NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

as a manuscript

Ilya Sergeevich Bychkov

MODELS AND ALGORITHMS FOR THE CELL FORMATION PROBLEM

PhD Dissertation Summary

for the purpose of obtaining academic degree
Doctor of Philosophy in Computer Science

Nizhny Novgorod — 2020

The PhD dissertation was prepared at the Laboratory of Algorithms and Technologies for Network Analysis at the National Research University Higher School of Economics

Academic Supervisor: Mikhail V. Batsyn

Candidate of Sciences (PhD) in Mathematical Modeling, Numerical Methods and Software Complexes

Leading Research Fellow, Laboratory of Algorithms and Technologies for Network Analysis

1 Introduction

Designing a manufacturing layout is one of the most important stages in the process of building a new manufacturing system or adjusting an existing system to changing environment (new resources allocation, changes in volumes or products etc.). To design a manufacturing layout means to locate manufacturing facilities in order to get different kind of advantages. These advantages include simplifying scheduling, reducing costs of material movement, machines utilization and many others.

Depending on the environment problems static or dynamic manufacturing layout is considered [28]. One of the most popular static layouts is the so-called cellular layout. Cellular manufacturing is a part of Group Technology (GT) concept which was introduced by Russian industrial engineering scientist Sergey Mitrofanov [47, 48] and John Burbidge in the west [10, 11, 12, 13]. GT is a strategy which helps to optimize production process in terms of materials handling and management of a batch-manufacturing system. The key idea of GT is that parts which are similar in terms of manufacturing characteristics should be processed within one unit called manufacturing cell. This idea leads to dividing the manufacturing system into subsystems by determining pars families and machine groups. Each manufacturing cell is associated with a certain physical location, machines set up and responsible for handling particular group of parts. The problem of obtaining such kind of grouping is called the manufacturing Cell Formation Problem (CFP). The goal of designing this cellular layout is to reduce production costs by maximizing loading of machines within cells and minimizing movement of parts from one cell to another. Effective partitions provide reduction in setup and transfer costs and savings in plant space.

In this thesis the CFP with variable number of production cells is considered.In the most general formulation the cell formation problem can be considered as the so-called biclustering problem. This kind of problems was first described by Hartigan [34, 35]. The term "biclustering" was introduced by Mirkin [45]. Later Mirkin also defined the notion of approximate biclusters (together with triclusters and p-clusters) and suggested two biclustering algorithms on binary data [46]. [18] considers the most widely used biclustering techniques and applications . [49] and [19] suggested models and methods for solving feature selection problem using consistent biclustering.

To the best of our knowledge there are no papers providing the proof of complexity (NP status) of the CFP problem. There are many publications, which mention that the problem NP-hard [44, 32, 20]. Some of them [57, 29] refer to the paper of Dimopoulos & Zalzala [27], but this paper does not include the proof of np-hardness.

Many papers including James et al. [36]; Chung et al. [25]; Paydar & Saidi-Mehrabad [52]; Solimanpur et al. [55]; Utkina et al. [58] refer to Ballakur & Steudel [4] when discussing the NP-hardness of the CFP. However Ballakur & Steudel [4] present a heuristic for the CFP with different objective functions and do not state anything about the NP status of these CFP formulations. Moreover, the most important objective function for the CFP, the grouping efficacy, was introduced later by Kumar & Chandrasekharan [42]. At the same time the grouping efficacy is currently widely accepted and considered as the best function successfully joining the both objectives of inter-cell part movement minimization and intra-cell machine loading maximization.

In the current research we provide rigorous proofs of NP-completeness for different

formulations of the CFP with the fractional grouping efficacy objective as well as the linear objective minimizing the total number of exceptions and voids.

Object of Research is the CFP in its formulation with given machines number, parts number and binary matrix describing production process. The number of production cells is not initially defined and can vary with respect to the objective function used.

Ph.D. Thesis Goal is the development of methods for solving the CFP effectively comparing to the existing state-of-art approaches and finding rigorous proofs of NP-completeness for different formulations of the CFP.

Novelties of Ph.D thesis are as follows:

- the proof of NP-completeness for the decision version of the CFP with the fractional grouping efficacy objective and linear E+V objective
- two original exact approaches for solving the CFP with fractional objective function
- an effective variable neighborhood search heuristic

The suggested algorithms have a wide range of **practical applications**. The first is designing effective cellular manufacturing layout which allows to improve productivity of batch-production systems. In addition, the formulation considered in this thesis can be also used for solving different biclustering problems (e.g. gene expression problems in biology) as well as clustering problems on bigraphs (e.g. bicluster graph editing problem).

Author's contribution includes the development and implementation of models and algorithms, proofs of theorems and propositions, algorithms testing, performing computational experiments and preparing research papers.

The author has the certificate of official registration of computer program with id №2014610434 - "Heuristic algorithm for solving the Cell Formation Problem".

Papers. Results of this work are published in 5 scientific papers in international peer-reviewed journals and conference proceedings.

First-tier publications:

1. Bychkov, I., Batsyn, M. (2018) An efficient exact model for the cell formation problem with a variable number of production cells. Computers & Operations Research, 91, 112-120, Q1 (Ilya Bychkov suggested and developed the exact approach, implemented it, performed computational experiments and prepared the article)

2. Bychkov, I., Batsyn, M., Pardalos, P. M. (2014). Exact model for the cell formation problem. Optimization Letters, 8(8), 2203-2210, Q2 (Ilya Bychkov developed the exact approach, implemented it, performed computational experiments, proved the propositions and prepared the article)

3. Batsyn, M., Batsyna E., Bychkov I. (2019) NP-completeness of cell formation problem with grouping efficacy objective, International Journal of Production Research (Q1), Published online: 26 Sep 2019, https://doi.org/10.1080/00207543.2019.1668072 (Ilya Bychkov participated in discussing and proving of all theorems and propositions)

4. Bychkov, I., Batsyn, M., Sukhov, P., Pardalos, P.M (2013) Heuristic Algorithm for the Cell Formation Problem. In: Goldengorin et al (eds) Models, Algorithms,

and Technologies for Network Analysis. Springer Proceedings in Mathematics & Statistics 59, 43–69. (Ilya Bychkov suggested and developed the heuristic algorithm, implemented it, performed computational experiments and prepared the article)

Second-tier publications:

5. Bychkov I., Batsyn M., Pardalos P.M. (2017) Heuristic for Maximizing Grouping Efficiency in the Cell Formation Problem. In: Kalyagin et al (eds) Models, Algorithms, and Technologies for Network Analysis. Springer Proceedings in Mathematics & Statistics 197, 11–26. (Ilya Bychkov suggested and developed the heuristic algorithm, implemented it, performed computational experiments and prepared the article)

6. Batsyn, M., Bychkov I., Goldengorin B., Pardalos P., Sukhov P. (2013) Pattern-based heuristic for the cell formation problem in Group Technology. In: Goldengorin et al (eds) Models, algorithms, and technologies for network analysis. Springer Proceedings in Mathematics & Statistics 32, 11–50 (Ilya Bychkov participated in discussion and implementation of the algorithm, performed computational experiments and prepared the article)

Reports at conferences and seminars:

• The 8th International Conference on Network Analysis (NET 2018), May 18-19, Yandex, Moscow, Russia. From Cell Formation Problem to Biclustering and Graph Editing.

• Research Seminar of the Graduate School of Computer Science, CS HSE, September 29, 2017, Computer Science Faculty, Higher School of Economics, Moscow, Russia. An effective exact model for solving the cell formation problem.

• LATNA research seminar, 2016, Nizhny Novgorod, Russia. On solving manufacturing cell formation via bicluster editing.

• The 20th Conference of the International Federation of Operational Research Societies IFORS-2014, July 13-18, 2014, Barcelona, Spain. Multi-start local search heuristic for the cell formation problem.

• Third International Conference on Network Analysis, 2013, Nizhny Novgorod, Russia. An exact model for the cell formation problem.

• Second International Conference on Network Analysis, 2012, Nizhny Novgorod, Russia. ”Patterns” for solving the Cell Formation Problem.

2 Problem statement

2.1 General problem statement

In classical formulation the CFP is defined by a binary matrix A with m rows representing machines and p columns representing parts. In this machine-part matrix $a_{ij} = 1$ if part j is processed on machine i. The objective is to form production cells, which consist of machines and parts together, optimizing some production metrics such as the ones we mention above.

