29},{15},{17},{30},{36}} and
E^P = {{36,38,42,51,52,64,65,70,72,74,75,76,80,87},{45,67,71,91},
{6,15,16,24,27,60},{35,47,53,78,79,83,88,93},
{8,11,13,14,20,22,23,32},{39,73},
{10,18,29,33,34,37,44,49,50,54,55},
{63,66,68,69,82,84,85,89,90,92,94,96,97,98,99,100},
{1,2,3,4,5,7,21,25,28},{9,12,17,19,26,30,31,40,43,46},
{56,57,58,59,61,62},{41},{48},{77},{81},{86},{95}}.

The threshold for machines is −42 and the number of opposites is 33 (Fig. 2: yellow coloured entries on the upper or lower triangle of the matrix, together with the entry coloured in colour bisque, which represents the entry having minimal $t_{i,j}$).

For parts, the threshold is −29 and the number of opposites is 46. First ten grouped machines and first eleven grouped parts form clearly separated sub-matrices. From Fig. 3 it is clear that all merged parts, except grouped part 6 must belong to the different cells. Thus, all ten cells from the optimal partition are defined, and the final procedure, almost immediately finds the optimal solution. Instance **No.35** is the largest instance in the list (40 × 100) with density 10.5%. Thus, the optimal solution is reached in 0.09 s. By comparison, the exact algorithm CPLEX (Elbenani & Ferland, 2012), obtains the same result, in 1572,184.5 s, while the best reported heuristics spent about 300 s for the same result.

Table 8
Matrix B' for the instance No.35.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2			36										4		1	
2							40		1			3	1				
3	62			2			1	1							4	4	1
4		11				5		1									
5	1							88		1				1		1	5
6		1				1		1		1	18				1		
7			12			1			1								
8			2						18	1			2				
9					23			1									
10										30		1					
11		4											1				
12			4						1						1		
13		1		1	1						6						
14	1	3		1		2		1									

avg. : -29, Treshold: -29, number of pairs: 46

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
01	0	-52	-60	-86	-76	-31	-105	-159	-78	-95	-66	-21	-21	-22	-11	-11	-20
02	-52		-34	-48	-34	-2	-41	-57	-31	-35	-30	-13	-10	-14	-15	-15	-15
03	-60	-34		-56	-38	-16	-47	-67	-27	-39	-42	-11	-11	-12	-10	-13	-13
04	-86	-48	-56		-64	-27	-77	-101	-59	-67	-62	-17	-17	-6	-19	-16	-19
05	-76	-34	-38	-64		-17	-61	-85	-43	-51	-44	-9	-9	-10	-11	-11	-11
06	-31	-2	-16	-27	-17		-22	-34	-16	-18	-14	-11	-11	-12	-13	-13	-13
07	-105	-41	-47	-77	-61	-22		-120	-59	-74	-53	-4	-10	-14	-15	-15	-15
08	-159	-57	-67	-101	-85	-34	-120		-88	-103	-73	-23	-23	-21	-25	-22	-10
09	-78	-31	-27	-59	-43	-16	-59	-88		-51	-39	-9	-3	-10	-11	-11	-11
10	-95	-35	-39	-67	-51	-18	-74	-103	-51		-45	-7	-10	-11	-12	-12	-12
11	-66	-30	-42	-62	-44	-14	-53	-73	-39	-45		-8	-8	-9	-7	-10	-10
12	-21	-13	-11	-17	-9	-11	-4	-23	-9	-7	-8		-8	-9	-10	-10	-10
13	-21	-10	-11	-17	-9	-11	-10	-23	-3	-10	-8	-8		-9	-10	-10	-10
14	-22	-14	-12	-6	-10	-12	-14	-21	-10	-11	-9	-9	-9		-11	-11	-11
15	-11	-15	-10	-19	-11	-13	-15	-25	-11	-12	-7	-10	-10	-11		-2	-12
16	-11	-15	-13	-16	-11	-13	-15	-22	-11	-12	-10	-10	-10	-11	-2		-12
17	-20	-15	-13	-19	-11	-13	-15	-10	-11	-12	-10	-10	-10	-11	-12	-12	

Fig. 3. Matrix T^P for the instance No.35.

5.6.3. Procedure ASSIGNMENT

Sub-procedure ASSIGNMENT, summarized in Algorithm 2, forms cells, C, based on the grouped sets E^M, E^P and the matrix $\hat{B}$. As a side effect, a further reduction of FSS can be obtained. For the clarity of presentation, it is assumed that the number of grouped machines, y^M is less than the number of grouped parts, y^P.

Algorithm 2 ASSIGNMENT.

```
1: procedure ASSIGNMENT(B̂, E^M, E^P, y^M, y^P)
2:     i, k ← max(sw^P_{i,k} | sw^P_{i,k} > 0, i ≠ k);
3:     if (i is not null)
4:       for j = 1 to y^M
5:           q = b_{k,j}; b_{k,j} = b_{i,j}; b_{i,j} = q
6:       next j
7:       go to 2;
8:   end if
9:   i, k ← max (sw_{i,k} | sw_{i,k} > 0);
10:    if ((i is not null) and (sw_{i,k} ∈ sw^P)
11:      for j = 1 to y^M
12:          b_{j,k} = b_{j,k} + b_{j,i}; y^p = y^P − 1;
13:          concatenate (en^p_k, en^p_i) in E^P
14:      next j
15:      go to 11;
16:  else if ((i is not null)
17:    for j = 1 to y^P
18:          b_{k,j} = b_{k,j} + b_{i,j}; y^M = y^M − 1;
19:          concatenate (en^M_k, en^M_i) in E^M
20:      next j
21:    go to 11;
22:  end if;
23: end
```

First (lines 2–8), the group of parts having maximal successor weight, $sw^P_{i,k}$ is determined. If this weight has a positive value, columns i and k are switched in B. The procedure continues until all potential group parts are assigned to dominator cells.

Next, additional groupings based on successors weights are performed (lines 9–22). The groups of parts are concatenated in the loop 11–14 while the groups of machines are concatenated in the loop 17–20.

5.7. Improvement heuristic GCACPF

Procedure GCACFP, presented as Algorithm 3, is a constructive heuristic based on the GCA, proposed in Danilovic & Ilic (2016) which, using the local search, directs the solution to a global optimum. Due to the modular structure of the new algorithm, the choice of improvement heuristic can be adjusted to the production environment of the specific problem and the problem instances. In the new hybrid algorithm GCA is used for two reasons. First, the constructive character of this heuristic is ideal for a parallel programing and second, the forbidden groupings are easily implemented.