As an example of input data, we present the instance of Waghodekar & Sahu [60] shown in Table 1. This instance consists of 5 machines and 7 parts. The ones in a machine-part matrix are called operations. In Table 2 a solution with 2 manufacturing cells is presented. The first manufacturing cell contains machines m_1, m_4 with parts p_1, p_7 and the second manufacturing cell contains machines m_2, m_3, m_5 with parts p_2, p_3, p_4, p_5, p_6. Some parts have to be moved from one cell to another for processing (e.g. part p_6 needs to be processed on machine m_1, so it should be transported from its cell 2 to cell 1). The operations lying outside cells are called exceptional elements or exceptions. There can be also non-operation elements inside cells where $a_{ij} = 0$. These elements reduce machine load and are called voids. The goal is to minimize the number of exceptions and the number of voids at the same time.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7
m_1	1	0	0	0	1	1	1
m_2	0	1	1	1	1	0	0
m_3	0	0	1	1	1	1	0
m_4	1	1	1	1	0	0	0
m_5	0	1	0	1	1	1	0

Table 1: Machine-part 5×7 matrix from Waghodekar & Sahu [60]

	p_7	p_1	p_2	p_3	p_4	p_5	p_6
m_1	1	1	0	0	0	1	1
m_4	0	1	1	1	1	0	0
m_2	0	0	1	1	1	1	0
m_3	0	0	0	1	1	1	1
m_5	0	0	1	0	1	1	1

Table 2: Solution with 2 production cells

2.2 Objective functions

There are a few different objective functions proposed to measure the quality of a machine-part clustering. Chandrasekharan & Rajagopalan [24] suggested one of the first quantitative measures for comparing cell formation problem solutions. This measure is called grouping efficiency measure. It is the weighted sum of two values η_1 and η_2:

$$\eta = q\eta_1 + (1-q)\eta_2$$

$$\eta_1 = \frac{n_1 - n_1^{out}}{n_1 - n_1^{out} + n_0^{in}} = \frac{n_1^{in}}{n^{in}},$$

$$\eta_2 = \frac{mp - n_1 - n_0^{in}}{mp - n_1 - n_0^{in} + n_1^{out}} = \frac{n_0^{out}}{n^{out}},$$

Here n_1 – the number of ones in the machine-part matrix, n^{in} – the number of elements inside the cells, n^{out} – the number of elements outside the cells, n_1^{in} –the number of ones inside the cells, n_1^{out} – the number of ones outside the cells, n_0^{in} – the number of zeros inside the cells, n_0^{out} – the number of zeros outside the cells. η_1 defines intracell utility of machines, η_2 defines inter-cell movements of parts (the number of zeros out of cells to the total number of elements out of cells), q – a coefficient ($0 \leq q \leq 1$) which is a borderline between importance of machines utility and the inter-cell movement in the objective function. It is usually taken equal to 0.5, so it is equally important to maximize the machine loading and minimize the intercell movement.

The grouping efficiency objective function has several drawbacks [54]. It is not always correct in differentiating good and bad quality solutions. This measure gives values 0.75 to 1 in the majority of cases depending on coefficient q [42]. As a result, some bad solutions can have a high value of objective function. Also, with increasing matrix size the influence of exceptional elements (ones out of clusters) becomes smaller. Another bottleneck of the grouping efficiency is the necessity of choosing parameter q.

Kumar & Chandrasekharan [42] suggested another quality measure for the CFP – grouping efficacy. It is defined as:

$$\tau = \frac{n_1^{in}}{n_1 + n_0^{in}},$$

Grouping efficacy overcomes limitations of the grouping efficiency measure and is the most popular among researchers in this field. Finally, some papers consider so called *exceptions* + *voids* measure:

$$E + V = n_1^{out} + n_0^{in}$$

Krushinsky & Goldengorin [39] solved to optimality some moderate instances of the manufacturing cell formation problem for exceptions + voids objective function.

2.3 Mathematical model

In this research the manufacturing cell formation problem formulation with grouping efficacy objective function is considered. There are several variants related to cell size in the literature:

- allowing any cells residual cells (cells containing only machines or parts)
- prohibiting residual cells (cells containing only machines or parts)
- prohibiting residual and singleton cells (cells with one machine and one or more parts or vice versa)

The most popular option is prohibiting only residual cells. For the classical formulation we assume that singletons can appear in solutions and residual cells are prohibited. In my new model and in computational experiments we consider the first two options.

A straightforward integer fractional programming (IFP) model for the cell formation problem with the grouping efficacy objective function allowing singletons and prohibiting residual cells is given below. Here the following notation is used: m is the number of machines, p is the number of parts, a_{ij} equals to 1 if machine i processes part j and

c is the maximal possible number of production cells. Since each production cell has to contain at least one machine and at least one part $c = \min(m; p)$.

(IFP model):

$$x_{ik} = \begin{cases} 1, & \text{if machine } i \text{ belongs to cell } k, \\ 0, & \text{otherwise} \end{cases}$$

$$y_{jk} = \begin{cases} 1, & \text{if part } j \text{ belongs to cell } k, \\ 0, & \text{otherwise} \end{cases}$$

$$max \quad \frac{\sum_{i=1}^{m}\sum_{j=1}^{p}\sum_{k=1}^{c} a_{ij}x_{ik}y_{jk}}{\sum_{i=1}^{m}\sum_{j=1}^{p} a_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{p}\sum_{k=1}^{c}(1-a_{ij})x_{ik}y_{jk}} \tag{1}$$

Subject to:

$$\sum_{k=1}^{c} x_{ik} = 1, \quad i = 1, ..., m \tag{2}$$

$$\sum_{k=1}^{c} y_{jk} = 1, \quad j = 1, ..., p \tag{3}$$

$$\sum_{i=1}^{m} x_{ik} \leq m \cdot \sum_{j=1}^{p} y_{jk}, \quad k = 1, ..., c \tag{4}$$

$$\sum_{j=1}^{p} y_{jk} \leq p \cdot \sum_{i=1}^{m} x_{ik}, \quad k = 1, ..., c \tag{5}$$

Objective function (1) is the grouping efficacy measure where the numerator is the number of ones inside cells (n_1^{in}) and two sums in the denominator are the total number of ones (n_1) and the number of zeros inside cells (n_0^{in}) respectively. Constraints (2) and (3) require that each machine and each part is assigned to exactly one production cell. The following two inequalities (4) and (5) prohibit residual cells (without machines or parts). The left part of (4) is the total number of machines assigned to the particular cell (this sum is not greater than m) and the right part is the total number of parts assigned to that cell (multiplied by m). It means that if we have at least one machine assigned to some cell there should be at least one part assigned to this cell. This model allows us to have any number of cells in the optimal solution. For example if optimal solution has only two cells then variables x_{ik} and y_{jk} will be zero for all k except only two values of k.

3 Existing approaches and models

Many different approaches have been proposed for solving the cell formation problem. Most of them provide heuristic solutions and only a few exact methods have been suggested.

King and Nakornchai [38] and Chandrasekaran & Rajagopalan [22] proposed MODROC and ROC2 - two improved versions of the so-called rank order clustering algorithm [37] and applied it to machine-part clustering. Chandrasekaran & Rajagopalan [21] presented ideal-seed clustering algorithm with an upper bound on the number of clusters, which was obtained using graph representation of the problem. Also Chandrasekaran & Rajagopalan [21] introduced a quantitative measure for the CFP called grouping efficiency. Later Chandrasekharan & Rajagopalan [23] presented ZODIAC approach where rows and columns were first clustered separately and then - using ideal seeds. Srinivasan & Narendran [56] introduced GRAFICS - another nonhierarchical clustering approach for the cell formation, which initially groups machines by solving the assignment problem.

Several authors used graph theory to solve the cell formation problem. Kumar et al. [40] formulated it as k-decomposition of the graph where machines and parts are vertices and edges are the interconnections for fixed number k of production cells. Vannelli & Kumar [59] and Kumar & Vannelli [41] used graph models to determine machines and parts needed to obtain a perfect block diagonal structure. Askin et al. [2] formed a graph structure using similarity coefficients between machines/parts and then formulated machine-part clustering as a problem of finding Hamiltonian path. Ng [50] and Ng [51] used spanning trees to solve the problem. Kusiak [43], Wang & Roze [61] and Won [62] proposed various mathematical programming models based on the p-median problem.

3.1 State-of-art heuristic algorithms

Today many state-of-art results for hard optimization problems are obtained by different metaheuristic or hybrid metaheuristic algorithms and the cell formation problem is not an exception. One of the well-known papers belongs to Gonçalves and Resende [32]. In this research a new effective genetic algorithm for machines clustering and a local search algorithm for the following parts assignment have been developed. These authors have collected and provided the 35 GT problems dataset which has started a race between heuristics in this area.

James et al. [36] provided a genetic algorithm where a chromosome encodes both machine and part clustering at the same time. They used local search by Gonçalves and Resende [32] and achieved impressive results for grouping efficacy objective.

Paydar & Saidi-Mehrabad [52] have provided a linear fractional programming model and hybrid variable neighborhood search algorithm [33] for real-sized CFP with exceptions plus voids objective.

Brusco [8] suggested a simple competitive iterated local search heuristic with machines/parts relocation as a local search routine and several sequential random relocations of machines and parts as a perturbation stage.

Goldengorin et al. [30] and Goldengorin et al. [31] have obtained good results for large size cell formation instances with grouping efficiency objective using p-median approach.

3.2 State-of-art exact models and approaches

Krushinsky & Goldengorin [39] and Goldengorin et al. [31] provided two MINpCUT exact models based on the well-known minimum multicut graph partition problem. The objective function considered in this research is minimization of the exceptional elements number for a fixed number of cells.

Elbenani & Ferland [29] presented a mixed-integer linear programming model which maximizes the most popular objective for the cell formation problem - the grouping efficacy, introduced by Kumar & Chandrasekharan [42]. These authors suggested to apply Dinkelbach algorithm since the grouping efficacy is a fractional objective function. Their model allows solving the cell formation problem only if the number of production cells is predefined. Thus, the suggested approach cannot guarantee global optimality of the obtained solutions with respect to a variable number of production cells. In many cases the computational times for this model are quite long or memory limitations are exceeded and the optimal solutions cannot be found.