Algorithm 3 GCACFP.

```
procedure GCACFP(B̂, E^M, E^P, track)
f1  sorts cells in a descending order of their corresponding entries in B̂;
f2  elects the entity en from C^{W−k} having maximal sw; in the case of ties,
all entities having maximal sw are tracked if the number of such entities is
below the value track, otherwise, first track entities are tracked;
f3
1:    η_opt ← current optimal value of the objective function; c_opt ← null;
2:    for each c in C*
3:        for each enc in c if opos(enc, en) next c;
4:      insert en in c;
5:      B̂, E^M, E^P, y^M, y^P ← ASSIGNMENT * (B̂, E^M, E^P, y^M, y^P);
6:      if (η (B̂) > η_opt) then c_opt ← c; η_opt ← η (B̂);
7:    next c
8:    if not c_opt = null then insert en in c_opt;
end
```

Asterisk in ASSIGNMENT* denotes that only entity *en* and cell *c* are processed.

Constructive search techniques work by constructing a solution step by step, evaluating that solution for feasibility and objective function. Insertion based constructive heuristics typically have common structures, comprising of the *initialization* phase and the loop in the *insertion* phase.

For the partitioning problems, $\hat{B}$ holds data regarding the initial cell partition. In the initialization phase, the function $f_1(\hat{B})$ sorts the cells, according to their corresponding entries in $\hat{B}$. The

Table 9
Referent instances from literature.

No.	N^M	N^P	Source
1	5	7	King & Nakornchai (1982)
1a	5	7	King & Nakornchai (1982)
2	5	7	Waghodekar & Sahu (1984)
3	5	18	Seifoddini (1989)
4	6	8	Kusiak & Cho (1992)
5	7	11	Kusiak & Chow (1987)
6	7	11	Boctor (1991)
7	8	12	Seifoddini & Wolfe (1986)
8	8	20	Chandrasekharan & Rajagopalan (1986)
9	8	20	Chandrasekharan & Rajagopalan (1986a)
10	10	10	Mosier & Taube (1985a)
11	10	15	Chan & Milner (1982)
12	14	23	Askin & Subramanian (1987)
13	14	24	Stanfel (1985)
14	16	23	McCormick, Schweitzer, & White (1972)
15	16	30	Srinivasan, Narendran, & Mahadevan (1990)
16	16	43	King (1980)
17	18	24	Carrie (1973)
18	20	20	Mosier & Taube (1985b)
19	20	23	Kumar, Kusiak, & Vannelli (1986)
20	20	35	Carrie (1973)
21	20	35	Boe & Cheng (1991)
22	24	40	Chandrasekharan & Rajagopalan (1989)
23	24	40	Chandrasekharan & Rajagopalan (1989)
24	24	40	Chandrasekharan & Rajagopalan (1989)
25	24	40	Chandrasekharan & Rajagopalan (1989)
26	24	40	Chandrasekharan & Rajagopalan (1989)
27	23	40	Chandrasekharan & Rajagopalan (1989)
28	27	27	McCormick et al. (1972)
29	28	46	Carrie (1973)
30	30	41	Kumar & Vannelli (1987)
31	30	50	Stanfel (1985)
32	30	50	Stanfel (1985)
33	30	90	King & Nakornchai (1982)
34	37	53	McCormick et al. (1972)
35	40	100	Chandrasekharan & Rajagopalan (1987)

insertion phase has w-1 iterations, where, at the beginning of the k-th iteration, all cells are divided into two subsequences: the sequence of already processed cells, C^* and the sequence of $w - k$ unprocessed cells, C^{w-k}. At each iteration k of the insertion phase, two types of selections are performed: a selection $f_2(k, C^{n-k})$ of the entity *en* from C^{w-k} and the selection $f_3(C^*, k, en)$ of the cell from C^* where *en* should be inserted. If there is no entity that meets the criterion, *en* remains in the current cell. After each improvement of the objective function value, the procedure restarts from the beginning.

In the GCACFP version of the GCA, the algorithm keeps all partial partitions that result in current optimal value of the objective function. The only input parameter, defined by the manager is an integer, *track*, equal to the upper limit of the number of the simultaneously tracked partitions. This definition also allows a trivial application of a parallel programming, because the individuals in the initial population can be treated completely independently of each other. According to the generalized form of GCA, the precise definition of GCACFP requires only a definition of functions f_1, f_2 and f_3 which are very simple for CFP:

6. Computational results

All relevant CFP algorithms have been tested on 35 instances which can be downloaded from http://mauricio.resende.info/data/cell-formation/. The instance sizes and their sources are presented in Table 9. Due to the inconsistency between the original published source for the test problems and data that has been published, reported values of the objective function may differ by a small amount. For example, instance **No.1*a*** is a different instance from **No.1,** but in certain papers referenced to the same source. Brusco (2015) has prepared a supplement associated with his manuscript that contains precise data for each problem, as well as FORTRAN source codes for his heuristic. In his paper, the best published results for all instances are obtained with slight deviations for instances 15, 27 and 31, due to a data discrepancy among some versions of this problem used in the literature.

The new algorithm was coded in C# and implemented on a laptop computer with Intel Core 2 Duo CPU T6600, 2.2 GHz, 6 GB of installed memory, running Microsoft Windows 7. The results obtained for each of the problems are compared with the best-known results reported for the same problems, by using the 19 methods marked with asterix in Table 1.

For the CPU time comparison, the logical choice was to compare our results with the procedure reporting the best results. Due to the associated supplement in (Brusco, An iterated local search heuristic for cell formation, 2015) and the fact that the reported results in this article are the best ones published yet, an objective efficacy comparison could be implemented. Brusco's FORTRAN source code was translated to C# and such obtained values for computation time were the object of comparison.

Table 10 presents a comparison between ILS and CFOPT. Data for two exact algorithms, CPLEX (Elbenani & Ferland, 2012) and BGEPS, Pinheiro et al. (2016) are reported as a reference. For certain instances (3, 4, 12 – 16, 18, 20, 21, and 25 - 34) the efficacy obtained by the exact algorithms is below the best values. This is

Table 10
Comparison between CPLEX, BGEPS, ILS and CFOPT.

No.	N^M	N^P	CPLEX		BGEPS		ILS		CFOPT	
			Max. Sol.	CPU	Solution	CPU	Avg. Sol.	CPU	Solution	CPU
1	5	7	–	–	75	–	–	–	75	0
1a	5	7	82.35	0.33	–	0.01	82.35	0.2	82.35	0
2	5	7	69.57	0.35	69.57	0.01	69.57	0.3	69.57	0
3	5	18	79.59	0.47	80.85	0.03	80.85	0.6	80.85	0
4	6	8	76.92	0.37	79.17	0.01	79.17	0.6	79.17	0
5	7	11	60.87	1.67	60.87	0.01	60.87	1.8	60.87	0
6	7	11	70.83	1.17	70.83	0.06	70.83	1.7	70.83	0
7	8	12	69.44	1.69	69.44	0.03	69.44	2.8	69.44	0
8	8	20	85.25	2	85.25	0.04	85.25	2.8	85.25	0
9	8	20	58.72	5.03	58.72	4.94	58.72	2.2	58.72	0
10	10	10	75	1.82	75	0.01	75	4.5	75	0
11	10	15	92	1.72	92	0.02	92	2.6	92	0
12	14	24	72.06	78.11	74.24	0.09	74.24	22.4	74.24	0
13	14	24	71.83	103.68	72.86	0.11	72.86	22.4	72.86	0
14	16	24	53.26	101,783.88	53.33	144.91	53.85	40.3	53.85	0
15	16	30	69.53	752.01	69.92	0.54	69.93	25	70.76	0
16	16	43	57.23	–	58.04	125.62	58.04	80.9	58.04	0.05
17	18	24	57.73	14,909.77	57.73	42.32	57.73	55.9	57.73	0
18	20	20	39.66	–	–	–	43.97	32.9	43.97	0
19	20	23	50.81	869,103	50.80	1771.99	50.81	41.20	50.81	0
20	20	35	77.91	144.63	79.37	305.48	78.88	35.50	79.38	0
21	20	35	55.49	–	58.79	14.55	58.6	41.80	58.79	0
22	24	40	100	108.58	100	0.15	100	52.80	100	0
23	24	40	85.11	496.95	85.10	0.44	85.11	58.00	85.11	0.04
24	24	40	73.51	6556.99	73.50	0.78	73.51	65.30	73.51	0.05
25	24	40	47.95	–	53.28	48,743.9	53.29	150.90	53.29	0.16
26	24	40	39.39	–	–	–	48.95	183.10	48.95	0.18
27	24	40	41.84	–	–	–	46.58	169.30	47.26	0.21
28	27	27	50.57	–	–	–	54.82	37.60	54.82	0.02
29	28	46	35.68	–	–	–	47.85	247.60	47.85	0.9
30	30	41	59.7	–	63.04	41.53	63.31	340.70	63.31	1.1
31	30	50	49.69	–	59.77	2622.06	59.77	295.80	60.12	1.14
32	30	50	41.42	–	–	–	50.84	399.20	50.84	0.19
33	36	90	43.77	–	–	–	48.29	740.90	48.29	0.35
34	37	53	57.47	–	–	–	61.36	48.20	61.36	0.02
35	40	100	84.03	1572,184.5	84.03	18.22	84.03	384.10	84.03	0.09

Table 11
Comparison between 19 methods and CFOPT.

No.	GA	EA	ES	HGGA	SACF	HGA	GAA	HHA	EnGGA	WFA	HGDE	ACO	SA	SAYLL	GRASP	GAVNS	ILS	CFPAS	GLCA	CFOPT
1		73.68	73.68			73.68				73.68			**75**		73.68		–	**75**	–	**75**
1a				**82.35**					**82.35**		**82.35**			**82.35**		**82.35**	**82.35**	**–**	**82.35**	**82.35**
2	62.50	62.50	60.87	**69.57**	**69.57**	**69.57**	**69.57**	**69.57**	**69.57**	62.50	**69.57**	**69.57**	**69.57**	**69.57**	62.50	**69.57**	**69.57**	**69.57**	**69.57**	**69.57**
3	77.36	79.59	79.59	79.59	79.59	79.59	79.59	79.59	79.59	79.59	79.59	79.59	80.85	79.59	79.59	**80.85**	**80.85**	**80.85**	79.84	**80.85**
4	76.92	76.92		76.92	76.92	76.92	76.92	76.92	76.92	76.92	76.92	76.92	79.17	76.92	76.92	**79.17**	**79.17**	**79.17**	76.92	**79.17**
5	50.00	53.13	53.13	**60.87**	**60.87**	58.62	**60.87**	**60.87**	**60.87**	53.13	**60.87**	**60.87**	60.00	**60.87**	53.13	**60.87**	**60.87**	**60.87**	**60.87**	**60.87**
6	70.37	70.37	70.37	**70.83**	**70.83**	70.37	**70.83**	**70.83**	**70.83**	70.37	**70.83**	**70.83**	**70.83**	**70.83**	70.37	**70.83**	**70.83**	**70.83**	**70.83**	**70.83**
7		68.29	68.29	69.44		68.30			**69.44**	68.29	**69.44**	**69.44**	**69.44**	**69.44**	68.29	**69.44**	**69.44**	**69.44**	**69.44**	**69.44**
8	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**	**85.25**
9	55.91	**58.72**	**58.72**	**58.72**	**58.41**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**	**58.72**
10	72.79	70.59	70.59	**75.00**	**75.00**	70.59	**75.00**	**75.00**		70.59	**75.00**	**75.00**	**75.00**	**75.00**	**70.59**	**75.00**	**75**	**75**	**75**	**75**
11	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**		**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92.00**	**92**	**92**	**92**	**92**
12		69.86	69.34	72.06		70.83				69.86	72.06	72.06	**74.24**	72.06	69.86	71.83	**74.24**	**74.24**	73.53	**74.24**
13	63.48	69.33	69.24	71.83	71.21	70.51	71.83	71.83		69.33	71.83	71.83	**72.86**	71.83	69.33	**72.86**	**72.86**	**72.86**	71.83	**72.86**
14		52.58	51.96	52.75		51.96			53.26	51.96	53.41	52.75	53.33	53.26	51.96	53.33	**53.85**	53.33	53.33	**53.85**
15		67.83	67.83	68.99		67.83			68.99	67.83	68.99	68.99	69.92	68.99	67.83	69.92	69.93	69.92		**70.76**
16		54.86	54.86	57.53	52.44	54.86	56.13	56.38	57.53	55.90	57.53	57.53	57.98	57.53	56.52	57.42	**58.04**	57.96	57.31	**58.04**
17		54.46	54.46	**57.73**		54.95			**57.73**	54.46	**57.73**	**57.73**	**57.73**	**57.73**	54.46	**57.73**	**57.73**	**57.73**	**57.73**	**57.73**
18	34.16	42.94	42.96	43.18	41.04	43.45	42.94	43.26		42.96	43.45	43.45	**43.97**	43.45	42.96	43.18	**43.97**	43.76	42.55	**43.97**
19	39.02	49.65	49.65	50.81	**50.81**	49.65		**50.81**		49.61	**50.81**	**50.81**	**50.81**	**50.81**	49.65	**50.81**	**50.81**	50.73	**50.81**	**50.81**
20	66.30	76.22	76.14	77.91	78.40	76.14	77.91	78.40	77.91	76.54	77.91	77.91	**79.38**	77.91	76.54	77.78	78.88	79.38	77.91	**79.38**
21	44.44	58.07	58.06	57.98	56.04	58.38		57.61	57.98	58.15	57.98	57.98	**58.79**	57.98	58.15	57.61	58.6	58.7	57.98	**58.79**
22	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100**	**100**	**100**	**100**
23	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**	**85.11**
24	73.03	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**	**73.51**
25	37.62	51.88	51.85	**53.29**	52.44	52.50	52.87	**53.29**	**53.29**	51.97	**53.29**	**53.29**	**53.29**	**53.29**	51.97	53.27	**53.29**	**53.29**	**53.29**	**53.29**
26	34.76	46.69	46.50	**48.95**	47.13	46.84	**48.95**	48.63	**48.95**	47.37	**48.95**	**48.95**	48.57	**48.95**	47.37	46.66	**48.95**	48.79		**48.95**
27	34.06	44.75	44.85	**47.26**	44.64	44.85	**47.26**	46.15	46.58	44.87	**47.26**	**47.26**	46.00	**47.26**	44.87	47.16	46.58	46.26		**47.26**
28		54.27	54.27	54.02		54.31			**54.82**	54.27		**54.82**	**54.82**	**54.82**	54.27	54.31	**54.82**	**54.82**		**54.82**
29		44.37	43.85	46.91		46.43				46.06		47.08	47.68	47.23	46.06	46.06	**47.85**	47.53		**47.85**
30	40.96	58.11	57.69	**63.31**	62.42	60.74		62.59	**63.31**	59.52	**63.31**	**63.31**	62.86	**63.31**	59.52	63.04	**63.31**	62.94	62.95	**63.31**
31	48.28	59.21	59.43	59.77	**60.12**	59.66	**60.12**	**60.12**		60.00	59.77	59.77	59.66	59.77	60.00	**60.12**	59.77	59.77	59.44	**60.12**
32	37.55	50.48	50.51	50.83	50.51	50.51	50.83	50.83		50.51		50.83	50.55	50.83	50.51	50.68	**50.84**	50.83		**50.84**
33		42.12	41.71	46.35		44.67				46.15		47.11	47.93	47.14	45.93	46.03	**48.29**	47.93		**48.29**
34		56.42	56.14	60.64		59.60			60.64	59.85	60.64	60.64	61.16	60.64	59.85	60.57	61.31	61.16	59.48	**61.36**
35	83.90	**84.03**	**84.03**	**84.03**	**84.03**	**84.03**	**84.03**	**84.03**		**84.03**		**84.03**	**84.03**	**84.03**	**84.03**	**84.03**	**84.03**	**84.03**		**84.03**