Brusco [9] introduced two exact approaches for solving the cell formation problem with the grouping efficacy objective. The first is a mixed-integer linear programming model which is based on a general two-mode clustering formulation with some simplifying assumptions (e.g. the numbers of clusters by rows and columns are equal). This model looks interesting, but requires too much time to be solved for many even medium-sized instances. The second approach is a branch-and-bound algorithm combined with a relocation heuristic to obtain an initial solution. The branch and bound approach is able to solve about two times more problem instances and the computational times are greatly improved as well. It runs fine on well-structured (with grouping efficacy value > 0.65 - 0.7) medium-sized problems. Two major assumptions are made for both of these approaches. Singletons are permitted (manufacturing cells containing only one machine or one part) that is quite a common practice. Residual cells are permitted (cells containing only machines without parts, or only parts without machines). Also, the number of production cells is predefined for the both approaches, but for some test instances several values for the number of cells are considered in computational experiments. Some authors used biclustering approaches to solve the cell formation problem.

Boutsinas [7] applied simultaneous clustering for both dimensions (machines and parts) and minimized the number of voids plus the number of exceptional elements.

[63] presented an exact approach which finds Pareto frontier and solves a bi-objective optimization problem which can be formulated as the CFP.

Pinheiro et al. [53] reduced the cell formation problem to another biclustering problem - bicluster graph editing problem and suggested an exact method and a linear programming model which provides good computational results for the grouping efficacy objective.

4 Contents

The contributions of this research are:

- an exact approach for solving the CFP based on fixing the grouping efficacy denominator and three-index integer linear programming model
- an exact approach based on the novel two-index integer linear programing model and Dinkelbach algorithm
- multi-start variable neighborhood search heuristic algorithm for grouping efficiency and grouping efficacy objectives

We would like to highlight that many researchers in the field use the 35 GT instances dataset provided by Gonçalves and Resende [32]. These instances are taken from different cell formation research papers (references to the original sources are shown in 4). Some problem instances in this 35 GT dataset have errors and differ from the ones presented in the original papers. Many researchers including Elbenani & Ferland [29] and Pinheiro et al. [53] have performed their computational experiments using these data from Gonçalves and Resende [32]. We have reviewed all the original sources, comparing and forming the corrected version of this popular dataset. We have also collected many other problem instances less popular and formed a new dataset. All data can be downloaded from website opt-hub.com or researchgate.net (full urls can be found in references).

4.1 Problem complexity

Although most papers in last decades consider the CFP in its biclustering formulation, there are no papers providing the proof of its NP status to the best of our knowledge. It this section the proof of NP-completeness is provided for the CFP with the fractional grouping efficacy objective. Hereafter we consider the decision versions of problems. All the results related to the complexity of the CFP are presented in Batsyn et al. [5].

First we consider the so-called Bicluster Graph Editing Problem (BGEP). The BGEP consists in determining the minimum number of edges which should be added to/removed from the given bipartite graph so that it transforms to a set of isolated bicliques. NP-completeness of BGEP has been proved in Amit [1] .

Theorem 1. ***[1]*** *The BGEP problem is NP-complete because the NP-complete 3-exact 3-cover problem can be polynomially reduced to BGEP.*

Using this result we prove that CFP with exceptions + voids objective is also NP-complete.

Theorem 2. *The CFP with linear objective e+v is NP-complete since it is equivalent to the BGEP problem.*

There is a one-to-one correspondence between these two problems. Every machine in the CFP corresponds to a vertex in one part of the bipartite graph in the BGEP, and every part in the CFP corresponds to a vertex in another part of this graph. The machine-part matrix in the CFP coincides with the bipartite graph biadjacency matrix in the BGEP. Every exception in a solution of the CFP corresponds to an edge which

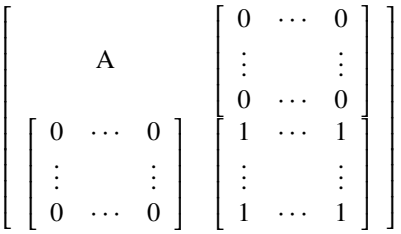

Figure 1: Initial matrix A is extended by adding a big block of ones

should be removed from the bipartite graph in the BGEP in order to transform it to a set of isolated bicliques. And every void in a CFP solution corresponds to an edge which should be added to the bipartite graph in the BGEP. Exceptions + voids objective $(e+v)$ is equivalent to the BGEP objective of minimizing the number of added / removed edges needed to transform the input bipartite graph to a set of isolated bicliques. Every biclique corresponds to a rectangular cell in the CFP.

To prove the NP-completeness of the CFP with grouping efficacy objective we suggest the reduction of the $e + v$ version to it. The grouping efficacy function can be rewritten as:

$$\frac{n_1^{in}}{n_1 + n_0^{in}} = 1 - \frac{e+v}{n_1+v} \rightarrow max \iff \frac{e+v}{n_1 + v} \rightarrow min$$

This expression is almost equivalent to the linear $e + v$ objective, except the value of v in the denominator. Our idea is to nullify the influence of this value by significant increasing of the number of ones in the matrix n_1. We reduce the CFP with $e + v$ to CFP with grouping efficacy by extending the original machine-part matrix A with a big block of ones as it is shown in Figure 1.

Proposition 1. *If the machine-part matrix for the CFP with grouping efficacy has identical rows then there will be optimal solutions in which these rows belong to the same cell.*

Proposition 2. *If the number of added ones in the extended matrix is equal to $(mp)^2$ then the maximum of grouping efficacy on this matrix is obtained at the same solution (extended with the cell of added ones) at which function e+v has its minimum on matrix A.*

These propositions are used in the proof of the following theorem:

Theorem 3. *The CFP with grouping efficacy objective is NP-complete because the CFP with e+v objective can be polynomially reduced to it.*

Since the decision version of the CFP with grouping efficacy is NP-complete, then its optimization version is NP-hard.

4.2 An exact approach based on fixing the value of grouping efficacy denominator

Since the grouping efficacy is a fractional objective function, we look over all the possible values of the denominator (specifically n_0^{in} , because n_1 is a constant). The CFP is then solved separately for every value of n_0^{in} adding a constraint requiring the number of zeros inside cells to be equal to the chosen constant n_0^{in}. This way the original objective function transforms into maximization of n_1^{in} and we have to solve several subproblems, one for every fixed value n_0^{in} . Then the optimal solution for the CFP is the optimal solution of the problem which has the greatest grouping efficacy among all others. To limit the maximum possible number of zeros inside cells a heuristic solution can be used. Proposition 3 provides an upper bound on the number of zeros inside cells. Proposition 4 provides a lower bound on the number of ones inside cells.

Proposition 3. *Let τ be the grouping efficacy value for some feasible CFP solution. Then n_0^{in} in the optimal solution is not greater than $\lfloor \frac{1-\tau}{\tau} \cdot n_1 \rfloor$.*

Proposition 3 is used to limit the possible number of zeros inside the cells (in the optimal solution) and as a result the number of subproblems we have to consider to solve a CFP instance.

Proposition 4. *Let τ be the grouping efficacy value for some feasible CFP solution. Then n_1^{in} in the optimal solution is not less than $\lceil \tau \cdot (n_1 + n_0^{in}) \rceil$.*

Using Proposition 4, the following inequalities can be added to three-index model:

$$\lceil \tau(n_1 + n_0^{in}) \rceil \leq \sum_{i=1}^{m} \sum_{j=1}^{p} \sum_{k=1}^{c} a_{ij} z_{ijk} \leq n_1 \tag{6}$$

Three-index ILP model is as follows:

$$x_{ik} = \begin{cases} 1, & \text{if machine } i \text{ belongs to cell } k, \\ 0, & \text{otherwise} \end{cases}$$

$$y_{jk} = \begin{cases} 1, & \text{if part } j \text{ belongs to cell } k, \\ 0, & \text{otherwise} \end{cases}$$

$$z_{ijk} = \begin{cases} 1, & \text{if both machine } i \text{ and part } j \text{ belongs to cell } k, \\ 0, & \text{otherwise} \end{cases}$$

$$max \sum_{i=1}^{m} \sum_{j=1}^{p} \sum_{k=1}^{c} a_{ij} z_{ijk} \tag{7}$$

subject to

$$z_{ijk} \leq x_{ik}, \quad \forall i = 1, ..., m, \quad \forall j = 1, ..., p, \quad \forall k = 1, ..., c \tag{8}$$

$$z_{ijk} \leq y_{jk}, \quad \forall i = 1, ..., m, \quad \forall j = 1, ..., p, \quad \forall k = 1, ..., c \tag{9}$$

$$z_{ijk} \geq x_{ik} + y_{jk} - 1, \quad \forall i = 1, ..., m, \quad \forall j = 1, ..., p, \quad \forall k = 1, ..., c \tag{10}$$

$$\sum_{k=1}^{c} x_{ik} = 1, \quad \forall i = 1, ..., m \tag{11}$$

$$\sum_{k=1}^{c} y_{jk} = 1, \quad \forall j = 1, ..., p \tag{12}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{p} z_{ijk} \geq \sum_{i=1}^{m} x_{ik}, \quad \forall k = 1, ..., c \tag{13}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{p} z_{ijk} \geq \sum_{j=1}^{p} y_{jk}, \quad \forall k = 1, ..., c \tag{14}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{p} \sum_{k=1}^{c} (1 - a_{ij}) z_{ijk} = n_0^{in} \tag{15}$$

$$\lceil \tau(n_1 + n_0^{in}) \rceil \leq \sum_{i=1}^{m} \sum_{j=1}^{p} \sum_{k=1}^{c} a_{ij} z_{ijk} \leq n_1 \tag{16}$$

$$x_{ik}, y_{jk}, z_{ijk} \in \{0, 1\} \quad \forall i = 1, ..., m, \quad \forall j = 1, ..., p, \quad \forall k = 1, ..., c \tag{17}$$

This model of the cell formation problem can be described using binary variables x_{ik} and y_{jk}. Machines index i takes values from 1 to m and parts index j - from 1 to p. Cells index k takes values from 1 to $c = \min(m, p)$ because every cell should contain at least one machine and one part and so the number of cells cannot be greater than m and p. The number of ones inside cells is equal to $\sum_{k=1}^{c} \sum_{i=1}^{m} \sum_{j=1}^{p} a_{ij} x_{ik} y_{jk}$, and the number of zeros inside cells is equal to $\sum_{k=1}^{c} \sum_{i=1}^{m} \sum_{j=1}^{p} (1 - a_{ij}) x_{ik} y_{jk}$. We linearize the product $x_{ik} y_{jk}$ in a standard way introducing new boolean variables $z_{ijk} = x_{ik} y_{jk}$ and additional linear constraints (8), (9), (10).