explained by the fact that the exact algorithms could not produce residuals.

For all tested instances, CFOPT has obtained best values for the objective function. Comparing to ILS, better results are obtained for No15, No20, No21, No27 and No31. Due to the possible deviations of No15, No27 and No31, the general conclusion can be derived that both algorithms have produced best known results yet. This is an expected result, since the 30-year period of testing on these instances consequently implies that the obtained results are also the optimal ones.

Comparison of CPU times, however, confirms the main hypothesis in the work: efficacy improvements with CFOPT are respectable. For 27 instances the optimal solution was obtained, almost instantly, without application of the GCACFP. The GCACFP was applied on seven (16, 25, 26, 27, 29, 30 and 31) ill-structured instances. Even for these instances, the time savings are greater than an order of magnitude. On the largest instances CFOPFT is:

- No.33, 138 times faster,
- No.34, 3068 times faster,
- No.35, 933 times faster.

The detailed information related to the execution of CFOPT on these three instances is presented in the examples:

- for No.33 and No.34 on pages 18 and 19 and Table 6;
- for No.35 on pages 21 and 22 and Table 8.

The detailed examples on functioning CFOPT are also given

- for No.22 on page 14 and Table 4,
- for No.11 on pages 19 and 20 and Table 7a,b,c.

The way instances No.2, No.3, No.4, No.9 and No.34 are, almost instantly processed using CFOPT is described on the page 17.

Table 11 presents the comparison of the objective function for 35 test instances, obtained by applying 19 best published methods and CFOPT. Best results are bolded.

The comparison of the results in Tables 10 and 11 impose an unambiguous and simple conclusion: CFOPT is, on the set of tested instances, the best published algorithm for the CFP. For each instance the best result is obtained with a huge time-savings.

7. Conclusions

Unlike all compared CFP studies which, by default, represent an application of some metaheuristic to the clustering of machines and parts, this study considers the CFP from a completely different view. The proposed approach attempts to explore the essence of the problem and, based on the specifies of the problem, defines the procedures that guide the search toward the global optimum. CFP is a typical example of the problem which is convenient for this approach. The paper investigates and establishes procedures to remove the parts of FSS from the search. This was resulted in the construction of a new hybrid algorithm CFOPT for the CFP. Before the evaluation of this algorithm the target was achieved; a very simple ways to narrow FSS are defined which so far did not receive any attention in the literature. The evaluation was carried out in the spirit of the recent evaluations of algorithms; all instances that are used in the literature are applied, and the results are compared to the best published results for the target group of algorithms.

The main contribution of this paper is a simple and efficient method to improve the efficacy of the procedures for an important problem in the field of production planning, the CFP. The presented approach combines several advantages over known solutions for the same problem:

- CFOPT is the best algorithm, related to the known algorithms for the same problem;
- Enormous time saving of the approach enables improvement of the quality of the results for any given instance;
- The proposed set of different reductions works well for all possible types of instances (well-structured and ill-structured; high and low density; …).
- Instead or after the execution of GCACFP, any improvement heuristic for CFP could be applied. This is a simple way to improve any heuristic for CFP;
- The algorithm does not implement any random diversification search, so it doesn't need any statistical analysis for its evaluation;
- Matrix *B* completely replaces matrix *A* for all necessary calculations. It can be adapted for more complex problems for CMS design and different objective functions;
- The reduction procedures scan the given instance and provides, at the managers' disposal, a set of useful information that can help them in the process optimization.

7.1. Future research

Modularity of the new algorithm routes different directions of the future research:

1. The degree of usefulness of an expert system in a real production environment depends on the degree of alignment between the problem model and the real requirements. It is necessary to analyze the application of this approach in various production environments for different models of the problem.
2. Definition of new, better bounds for the Threshold and the Threshold1, which would further narrow the search of a FSS.
3. Definition of complex, similarity and dissimilarity measures and their implementation in the iterations of the algorithm.
4. Design of improvement heuristics that better fits the problem.
5. Implementation of algorithm for multi-criteria optimization CFP problems.
6. The construction of new, "more difficult" instances for evaluation of CFP algorithms is a prerequisite for objective evaluation of the results related to this problem

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix 1

A new Application is developed in C# as the utility for the analysis of the steps of the hybrid algorithm. Through the Windows of the application, it is possible to track all important parameters and values through the execution of the algorithm. Here we present the execution of the steps of CFOPT on the instance No.14. All results are copied from the Window of the Application interface through the execution of the algorithm.

Input processing routes:

	1	2	3	4	5	6	7	8	9	10	11	12	13
	1	3	4	6	7	8	10	11	15	18	19	22	23
	10	17	22	23									
	1	4	6	16	17	19							
	16	19											
	21												
	2	3	13	15	17	18	21						
	1	19	24										
	1	3	7	10	13	22	23						
	7	24											
	2	9	10	14	17	20	22	23	24				
	7	9	14	20	24								
	2	3	9	11	14	20	24						
	2	7	8	11									
	1	4	6	12	14	24							
	6	9	20										
	9	11	14	17	20	24							

Incidence matrix:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
▶ 01	1		1	1		1	1	1		1	1				1			1	1			1	1	
02										1							1					1	1	
03	1			1		1										1	1		1					
04																1			1					
05																					1			
06		1	1										1		1		1	1			1			
07	1																		1					1
08	1		1				1			1			1									1	1	
09							1																	1
10		1							1	1				1			1			1		1	1	1
11							1		1					1						1				1
12		1	1						1		1			1						1				1
13		1					1	1			1													
14	1			1		1						1		1										1
15						1			1											1				
16									1		1			1			1			1				1

Global parameters:

residual parts: 5
min number of cells: 4
max number of cells: 11

group of machines:
2 3 4 1 1 5 6 7 8 9 10 11 12 13
group of parts:
3 1 4 2 2 5 6 7 8 9 1 10 11 12 13 14 15

degrees of machines:
13 4 6 2 1 7 3 7 2 9 5 7 4 6 3 6
degrees of parts :
5 4 4 3 0 4 5 2 5 4 4 1 2 5 2 2 5 2 4 5 2 4 4 7

resid. of machines:
5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
resid. of parts:
5 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Total Similarity machines: 14.302
Similar density machines: 0.151432941176470588235294117 6
Total Similarity parts: 26.293
Similar density parts: 0.278396470588235294117 6470588

Exact reductions:

Grupped machines:

1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16

Grupped parts:

1	9 20
2	10 22 23
3	15 18
4	1
5	2
6	3
7	4
8	6
9	7
10	8
11	11
12	12
13	13
14	14
15	16
16	17
17	19
18	21
19	24

Matrix B:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
▶ 01	0	3	2	1		1	1	1	1	1	1						1		
02		3														1			
03				1			1	1							1	1	1		
04															1		1		
05																		1	
06			2		1	1							1			1		1	
07				1													1		1
08		3		1		1			1				1						
09									1										1
10	2	3			1									1		1			1
11	2								1					1					1
12	2				1	1					1			1					1
13					1				1	1	1								
14				1			1	1				1		1					1
15	2							1											
16	2										1			1		1			1

Parts processed on each machine:

1 2 3 4 6 7 8 9 10 11 17
2 2 16
3 4 7 8 15 16 17
4 15 17
5 18
6 3 5 6 13 16 18
7 4 17 19
8 2 4 6 9 13
9 9 19
10 1 2 5 14 16 19
11 1 9 14 19
12 1 5 6 11 14 19
13 5 9 10 11
14 4 7 8 12 14 19
15 1 8
16 1 11 14 16 19

Total degrees of parts processed on each machine:

34 13 17 4 1 14 13 21 10 34 22 27 11 19 11 25

Degrees of parts processed on each machine:

1 9 2 4 3 2 3 4 1 3 3
2 9 4
3 4 2 3 1 4 3
4 1 3
5 1
6 2 3 3 1 4 1
7 4 3 6
8 9 4 3 4 1
9 4 6
10 8 9 3 4 4 6
11 8 4 4 6
12 8 3 3 3 4 6
13 3 4 1 3
14 4 2 3 0 4 6
15 8 3
16 8 3 4 4 6

10 6 3 4 4 6

Machines processing each part:

1	10 11 12 15 16
2	1 2 8 10
3	1 6
4	1 3 7 8 14
5	6 10 12 13
6	1 6 8 12
7	1 3 14
8	1 3 14 15
9	1 8 9 11 13
10	1 13
11	1 12 13 16
12	14
13	6 8
14	10 11 12 14 16
15	3 4
16	2 3 6 10 16
17	1 3 4 7
18	5 6
19	7 9 10 11 12 14 16

Total degrees of machines processing each part:

20 21 16 30 23 30 22 24 26 15 26 5 12 28 6 27 20 6 31

Degrees of machines processing each part:

1	7 3 5 1 4
2	10 1 4 6
3	11 5
4	12 5 2 6 5
5	6 8 6 3
6	12 6 6 6
7	12 5 5
8	12 5 5 2
9	12 6 1 4 3
10	12 3
11	12 6 3 5
12	5
13	6 6
14	8 4 6 5 5
15	5 1
16	3 5 6 8 5
17	12 5 1 2
18	0 6
19	2 1 8 4 6 5 5

Similarity matrix for machines:

Input data | Reductions | Dissimil M | Dissimil P | Similarity M | Similarity P | Results

2 3 4 5 6 7 8 9 10 11 12 1 13 14 15 1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
▶ 01	0	0.214	0.267	0.071	0	0.176	0.143	0.429	0.071	0.158	0.059	0.111	0.214	0.188	0.067	0.056
02			0.111	0	0	0.1	0	0.375	0	0.444	0	0	0	0	0	0.111
03				0.333	0	0.083	0.286	0.083	0	0.071	0	0	0	0.333	0.125	0.091
04					0	0	0.25	0	0	0	0	0	0	0	0	0
05						0.143	0	0	0	0	0	0	0	0	0	0
06							0	0.167	0	0.143	0	0.167	0.1	0	0	0.083
07								0.111	0.25	0.091	0.143	0.111	0	0.286	0	0.125
08									0.125	0.231	0.091	0.077	0.1	0.083	0	0
09										0.1	0.4	0.125	0.2	0.143	0	0.143
10											0.4	0.455	0.083	0.154	0.2	0.5
11												0.5	0.125	0.222	0.333	0.571
12													0.222	0.182	0.25	0.625
13														0	0	0.111
14															0.125	0.2
15																0.286
16																

Similarity matrix for parts:

Input data | Reductions | Dissimil M | Dissimil P | Similarity M | Similarity P | Results