Constraints (11), (12) require that every machine and every part is assigned to exactly one cell. Constraints (13), (14) require that there are no cells which have only machines without parts or only parts without machines. Constraint (15) fixes the total number of zeroes inside cells to be equal to the chosen constant n_0^{in}. Model (7) - (17) is solved for all possible values of n_0^{in} using CPLEX solver.

For the computational experiments, we use the set of the most popular 35 CFP instances from the literature. The results are summarized in Table 3. The first 13 problems and the 22nd problem could be solved exactly in quite a little computational time. For all these instances, the solutions found by our approach are equal to the best-known solutions. Thus, for the 14 instances, the global optimality of the best-known solutions is proved (grouping efficacy values for these instances are in bold).

Solving the remaining instances requires too much memory and computational time (these instances are marked with a star in time column). So, for these instances we run CPLEX with 300s time limit for every subproblem and report the best value of grouping efficacy found. All the computations are performed on Intel Core i7 processor running at 2.2 GHz with 8 GB RAM.

#	Source	Size	Efficacy	Time, sec	Zeros in
1	King and Nakornchai (1982) - Figure 1a	5x7	**0.8235**	0.63	3
2	Waghodekar and Sahu (1984) - Problem 2	5x7	**0.6957**	2.29	3
3	Seifoddini (1989b)	5x18	**0.7959**	5.69	3
4	Kusiak and Cho (1992)	6x8	**0.7692**	1.86	4
5	Kusiak and Chow (1987)	7x11	**0.6087**	9.14	0
6	Boctor (1991) - Example 1	7x11	**0.7083**	5.15	3
7	Seifoddini and Wolfe (1986)	8x12	**0.6944**	13.37	1
8	Chandrasekaran and Rajagopalan (1986a)	8x20	**0.8525**	18.33	0
9	Chandrasekaran and Rajagopalan (1986b)	8x20	**0.5872**	208.36	18
10	Mosier and Taube (1985a)	10x10	**0.75**	6.25	4
11	Chan and Milner (1982)	10x15	**0.92**	2.93	4
12	Askin and Subramanian (1987)	14x23	**0.7206**	259.19	10
13	Stanfel (1985)	14x24	**0.7183**	179.21	10
14	McCormick et al. (1972)	16x24	0.5161*	20829.38 *	8
15	Srinivasan et al. (1990)	16x30	0.6900*	13719.99 *	13
16	King (1980)	16x43	0.5753*	24930.93 *	20
17	Carrie (1973)	18x24	0.5773*	13250.01 *	8
18	Mosier and Taube (1985b)	20x20	0.3871*	43531.77 *	44
19	Kumar et al. (1986)	20x23	0.4672*	33020.13 *	9
20	Carrie (1973)	20x35	0.7785*	11626.98 *	22
21	Boe and Cheng (1991)	20x35	0.4675*	33322.08 *	1
22	Chandrasekharan and Rajagopalan (1989) - Dataset 1	24x40	**1**	1.64	0
23	Chandrasekharan and Rajagopalan (1989) - Dataset 2	24x40	0.8511*	6916.24 *	11
24	Chandrasekharan and Rajagopalan (1989) - Dataset 3	24x40	0.5649*	14408.88 *	0
25	Chandrasekharan and Rajagopalan (1989) - Dataset 5	24x40	0.4656*	34524.47 *	0
26	Chandrasekharan and Rajagopalan (1989) - Dataset 6	24x40	0.4351*	41140.94 *	0
27	Chandrasekharan and Rajagopalan (1989) - Dataset 7	24x40	0.4122*	44126.76 *	0
28	McCormick et al. (1972)	27x27	0.54.02*	22627.28 *	31
29	Carrie (1973)	28x46	0.2465*	71671.08 *	4
30	Kumar and Vannelli (1987)	30x41	0.4844*	22594.2 *	0
31	Stanfel (1985) - Figure 5	30x50	0.5065*	31080.82 *	0
32	Stanfel (1985) - Figure 6	30x50	0.3832*	48977.01 *	0
33	King and Nakornchai (1982)	30x90	0.3941*	99435.64 *	29
34	McCormick et al. (1972)	37x53	0.5960*	47744.04 *	17
35	Chandrasekharan and Rajagopalan (1987)	40x100	0.8403*	24167.76 *	37

Table 3: Three-index model based approach results

4.3 An exact approach based on the two-index model and Dinkelbach algorithm

Due to a large number of variables and constraints in three-index model [15] CPLEX spends too much computational resources solving even small-sized instances (we have

solved only 14 of 35 problem instances). Also, the number of subproblems required to be considered can sometimes be large when grouping efficacy is not very high. As a more powerful approach we introduce a two-index integer linear programming model combined with the Dinkelbach algorithm for solving the CFP. The key idea of this model is removing machine-part-cell relation. Instead of mapping elements to cells we use a simple fact that machines within the same production cell have the same set of parts assigned to that cell.

Two-index integer linear programming model is a novel model providing optimal solutions for the cell formation problem with a variable number of manufacturing cells and the grouping efficacy objective. Unlike the majority of mathematical programming models our model does not contain a direct assignment of machines or parts to cells. We use machine-machine and part-machine assignments instead of the widely used machine-part-cell assignment. This leads to a compact and elegant formulation considering only constraints which ensure a block-diagonal structure of solutions. It allows us to drastically reduce the number of variables and constraints in the model and obtain globally optimal solutions even for some large-sized problem instances.

Together with using our new integer linear programming model we use another way of linearization – Dinkelbach algorithm [26]. The parametric approach introduced by W.Dinkelbach is one of the popular strategies for fractional programming. It reduces the solution of a fractional programming problem to the solution of a sequence of simpler problems. If we consider a fractional programming model with the following objective function:

$$Q(x) = \frac{P(x)}{D(x)}, \tag{18}$$

then Dinkelbach procedure is the following:

- **Step 1** take some feasible solution x^0, compute $\lambda_1 = \frac{P(x^0)}{D(x^0)}$ and let $k = 1$
- **Step 2** solve the original problem with objective function $Q(x)$ replaced with $F(\lambda_k) = P(x) - \lambda_k D(x) \rightarrow max$ and let x^k be the optimal solution
- **Step 3** If $F(\lambda_k)$ is equal to 0 (or less than some predefined tolerance) then stop the procedure and return x^k as the optimal solution.
  Else $k = k + 1, \lambda_k = \frac{P(x^k)}{D(x^k)}$ and go to step 2.

Elbenani & Ferland [29] have also used Dinkelbach approach for linearization of grouping efficacy measure. Although, their computational times are quite high, and the results are given only for the particular number of production cells.

Two-index model:

$$x_{ik} = \begin{cases} 1, & \text{if machines } i \text{ and } k \text{ are in the same cell,} \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ij} = \begin{cases} 1, & \text{if machine } i \text{ and part } j \text{ are in the same cell,} \\ 0, & \text{otherwise} \end{cases}$$

$$max \quad \sum_{i=1}^{m}\sum_{j=1}^{p} a_{ij}y_{ij} - \lambda \cdot (\sum_{i=1}^{m}\sum_{j=1}^{p}(1-a_{ij})y_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{p} a_{ij}) \tag{19}$$

Subject to:

$$2x_{ik} - y_{ij} - y_{kj} \geq -1 \quad i,k = 1,\ldots,m \quad j = 1,\ldots,p \tag{20}$$

$$y_{ij} - y_{kj} - x_{ik} \geq -1 \quad i,k = 1,\ldots,m \quad j = 1,\ldots,p \tag{21}$$

$$y_{kj} - y_{ij} - x_{ij} \geq -1 \quad i,k = 1,\ldots,m \quad j = 1,\ldots,p \tag{22}$$

$$\sum_{j=1}^{p} y_{ij} \geq 1 \quad i = 1,...,m \tag{23}$$

$$\sum_{i=1}^{m} y_{ij} \geq 1 \quad j = 1,...,p \tag{24}$$

Objective function (19) is the grouping efficacy measure linearized using Dinkelbach method. Constraints (20), (21), (22) set relations between machines and parts to ensure the solution can be transformed into the block-diagonal form (which means its feasibility). The last two inequalities (23) and (24) are optional and prohibit residual cells.