2 3 4 5 6 7 1 1 8 9 10 11 12 13 14 15 16 17 18

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
▶ 01	0																		
02	0.125																		
03	0	0.2																	
04	0	0.286	0.167																
05	0.286	0.143	0.2	0															
06	0.125	0.333	0.5	0.286	0.333														
07	0	0.167	0.25	0.6	0	0.167													
08	0.125	0.143	0.2	0.5	0	0.143	0.75												
09	0.111	0.286	0.167	0.25	0.125	0.286	0.143	0.125											
10	0	0.2	0.333	0.167	0.2	0.2	0.25	0.2	0.4										
11	0.286	0.143	0.2	0.125	0.333	0.333	0.167	0.143	0.286	0.5									
12	0	0	0	0.2	0	0	0.333	0.25	0	0	0								
13	0	0.2	0.333	0.167	0.2	0.5	0	0	0.167	0	0	0							
14	0.667	0.125	0	0.111	0.286	0.125	0.143	0.125	0.111	0	0.286	0.2	0						
15	0	0	0	0.167	0	0	0.25	0.2	0	0	0	0	0	0					
16	0.25	0.286	0.167	0.111	0.286	0.125	0.143	0.125	0	0	0.125	0	0.167	0.25	0.167				
17	0	0.143	0.2	0.5	0	0.143	0.4	0.333	0.125	0.2	0.143	0	0	0	0.5	0.125			
18	0	0	0.333	0	0.2	0.2	0	0	0	0	0	0	0.333	0	0	0.167	0		
19	0.5	0.1	0	0.2	0.222	0.1	0.111	0.1	0.2	0	0.222	0.143	0	0.714	0	0.2	0.1	0	

Reduction matrix through the iterations:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
▸ 01	0	3	2	1		1	1	1	1	1	1						1		
02		3														1			
03				1			1	1							1	1	1		
04															1		1		
05																		1	
06			2		1	1							1			1		1	
07				1													1		1
08		3		1		1			1				1						
09									1										1
10	2	3			1									1		1			1
11	2								1					1					1
12	2				1	1					1			1					1
13					1				1	1	1								
14				1			1	1				1		1					1
15	2							1											
16	2										1			1		1			1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
▸ 01	0	4				1	1			2			2		1			2
02	2		3	2	1		1	1	1	1						1		
03			3												1			
04	2				1									1	1	1		
05														1		1		
06																	1	
07				2		1	1					1			1		1	
08					1											1		1
09			3		1		1	1				1						
10								1										1
11		2	3			1							1		1			1
12		2						1					1					1
13						1		1	1	1								
14	2				1						1		1					1
15	1	2																

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
▸ 01	6		6	3			2	1			2				2		
02		2		3	2	1		1	1	1	1					1	
03				3											1		
04		2				1								1	1	1	
05														1		1	
06																	1
07					2		1	1					1		1		1
08	1					1										1	
09				3		1		1	1				1				
10	1								1								
11	2		2						1								
12							1		1	1	1						
13	2	2				1						1					
14		1	2														

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
▶ 01	0	2	6	2	2		2	2	1	1		1			1	
02	12		3			2	1			2				2		
03			3											1		
04		2			1								1	1	1	
05													1		1	
06																1
07				2		1	1					1		1		1
08	1				1										1	
09	1							1								
10	4							1								
11						1		1	1	1						
12	2	2			1						1					
13	2	1														

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
▶ 01	0		5					2						
02	4	1		2	6	2		2	1	1		1		
03	1		12		3		2			2			2	
04					3								1	
05		2		2		1							1	
06		2												
07														1
08	3						1					1	1	1
09		1	1			1								
10							1	1	1	1				
11			2	2		1					1			
12			2	1										

	1	2	3	4	5	6	7	8	9	10	11	12	13
▶ 01	3		4									1	
02				5			2						
03	4	4	1		6		2	1	1		1		
04		1		12	3	2			2			2	
05					3							1	
06													1
07		3				1					1	1	1
08	1		1	1									
09						1	1	1	1				
10	3			2						1			
11	1			2									

End of Loop; Dissimilarity matrix for machines:

Input data | Reductions | Dissimil M | Dissimil P | Similarity M

avg. : -10, Treshold: -10, number of pairs: 22

	1	2	3	4	5	6	7	8	9	10	11
▶ 01	0	-15	-20	-24	-8	-7	-11	-6	-10	-6	-9
02	-15		-21	-10	-10	-7	-13	-9	-7	-8	-5
03	-20	-21		-30	-11	-17	-13	-17	-13	-14	-17
04	-24	-10	-30		-15	-15	-15	-14	-12	-14	-11
05	-8	-10	-11	-15		-5	-8	-7	-8	-10	-7
06	-7	-7	-17	-15	-5		-5	-4	-5	-7	-4
07	-11	-13	-13	-15	-8	-5		-10	-8	-13	-10
08	-6	-9	-17	-14	-7	-4	-10		-7	-6	-6
09	-10	-7	-13	-12	-8	-5	-8	-7		-10	-7
10	-6	-8	-14	-14	-10	-7	-13	-6	-10		-6
11	-9	-5	-17	-11	-7	-4	-10	-6	-7	-6	

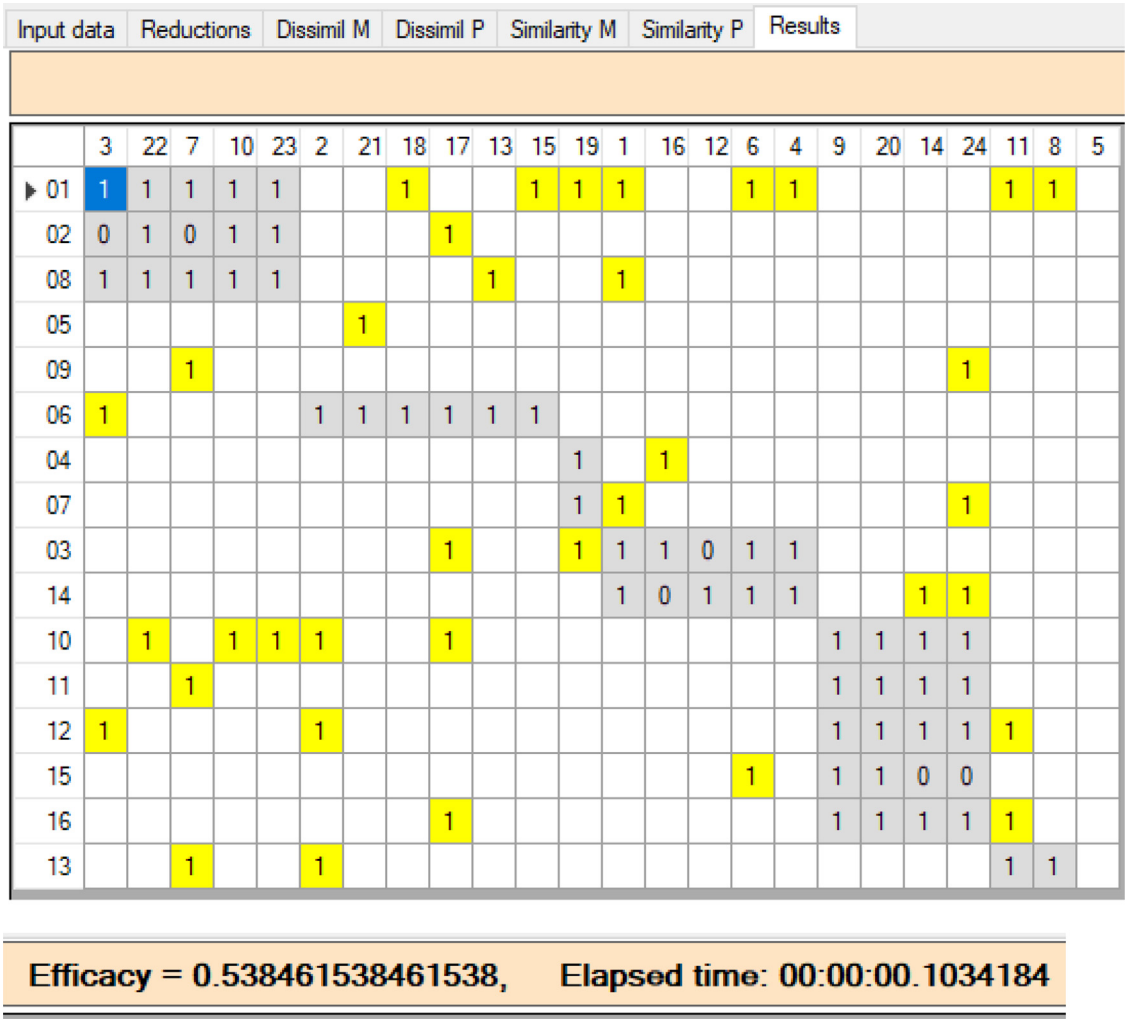
Dissimilarity matrix for parts:

Input data | Reductions | Dissimil M | Dissimil P | Similarity M | Similarity

avg. : -8, Treshold: -8, number of pairs: 27

	1	2	3	4	5	6	7	8	9	10	11	12	13
▶ 01	0	-12	-10	-26	-12	-14	-9	-9	-11	-8	-9	-15	-12
02	-12		-13	-24	-8	-7	-5	-5	-7	-7	-2	-8	-5
03	-10	-13		-22	-12	-8	-9	-6	-8	-5	-6	-6	-6
04	-26	-24	-22		-22	-11	-12	-15	-11	-12	-15	-12	-15
05	-12	-8	-12	-22		-10	-5	-5	-7	-7	-5	-8	-8
06	-14	-7	-8	-11	-10		-6	-3	-1	-5	-3	-2	-3
07	-9	-5	-9	-12	-5	-6		-1	-3	-6	-4	-10	-7
08	-9	-5	-6	-15	-5	-3	-1		-3	-3	-4	-7	-4
09	-11	-7	-8	-11	-7	-1	-3	-3		-5	-6	-5	-6
10	-8	-7	-5	-12	-7	-5	-6	-3	-5		-3	-6	-3
11	-9	-2	-6	-15	-5	-3	-4	-4	-6	-3		-4	-1
12	-15	-8	-6	-12	-8	-2	-10	-7	-5	-6	-4		-4
13	-12	-5	-6	-15	-8	-3	-7	-4	-6	-3	-1	-4	

Final grouping:



Credit authorship contribution statement

Milos Danilovic: Conceptualization, Methodology, Software, Data curation, Writing - original draft, Visualization, Investigation, Supervision, Validation, Writing - review & editing. **Oliver Ilic:** Conceptualization, Methodology, Visualization, Investigation, Supervision, Validation.

References

Arkat, J., Hosseini, L., & Farahani, M. (2011). Minimization of exceptional elements and voids in the cell formation problem using a multi-objective genetic algorithm. *Expert System with Applications, 38*, 9597–9602.

Askin, R., & Subramanian, S. (1987). A cost-based heuristic for group technology configuration. *International Journal of Production Research, 25*(1), 101–113.

Ballakur, A., & Steudel, H. (1987). A within in cell utilization based heuristic for designing cellular manufacturing system. *International Journal of Production Research, 25*, 639–655.

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

Boe, W., & Cheng, C. (1991). A close neighbour algorithm for designing cellular manufacturing systems. *International Journal of Production Research, 29*(10), 2097–2116.

Brusco, M. (2015). An exact algorithm for maximizing grouping efficacy in part-machine clustering. *IIE Transactions, 47*(6), 653–671.

Brusco, M. (2015). An iterated local search heuristic for cell formation. *Computers & Industrial Engineering, 90*, 292–304.

Bychkov, I., & Batsyn, M. (2018). An efficient exact model for the cell formation problem with a variable number of production cells. *Computers and Operations Research, 91*, 112–120.

Carrie, A. (1973). Numerical taxonomy applied to group technology and plant layout. *International Journal of Production Research, 11*(4), 399–416.

Chan, H., & Milner, D. (1982). Direct clustering algorithm for group formation in cellular manufacture. *Journal of Manufacturing Systems, 1*, 65–75.

Chandrasekharan, M., & Rajagopalan, R. (1986). MODROC: An extension of rank order clustering for group technology. *International Journal of Production Research, 24*(5), 1221–1233.

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

Chandrasekharan, M., & Rajagopalan, R. (1987). ZODIAC: An algorithm for concurrent formation of part-families and machine-cells. *International Journal of Production Research, 25*(6), 835–850.

Chandrasekharan, M., & Rajagopalan, R. (1989). GROUPABILITY: An analysis of the properties of binary data matrices for group technology. *International Journal of Production Research, 27*(6), 1035–1052 .

Danilovic, M., & Ilic, O. (2016). A generalized constructive algorithm using insertion-based heuristics. *Computers and Operations Research, 66*, 29–43.

Dıaz, J., Luna, D., & Luna, R. (2012). A GRASP heuristic for the manufacturing cell formation problem. *Top, 20*(3), 679–706.

Durań, O., Rodriguez, N., & Consalter, L. (2010). Collaborative particle swarm optimization with a data mining technique for manufacturing cell design. *Expert Systems with Applications, 37*, 1563–1567.

Elbenani, B., & Ferland, J. (2012). An exact method for solving the manufacturing cell formation problem. *International Journal of Production Research, 50*, 4038–4045.

Feng, Y., & Pheng, K. (2011). An exact schema theorem for adaptive genetic algorithm and its application to machine cell formation. *Expert Systems with Applications, 38*, 8538–8552.