We start with setting λ equal to the best known efficacy value for the considered cell formation problem instance. Then we sequentially solve several two-index problems according to the Dinkelbach algorithm and update λ value with the solutions found until our objective function is above 0. To test our second exact approach, we have used two datasets, Testset A and Testset B. All the references to problem instances can be found in Bychkov & Batsyn [17].

Testset A - Classic. The first dataset is a classical 35 GT problem set taken from Gonçalves and Resende [32]. It contains 35 test instances with sizes from 5×7 up to 40×100 (machines $\times$ parts notation). This dataset is very popular and there are lots of computational results obtained by different methods (heuristics and metaheuristics generally). As we highlighted before some problem instances in this dataset have inconsistencies with the original papers they are published in. We have compared these instances to the original ones and corrected the dataset.

Testset B - Extra. Another dataset named *Testset B* consists of other instances taken from different papers. We have looked through many papers on the cell formation problem and formed this new set. There are 32 test instances with sizes from 6×6 to 50×150. A couple of instances from this set have been adopted to the classical formulation of the cell formation problem.

ID	Source	m	p
A1	King and Nakornchai (1982) - Figure 1a	5	7
A2	Waghodekar and Sahu (1984) - Problem 2	5	7
A3	Seifoddini (1989b)	5	18
A4	Kusiak and Cho (1992)	6	8
A5	Kusiak and Chow (1987)	7	11
A6	Boctor (1991) - Example 1	7	11
A7	Seifoddini and Wolfe (1986)	8	12
A8	Chandrasekaran and Rajagopalan (1986a)	8	20
A9	Chandrasekaran and Rajagopalan (1986b)	8	20
A10	Mosier and Taube (1985a)	10	10
A11	Chan and Milner (1982)	15	10
A12	Askin and Subramanian (1987)	14	24
A13	Stanfel (1985)	14	24
A14	McCormick et al. (1972)	16	24
A15	Srinivasan et al. (1990)	16	30
A16	King (1980)	16	43
A17	Carrie (1973)	18	24
A18	Mosier and Taube (1985b)	20	20
A19	Kumar et al. (1986)	23	20
A20	Carrie (1973)	20	35
A21	Boe and Cheng (1991)	20	35
A22	Chandrasekharan and Rajagopalan (1989) - Dataset 1	24	40
A23	Chandrasekharan and Rajagopalan (1989) - Dataset 2	24	40
A24	Chandrasekharan and Rajagopalan (1989) - Dataset 3	24	40
A25	Chandrasekharan and Rajagopalan (1989) - Dataset 5	24	40
A26	Chandrasekharan and Rajagopalan (1989) - Dataset 6	24	40
A27	Chandrasekharan and Rajagopalan (1989) - Dataset 7	24	40
A28	McCormick et al. (1972)	27	27
A29	Carrie (1973)	28	46
A30	Kumar and Vannelli (1987)	30	41
A31	Stanfel (1985) - Figure 5	30	50
A32	Stanfel (1985) - Figure 6	30	50
A33	King and Nakornchai (1982)	30	90
A34	McCormick et al. (1972)	37	53
A35	Chandrasekharan and Rajagopalan (1987)	40	100

Table 4: Testset A - Instances

For the computational experiments we consider two most popular versions of cell size constraints:

1. Residual cells are prohibited, singletons are allowed (each cell has at least 1 machine and 1 part)

2. Residual cells are allowed (cells with only machines or only parts can appear in the final solution)

Several state-of-art exact approaches have been chosen for comparisons. As a platform for our computations we have used a system with Intel Xeon processor running

at 3.4 GHz with 16GB RAM and CPLEX 12.4.0 installed. Due to high-quality initial solutions the Dinkelbach algorithm makes only one or, in rare cases, two iterations.

ID	Source	m	p
B1	Adil (1996)	6	6
B2	Parkin and Li (1997)	6	7
B3	Brown and Sumichrast (2001)	6	11
B4	Chan and Milner (1982)	7	5
B5	Kusiak and Chow (1987)	7	8
B6	Zolfaghari and Liang (2002)	7	8
B7	Won and Kim (1997)	7	10
B8	Sarker and Khan (2001)	8	8
B9	Nair (1999)	8	10
B10	Islam and Sarker (2000)	8	10
B11	Kumar et al. (1986)	9	15
B12	Ham et al. (1985)	10	8
B13	Viswanathan (1996)	10	12
B14	Shargal et al. (1995)	10	38
B15	Won and Kim (1997)	11	10
B16	Seifoddini (1988)	11	22
B17	Moon and Chi (1992)	12	19
B18	Li (2003)	14	14
B19	Chan and Milner (1982) - Fig.3a	15	10
B20	Yang and Yang (2008) - Fig.6b	15	15
B21	Yang and Yang (2008) - Fig.6c	15	15
B22	Yang and Yang (2008) - Fig.6d	15	15
B23	Harhalakis et al. (1994)	17	20
B24	Seifoddini and Djassemi (1991)	18	24
B25	Sandbothe (1998)	20	10
B26	Nagi et al. (1990)	20	51
B27	Won and Kim (1997)	26	28
B28	Yang and Yang (2008) - Fig.7	28	35
B29	Seifoddini and Djassemi (1996)	35	15
B30	Seifoddini and Djassemi (1996)	41	50
B31	Yang and Yang (2008) - Fig.12	46	105
B32	Zolfaghari and Liang (1997)	50	150

Table 5: Testset B - Instances

The instances from Table 4 have been studied widely in the literature. We report results separately for the formulation where minimal cell size is 1×1 (Table 6 and Figure 2) and the formulation with residual cells allowed (Table 7 and Figure 3). In the first case we have selected two approaches for the results comparison:

1. The MILP model by by Elbenani & Ferland [29]
2. The ILP model by Bychkov et al. [15]

Elbenani & Ferland [29] considered a simplified formulation of the cell formation problem solving every problem instance only for one fixed number of production cells.

These authors have performed computational experiments on an AMD processor 2.2 GHz with 4GB RAM. For Testset A we use the best efficacy results from the literature as initial values for λ parameter.

#	Time, sec			Efficacy		
	Elbenani & Ferland (2012)	Bychkov et al. (2014)	two-index model	Elbenani & Ferland (2012) (cells)	Bychkov et al. (2014)	two-index model
A1	2.3	0.63	0.00	0.8235(2)	0.8235	0.8235
A2	1.6	2.29	0.00	0.6957(2)	0.6957	0.6957
A3	3.1	5.69	0.00	0.7959(2)	0.7959	0.7959
A4	2.0	1.86	0.09	0.7692(2)	0.7692	0.7692
A5	30.6	9.14	0.17	0.6087(5)	0.6087	0.6087
A6	4.3	5.15	0.01	0.7083(4)	0.7083	0.7083
A7	9.6	13.37	0.02	0.6944(4)	0.6944	0.6944
A8	3.1	18.33	0.01	0.8525(3)	0.8525	0.8525
A9	3.5	208.36	0.45	0.5872(2)	0.5872	0.5872
A10	1.1	6.25	0.00	0.7500(5)	0.7500	0.7500
A11	1.6	2.93	0.02	0.9200(3)	0.9200	0.9200
A12	2188.7	259.19	0.19	0.7206(7)	0.7206	0.7206
A13	593.2	179.21	0.23	0.7183(7)	0.7183	0.7183
A14	15130.5	*	4.24	0.5326(8)	*	**0.5326**
A15	252.5	*	0.25	0.6953(6)^{E}	*	**0.6899**
A16	183232.5	*	4.80	0.5753(8)	*	**0.5753**
A17	2345.6	*	3.82	0.5773(9)	*	**0.5773**
A18	*	*	32243.10	*	*	**0.4345**
A19	131357.5	*	245.59	0.5081(7)	*	**0.5081**
A20	31.1	*	0.22	0.7791(5)	*	**0.7791**
A21	14583.6	*	24.34	0.5798(5)	*	**0.5798**
A22	11.3	1.64	0.14	1.0000(7)	1.0000	1.0000
A23	230.7	*	0.12	0.8511(7)	*	**0.8511**
A24	1101.1	*	0.16	0.7351(7)	*	**0.7351**
A25	*	*	1026.96	*	*	**0.5329**
A26	*	*	178182.24	*	*	**0.4895**
A27	*	*	*	*	*	*
A28	958714.1	*	1964.00	0.5482(5)	*	**0.5482**
A29	*	*	*	*	*	*
A30	378300.0	*	8.72	0.6331(14)	*	**0.6331**
A31	*	*	136.00	0.6012(13)^{E}	*	**0.5977**
A32	*	*	*	*	*	*
A33	*	*	*	*	*	0.4800
A34	268007.6	*	16323.71	0.6064(3)	*	**0.6064**
A35	7365.3	*	1.34	0.8403(10)	*	**0.8403**

Table 6: Testset A - Computational results (residual cells are prohibited, singletons are allowed)

In case of unrestricted cell sizes (residual cells are allowed) we have compared our results to the following approaches:

1. The branch-and-bound algorithm by Brusco [9]
2. The ILP model by Pinheiro et al. [53]
3. The iterative exact method by Pinheiro et al. [53]

Brusco [9] considers several values for the number of cells, so we compare our

computational time with these times summed up for every test instance. As the hardware platform Brusco [9] reports 3.4 GHz Intel Core i7-2600 with 8GB RAM and Pinheiro et al. [53] the same 3.4 GHz Intel Core i7-2600 with 32 GB RAM.