Goncalves, J., & Resende, M. (2004). An evolutionary algorithm for manufacturing cell formation. *Computers & Industrial Engineering, 47*, 247–273.
James, T., Brown, E., & Keeling, K. (2007). A hybrid grouping algorithm for the cell formation problem. *Computers & Operations Research, 34*, 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*, 213–232.
King, R., & Nakornchai, V. (1982). Machine-component group formation in group technology: Review and extension. *International Journal of Production Research, 20*(2), 117–133.
Knuth, D. E. (2011). The *art of computer programming, volume 4A combinatorial algorithms*. Boston: ADDISON-WESLEY, Pearson Education Inc.
Kumar, K., & Vannelli, A. (1987). Strategic subcontracting for efficient disaggregated manufacturing. *International Journal of Production Research, 25*(12), 1715–1728.
Kumar, K., Kusiak, A., & Vannelli, A. (1986). Grouping of parts and components in flexible manufacturing systems. *European Journal of Operations Research, 24*, 387–397.
Kusiak, A., & Cho, M. (1992). Similarity coefficient algorithm for solving the group technology problem. *International Journal of Production Research, 30*(11), 2633–2646.
Kusiak, A., & Chow, W. (1987). Efficient solving of the group technology problem. *Journal of Manufacturing Systems, 6*(2), 117–124.
Li, X., Baki, M., & Aneja, Y. (2010). An ant colony optimization metaheuristic for machine-part cell formation problems. *Computers & Operations Research, 37*, 2071–2081.
Mahdavi, I., Paydar, M., Solimanpur, M., & Heidarzade, A. (2009). Genetic algorithm approach for solving a cell formation problem in cellular manufacturing. *Expert Systems with Applications, 36*, 6598–6604.
Martinsa, I., Pinheiroa, R., Prottia, F., & Ochi, L. (2015). A hybrid iterated local search and variable neighborhood descent heuristic applied to the cell formation problem. *Expert Systems With Applications, 42*, 8947–8955.
McCormick, W., Schweitzer, P., & White, T. (1972). Problem decomposition and data reorganization by a clustering technique. *Operations Research, 20*, 993–1009.
McLean, C., Bloom, H., & Hopp, T. (1982). The virtual manufacturing cell. In *Proceedings of the 4th IFAC/IFIP conference on information control problems in manufacturing technology* (pp. 105–111).
Mosier, C., & Taube, L. (1985a). The facets of group technology and their impact on implementation. *OMEGA, 13*(5), 381–391.
Mosier, C., & Taube, L. (1985b). Weighted similarity measure heuristics for the group technology machine clustering problem. *OMEGA, 13*(6), 577–578.
Mutel, B., & Ostrosi, E. (2002). Feature-based manufacturing cell formation using a fuzzy-set approach. *International Journal Computer Integrated Manufacturing, 15*(2), 152–167.
Noktehdan, A., Karimi, B., & Husseinzadeh Kashan, A. (2010). A differential evolution algorithm for the manufacturing cell formation problem using group based operators. *Expert Systems with Applications, 37*, 4822–4829.
Noktehdan, A., Seyedhosseini, S., & Saidi-Mehrabad, M. (2016). A Metaheuristic algorithm for the manufacturing cell formation problem based on grouping efficacy. *International Journal of Advanced Manufacturing Technology, 82*(1), 25–37.
Onwubolu, G., & Mutingi, M. (2001). A genetic algorithm approach to cellular manufacturing systems. *Computers & Industrial Engineering, 39*(1–2), 125–144.
Ostrosi, E., & Fougères, A.-J. (2018). Intelligent virtual manufacturing cell formation in cloud-based design and manufacturing. *Engineering Applications of Artificial Intelligence, 76*, 80–95.
Pailla, A., Trindade, A., Parada, V., & Ochi, V. (2010). A numerical comparison between simulated annealing and evolutionary approaches to the cell formation problem. *Expert Systems with Applications, 37*, 5476–5483.
Paydar, M., & Saidi-Mehrabad, M. (2013). A hybrid genetic-variable neighborhood cell formation problem based on group efficiency. *Computers & Operations Research, 40*, 980–990.
Pinheiro, R., Martins, I., Protti, F., Ochi, L., Simonetti, L., & Subramanian, A. (2016). On solving manufacturing cell formation via bicluster editing. *European Journal of Operational Research, 254*, 769–779.
Seifoddini, H. (1989). A note on the similarity coefficient method and the problem of improper machine assignment in group technology applications. *International Journal of Production Research, 27*(7), 1161–1165.
Seifoddini, H., & Wolfe, P. (1986). Application of the similarity coefficient method in group technology. *IIE Transactions, 18*(3), 271–277.
Srinivasan, G., Narendran, T., & Mahadevan, B. (1990). An assignment model for the part- families problem in group technology. *International Journal of Production Research, 28*(1), 145–152.
Stanfel, L. (1985). Machine clustering for economic production. *Engineering Costs and Production Economics, 9*, 73–81.
Stawowy, A. (2006). Evolutionary strategy for manufacturing cell design. *Omega, The International Journal of Management Science, 34*, 1–18.
Tariq, A., Hussain, I., & Ghafoor, A. (2009). A hybrid genetic algorithm for machine–part grouping. *Computers & Industrial Engineering, 56*, 347–356.
Tunnukij, T., & Hicks, C. (2009). An enhanced grouping genetic algorithm for solving the cell formation problem. *International Journal of Production Research, 47*(7), 1989–2007.
Unler, A., & Gungo, Z. (2009). Applying K-harmonic means clustering to the part–machine classification problem. *Expert Systems with Applications, 36*, 1179–1194.
Waghodekar, P., & Sahu, S. (1984). Machine-component cell formation in group technology MACE. *International Journal of Production Research, 22*, 937–948.
Wang, J. (1999). A linear assignment clustering algorithm based on the least similar cluster representatives. *IEEE Transactions on Systems Man and Cybernetics Part A–Systems and Humans, 29*(1), 100–104.
Wu, T., Chang, C., & Chung, S. (2008). A simulated annealing algorithm for manufacturing cell formation problems. *Expert Systems with Applications, 34*, 1609–1617.
Wu, T., Chang, C., & Yeh, J. (2009). A hybrid heuristic algorithm adopting both Boltzmann function and mutation operator for manufacturing cell formation problems. *International Journal of Production Economics, 120*, 669–688.
Wu, T., Chung, S., & Chang, C. (2010). A water flow-like algorithm for manufacturing cell formation problems. *European Journal of Operational Research, 205*, 346–360.
Ying, K., Lin, S., & Lu, C. (2011). Cell formation using a simulated annealing algorithm with variable neighbourhood. *European Journal of Industrial Engineering, 5*(1), 22–42.