Since Elbenani & Ferland [29] and Brusco [9] do not consider all possible numbers of production cells we show the number of cells in brackets for these approaches.

#	**Time, sec**				**Efficacy**		
	Brusco (2015)	Pinheiro et al. (2016) IM	Pinheiro et al. (2016) ILP	two-index	Brusco (2015) (cells)	Pinheiro et al. (2016)	two-index
A1	0.01	0.16	0.01	0.01	0.8235(2,3,4)	0.7500^{E}	0.8235
A2	0.01	0.07	0.01	0.01	0.6957(2,3,4)	0.6956	0.6957
A3	0.02	0.09	0.03	0.01	0.8085(2,3,4)	0.8085	0.8085
A4	0.01	0.02	0.01	0.01	0.7916(2,3,4)	0.7917	0.7917
A5	0.6	0,29	0.06	0.17	0.6087(2,3,4,5,6)	0.6087	0.6087
A6	0.04	0.14	0.01	0.01	0.7083(2,3,4,5)	0.7083	0.7083
A7	0.08	0.18	0.03	0.01	0.6944(2,3,4,5)	0.6944^{E}	0.6944
A8	0.01	2.06	0.04	0.01	0.8525(2,3,4)	0.8525	0.8525
A9	35.86	81.46	4.94	0.45	0.5872(2,3,4)	0.5872	0.5872
A10	0.06	0.03	0.01	0.01	0.7500(2,3,4,5,6)	0.7500	0.7500
A11	0.01	0.01	0.02	0.02	0.9200(2,3,4)	0.9200	0.9200
A12	633.91	0.49	0.09	0.03	0.7424(6,7,8)	0.7424	0.7424
A13	2631.76	0.49	0.11	0.03	0.7285(6,7,8)	0.7286	0.7286
A14	24716.34	600.98	144.91	4.88	0.5385(8)	0.5333^{E}	0.5385
A15	1279.93	7.24	0.54	0.16	0.6992(5,6,7)	0.6992^{E}	0.6992
A16	-	1156.23	125.62	4.24	-	0.5804	0.5804
A17	20840.55	87.13	42.32	3.84	0.5773(9)	0.5773^{E}	0.5773
A18	-	*	*	52810.10	-	*	**0.4397**
A19	1375608.66	23928.70	1771.99	249.52	0.5081(7)	0.5081	0.5081
A20	4830.00	1.78	14.55	0.09	0.7888(5,6,7)	0.7938^{E}	0.7888
A21	-	2145.24	305.48	22.60	-	0.5879^{E}	0.5860
A22	0.01	0.02	0.15	0.14	1.0000(7)	1.0000	1.0000
A23	42.29	10.08	0.44	0.14	0.8511(7)	0.8511	0.8511
A24	208158.02	17.46	0.78	0.20	0.7351(7)	0.7351	0.7351
A25	-	371233.00	48743.90	759.70	-	0.5329^{E}	0.5329
A26	-	*	*	134418.65	-	*	**0.4895**
A27	-	*	*	*	-	*	*
A28	-	*	*	46361.97	-	*	**0.5482**
A29	-	*	*	*	-	*	*
A30	-	183.71	41.53	8.00	-	0.6304^{E}	0.6331
A31	-	13807.50	2622.06	64.82	-	0.5977	0.5977
A32	-	*	*	234055.90	-	*	**0.5084**
A33	-	*	*	*	-	*	0.4829
A34	-	*	*	14212.57	-	*	**0.6131**
A35	-	325.53	18.22	1.61	-	0.8403	0.8403

Table 7: Testset A - Computational results (residual cells are allowed)

The results for Testset A are summarized in Table 6 and Table 7. For each algorithm we report the grouping efficacy value and the running time in seconds. Since our testset is larger than the one used by Brusco [9] the missing results are marked as ”-”. For some problems exact solutions have not been obtained because CPLEX runs too long or takes too much memory. These instances are marked as "*". Table 6 shows the results for the

case where we prohibit residual cells. Our previous model from Bychkov et al. [15] also considers a variable number of production cells, but due to its complexity and it is able to solve only 14 test problems of 35. The model suggested by Elbenani & Ferland [29] solved 27 problem instances but only for the one fixed number of production cells for each problem instance. Our new model provides global optimal solutions (with respect to any possible number of cells) for 31 of 35 problem instances. For problem instance A33 we have found a new solution with grouping efficacy 0.48 unknown before.

For 17 instances: A14-A21, A23-A26, A28, A30, A31, A34 and A35 we are the first to prove the global optimality of the best known solutions with respect to a variable number of production cells.

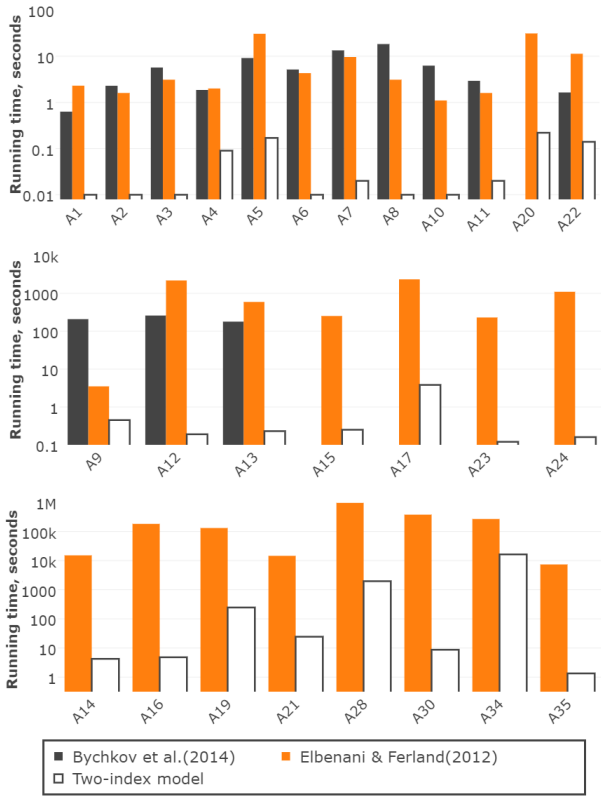

Figure 2: Testset A - No residual cells. Running times comparison.

Running times bar charts for Table 6 are presented in Figure 2. Here we have used logarithmic scale with base 10 for Y axis (running time). Our new model shows really good performance, it works from 7 to 43383 times faster than the model from Elbenani & Ferland [29] and from 11 to 1833 times faster than the model from Bychkov et al. [15]. We must underline that although we use a better hardware platform than Elbenani & Ferland [29], our problem formulation is more complicated than a formulation with

a fixed number of cells.

The results for the formulation with no constraints on cell sizes are summarized in Table 7. The model suggested by Pinheiro et al. [53] solved 27 problem instances to the global optimum. Our approach has obtained exact solutions for 32 of 35 test instances. In addition for problem instances A18, A33 and A34 we have found new solutions unknown before.

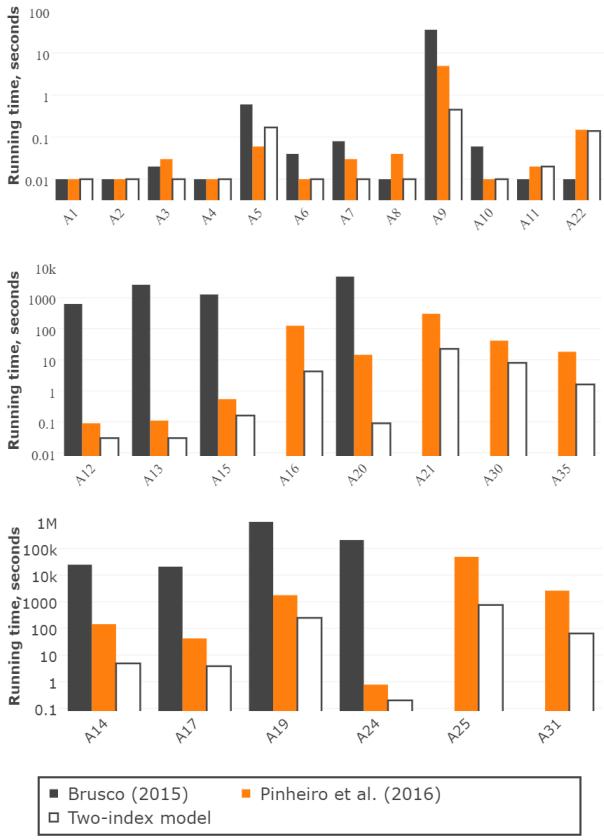

Figure 3: Testset A - Allowed residual cells. Running times comparison.

Since the test instances from Table 5 are less popular in research papers our goal is just to obtain optimal solutions for this set. We have used our multi-start variable neighborhood search heuristic [14] to get good solutions which are then passed as initial values for λ parameter (we pick the best solution found by the heuristic within 30 seconds).

The results for Testset B are shown in Table 8. Here we have found optimal solutions for 31 of 32 test problems. Another result is an excellent performance of our multi-start variable neighborhood search heuristic algorithm: only one of 32 instances solved by the heuristic differs from the exact solution (instance B18). Due to the high computational

#	Time		Efficacy		
	two-index (no residual cells)	two-index (allow residual cells)	Heuristic bound	two-index (no residual cells)	two-index (allow residual cells)
B1	0.01	0.01	0.8095	0.8095	0.8095
B2	0.01	0.01	0.7222	0.7222	0.7222
B3	0.25	0.03	0.6071	0.6071	0.6071
B4	0.01	0.01	0.8889	0.8889	0.8889
B5	0.01	0.01	0.7500	0.7500	0.7500
B6	0.01	0.01	0.7391	0.7391	0.7391
B7	0.01	0.01	0.8148	0.8148	0.8148
B8	0.01	0.01	0.7222	0.7222	0.7222
B9	0.01	0.01	0.7576	0.7576	0.7576
B10	0.01	0.01	0.9000	0.9000	0.9000
B11	0.01	0.02	0.7273	0.7273	0.7297
B12	0.01	0.01	0.8276	0.8276	0.8276
B13	0.36	0.80	0.5962	0.5962	0.6042
B14	0.25	0.30	0.6405	0.6405	0.6405
B15	0.01	0.01	0.8333	0.8333	0.8333
B16	0.16	0.06	0.7391	0.7391	0.7444
B17	0.98	0.26	0.6552	0.6552	0.6842
B18	1.82	1.65	0.6027	0.6129	0.6129
B19	0.03	0.06	0.8000	0.8000	0.8113
B20	0.05	0.03	0.8710	0.8710	0.8710
B21	0.03	0.04	0.8333	0.8333	0.8333
B22	0.05	0.01	0.7258	0.7258	0.7258
B23	0.05	0.06	0.8111	0.8111	0.8111
B24	4.79	7.80	0.5673	0.5673	0.5728
B25	0.20	0.10	0.7600	0.7600	0.8000
B26	13.81	25.75	0.6068	0.6068	0.6078
B27	0.25	0.28	0.7248	0.7248	0.7248
B28	0.83	1.04	0.6729	0.6729	0.6729
B29	33.82	51.76	0.5730	0.5730	0.5745
B30	4.76	8.67	0.7308	0.7308	0.7325
B31	19.69	17.50	0.6799	0.6799	0.6799
B32	*	*	0.6193	*	*

Table 8: Testset B - Computational results

complexity we are unable to solve the largest problem in the set – problem B32 (50×150).

4.4 Multi-start variable neighborhood search algorithm

We present a heuristic approach for obtaining high-quality solutions of the CFP. The suggested heuristic applies an improvement procedure to obtain solutions with high grouping efficiency. This procedure is repeated many times for randomly generated cell configurations. Our computational experiments are performed for popular benchmark instances taken from the literature with sizes from 10×20 to 50×150. Better solutions unknown before are found for 23 instances of the 24 considered.

The main function of our heuristic is presented by Algorithm 1.

First we call FINDOPTIMALCELLRANGE($MinCells, MaxCells$) function that returns

Algorithm 1 Main function

```
function Solve( )
    FindOptimalCellRange(MinCells, MaxCells)
    ConfigsNumber = 2000
    AllConfigs = GenerateConfigs(MinCells, MaxCells, ConfigsNumber)
    return CMHeuristic(AllConfigs)
end function
```

a potentially optimal range of cells - from $MinCells$ to $MaxCells$. Then these values and $ConfigsNumber$ (the number of cell configurations to be generated) are passed to GenerateConfigs($MinCells, MaxCells, ConfigsNumber$) function which generates random cell configurations. The generated configurations $AllConfigs$ are passed to CMHeuristic($AllConfigs$) function which finds a high-quality solution for every cell configuration and then chooses the solution with the greatest efficiency value.

Algorithm 2 Procedure for finding the optimal cell range

```
function FindOptimalCellRange( MinCells, MaxCells)
    if (m > p) then
        minDimension = p
    else
        minDimension = m
    end if
    ConfigsNumber = 500
    Configs = GenerateConfigs(2, minDimension, ConfigsNumber)
    Solution = CMHeuristic(Configs)
    BestCells = GetCellsNumber(Solution)
    MinCells = BestCells - [minDimension * 0,1 ]        ▷ [ ] - integer part
    MaxCells = BestCells + [minDimension * 0,1 ]
end function
```

In function FindOptimalCellRange($MinCells, MaxCells$) (algorithm 2) we look over all the possible numbers of cells from 2 to maximal possible, which equal to $\min(m, p)$. For every number of cells in this interval we generate a fixed number of configurations (we use 500 in this paper) calling GenerateConfigs(2, $minDimension$, $ConfigsNumber$) and then use our CMHeuristic($Configs$) to obtain a potentially optimal number of cells. But we consider not only one number of cells but together with its 10%-neighborhood $[MinCells, MaxCells]$.

Function GenerateConfigs($MinCells, MaxCells, ConfigsNumber$) (algorithm 3) returns a set of randomly generated cell configurations with a number of cells ranging from $MinCells$ to $MaxCells$. We call GenerateConfigsUniform($cells$, $ConfigsNumber$) function which randomly selects with uniform distribution $Configs\ Number$ configurations from all possible cell configurations with the specified number of cells. Note that mathematically a cell configuration with k cells can be represented as an integer partition of m and p values into sums of k summands. We form a set of configurations for every number of cells and then join them.

Algorithm 3 Configurations generation

```
function GenerateConfigs(MinCells, MaxCells, ConfigsNumber)
    Configs = ∅
    for cells = MinCell, MaxCells do
        Generated = GenerateConfigs(cells, ConfigsNumber)
        Configs = Configs ∪ Generated
        return Configs
    end for
end function
```

Algorithm 4 CMHeuristic

```
function CMHeuristic(Configs)
    Best = 0
    for all config ∈ Configs do
        Solution = ImproveSolution(config)
        if Solution > Best then
            Best = Solution
        end if
    end for
    return Best
end function
```

Function CMHeuristic(Configs) (algorithm 4) gets a set of cell configurations and for each configuration runs an improvement algorithm to obtain a good solution. A solution includes a permuted machine-part matrix, a cell configuration, and the corresponding grouping efficiency value. The function chooses the best solution and returns it.

Improvement procedure ImproveSolution($config, \eta_{current}$) (algorithm 5) works as follows. We consider all the machines and the parts in order to know if there is a machine or a part that we can move to another cell and improve the current efficiency $\eta_{current}$. First we consider moving of every part on all other cells and compute how the efficiency value changes. Here $\eta_{part,cell}$ is the efficiency of the current solution where the part with index $part$ is moved to the cell with index $cell$. This operation is performed for all the parts and the part with the maximum increase in efficiency Δ_{parts} is chosen. Then we repeat the same operations for all the machines. Finally, we compare the best part movement and the best machine movement and choose the one with the highest efficiency. This procedure is performed until any improvement is possible and after that we get the final solution.

Testing of the multi-start heuristic algorithm is shown in Table 9. All the references to problem instances can be found in Bychkov et al. [16]. In all the experiments for determining a potentially optimal range of cells we use 500 random cell configurations for each cells number and for obtaining the final solution we use 2000 random configurations.

The heuristic is run on 24 CFP benchmark instances taken from the literature.

Algorithm 5 Solution improvement procedure

```
function ImproveSolution(config, η_current)
    η_current = GroupingEfficiency(config)
    repeat
        PartFrom = 0
        PartTo = 0
        for part = 1, partsNumber do
            for cell = 1, cellsNumber do
                if (η_part,cell > η_current) then
                    Δ_parts = (η_part,cell − η_current)
                    PartFrom = GetPartCell(part)
                    PartTo = cell
                end if
            end for
        end for
        MachineFrom = 0
        MachineTo = 0
        for machine = 1, machinesNumber do
            for cell = 1, cellsNumber do
                if (η_machine,cell > η_current) then
                    Δ_machines = (η_machine,cell − η_current)
                    MachineFrom = GetMachineCell(machine)
                    MachineTo = cell
                end if
            end for
        end for
        if Δ_parts > Δ_machines then
            MovePart(PartFrom, PartTo)
        else
            MoveMachine(MachineFrom, MachineTo)
        end if
    until Δ > 0
end function
```

The sizes of the considered problems vary from 10x20 to 50x150. For every instance we make 50 algorithm runs and report minimum, average and maximum value of the grouping efficiency obtained by the suggested. We compare our results with the best known values taken from Goldengorin et al. [30] and Bhatnagar & Saddikuti [6]. Better solutions unknown before have been found for 23 instances of the 24 considered (best results are in bold). For the CFP instance 6 we have found the same optimal solution with 100% of grouping efficiency as in Goldengorin et al. [30].

5 Conclusion

The cell formation problem is a well known combinatorial optimization problem. Only a few authors have suggested exact approaches for the most popular problem formulation with the grouping efficacy objective function. Most of these works assume that the number of production cells is predefined. In this research two exact approaches are suggested for solving the CFP with variable number of production cells. The first exact approach Bychkov et al. [15] is based on fixing the value of grouping efficacy denominator. Propositions 3 and 4 allows finding a lower and upper bounds on the number of zeros inside cells. Then the initial problem is split into subproblems (one possible number of zeros inside for each) and solved using the three-index integer linear programming model. This was the first approach int the literature which found optimal solutions (with respect to every possible number of production cells) for 14 of 35 problems instances in the popular 35 GT dataset.

As a more powerful approach a two-index integer linear programming model is introduced in combination with the Dinkelbach algorithm [17]. The key idea of this model is removing machine-part-cell relation. Instead of mapping elements to cells we use a simple fact that machines within the same production cell have the same set of parts assigned to that cell. It allows to drastically reduce the number of variables and constraints in the model and obtain globally optimal solutions even for some large-sized problem instances. Computational experiments show that the new approach outperforms the state-of-art exact methods. The 63 of 67 problem instances have been solved to the global optimum with respect to a variable number of production cells. In addition, several new solutions unknown before have been found.

A heuristic approach for obtaining high-quality solutions has been developed [16]. The suggested heuristic applies a variable neighborhood search procedure to obtain solutions with high grouping efficiency. This procedure is repeated many times for randomly generated cell configurations. The computational experiments are performed for popular benchmark instances taken from the literature with sizes from 10×20 to 50×150. Better solutions unknown before have been found for 23 instances of the 24 considered.

Finally, the NP-completeness is proved for the CFP with fractional grouping efficacy objective [5].

Table 9: Computational results of MS VNS heuristic

#	Source	mxp	Efficiency value					Time, sec	Cells
			Bhatnagar & Saddikuti	Goldengorin et al.	Our				
					Min	Avg	Max		
1	Sandbothe (1998)	10x20	0.9540	0.9593 [a]	0.9566	0.9566	**0.9566**	0.36	7
2	Ahi et al. (2009)	20x20	0.9262	0.9385	0.9599	0.9599	**0.9599**	0.62	9
3	Mosier & Taube (1985)	20x20	0.8563	0.8871	0.9011	0.9016	**0.9022**	0,88	9
4	Boe & Cheng (1991)	20x35	0.8831	0.8805	0.9334	0.9347	**0.9355**	1,62	10
5	Carrie (1973)	20x35	0.9076	0.9564	0.9543	0.9578	**0.9579**	1,54	10
6	Ahi et al. (2009)	20x51	0.8786	0.9411	0.9536	0.954	**0.9545**	3,1	12
7	Chandrasekharan & Rajagopalan (1989)	20x40	0.9882	1	1	1	1	1,8	7
8	Chandrasekharan & Rajagopalan (1989)	20x40	0.9533	0.9748	0.977	0.9775	**0.9776**	2,42	12
9	Chandrasekharan & Rajagopalan (1989)	20x40	0.9378	0.9636	0.9684	0.9688	**0.9689**	2,56	12
10	Chandrasekharan & Rajagopalan (1989)	20x40	0.8792	0.9432	0.9611	0.9616	**0.9621**	3,3	15
11	Chandrasekharan & Rajagopalan (1989)	20x40	0.8495	0.9421	0.9594	0.9603	**0.961**	2,84	15
12	Chandrasekharan & Rajagopalan (1989)	20x40	0.8506	0.9232	0.9585	0.959	**0.9595**	2,76	15
13	Nair & Narendran (1996)	20x40	0.9644	0.9739	0.9778	0.9778	**0.9778**	2,12	10
14	Nair & Narendran (1996)	20x40	0.9235	0.9574	0.974	0.974	**0.974**	2,2	14
15	Nair & Narendran (1996)	20x40	0.9325	0.9570	0.9581	0.9603	**0.9617**	2,48	12
16	Nair & Narendran (1996)	20x40	0.9111	0.9640	0.9698	0.9698	**0.9698**	2,78	14
17	Ahi et al. (2009)	25x40	0.9109	0.9552	0.9648	0.9648	**0.9648**	2,58	14
18	Yang & Yang (2008)	28x35	0.9343	0.9382	0.9481	0.9485	**0.9486**	2,46	10
19	Kumar & Vannelli (1987)	30x41	0.9066	0.9722	0.9738	0.9753	**0.9762**	3,54	18
20	Stanfel (1985)	30x50	0.8817	0.9648	0.9677	0.9683	**0.969**	5,02	18
21	King & Nakornchai (1982)	30x90	0.8318	0.9462	0.9537	0.9584	**0.9627**	13,1	25
22	Chandrasekharan & Rajagopalan (1987)	40x100	0.9475	0.9591	0.9806	0.981	**0.9813**	16,88	17
23	Yang & Yang (2008)	46x105	0.9098	0.9520	0.961	0.9618	**0.9629**	23,9	18
24	Liang & Zolfaghari (1999)	50x150	0.9305	0.9292	0.9608	0.9617	**0.9627**	51,66	24

[a] This solution has a mistake.

6 References

[1] Amit N. (2004). The bicluster graph editing problem. Master thesis. Tel Aviv University, 50 p.

[2] Askin, R.G., Cresswell, S.H., Goldberg, J.B. and Vakharia, A.J., 1991. A Hamiltonian path approach to reordering the part-machine matrix for cellular manufacturing. International Journal of Production Research, 29(6), pp.1081-1100.

[3] Ballakur A. (1985). An Investigation of Part Family/Machine Group Formation for Designing Cellular Manufacturing Systems. Ph.D. Thesis, University Wisconsin, Madison.

[4] Ballakur, A., & Steudel, H. J. (1987). A within cell utilization based heuristic for designing cellular manufacturing systems. International Journal of Production Research, 25, 639–655.

[5] Batsyn, M., Batsyna E., Bychkov I. (2019) NP-completeness of cell formation problem with grouping efficacy objective, International Journal of Production Research (Q1), Published online: 26 Sep 2019, https://doi.org/10.1080/00207543.2019.1668072

[6] Bhatnagar, R., V. Saddikuti. (2010). Models for cellular manufacturing systems design: matching processing requirements and operator capabilities. J. Opl. Res. Soc. 61, 827-839

[7] Boutsinas, B. , 2013. Machine-part cell formation using biclustering. Eur. J. Oper. Res. 230, 563–572 .

[8] Brusco, M. J. (2015). An iterated local search heuristic for cell formation. Computers & Industrial Engineering, 90, 292-304.

[9] Brusco, M.J. (2015). An exact algorithm for maximizing grouping efficacy in part-machine clustering. IIE Trans. 47 (6), 653–671 .

[10] Burbidge, J. L. (1961). The "new approach" to production. Production Engineer, 40(12), 769-784.

[11] Burbidge, J. L. (1963). Production flow analysis. Production Engineer, 42(12), 742-752.

[12] Burbidge, J. L. (1971). Production flow analysis. Production Engineer, 50(4.5), 139-152.

[13] Burbidge, J. L. (1975). The introduction of group technology. Halsted Press.

[14] Bychkov, I., Batsyn, M., Sukhov, P., Pardalos, P.M.(2013) Heuristic Algorithm for the Cell Formation Problem. In: Goldengorin B. I., Kalyagin V. A., Pardalos P. M. (eds.) Models, Algorithms, and Technologies for Network Analysis. Springer Proceedings in Mathematics & Statistics 59, 43–69.

[15] Bychkov, I., Batsyn, M., Pardalos, P. (2014). Exact model for the cell formation problem. Optimization Letters 8(8), 2203–2210.

[16] Bychkov I., Batsyn M., Pardalos P.M. Heuristic for Maximizing Grouping Efficiency in the Cell Formation Problem (2017), in: Models, Algorithms, and Technologies for Network Analysis. Springer Proceedings in Mathematics & Statistics / Ed. by V. A. Kalyagin, A. I. Nikolaev, P. M. Pardalos, O. Prokopyev . Vol. 197. Springer, 2017.

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

[18] Busygin S., Prokopyev, O., Pardalos, P. M. (2008). Biclustering in data mining. Computers Operations Research 35(9), 2964–2987

[19] Busygin S., Prokopyev, O.A. and Pardalos, P.M., 2005. Feature selection for consistent biclustering via fractional 0–1 programming. Journal of Combinatorial Optimization, 10(1), pp.7-21.

[20] Chan F. T. S., Lau K. W., Chan L. Y., Lo V. H. Y. (2008). Cell formation problem with consideration of both intracellular and intercellular movements. International Journal of Production Research 46(10), 2589-2620.

[21] Chandrasekaran, M.P., Rajagopalan, R., (1986a). An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. International Journal of Production Research 24, 451-463.

[22] Chandrasekaran, M.P., Rajagopalan, R., (1986b). MODROC: An extension of rank order clustering of group technology. International Journal of Production Research 24 (5), 1221-1233.

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

[24] Chandrasekharan, M. P., & Rajagopalan, R. (1989). Groupability: An analysis of the properties of binary data matrices for group technology. International Journal of Production Research , 27(6), 1035-1052.

[25] Chung S.-H., Wu T.-H., Chang C.-C. (2011). An efficient tabu search algorithm to the cell formation problem with alternative routings and machine reliability considerations. Computers & Industrial Engineering 60(1), 7-15.

[26] Dinkelbach, W. , 1967. On nonlinear fractional programming. Manage. Sci. 13 (7), 4 92–498 .

[27] Dimopoulos C., Zalzala A. (2000). Recent developments in evolutionary computation for manufacturing optimization: problems, solutions and comparisons. IEEE Transactions on Evolutionary Computation 4(2), 93-113.

[28] Dolgui, A. and Proth, J.M., 2010. Supply chain engineering: useful methods and techniques. Springer Science & Business Media.

[29] Elbenani, B., Ferland, J. A. (2012). An exact method for solving the manufacturing cell formation problem. International Journal of Production Research, 50(15), 4038–4045.

[30] Goldengorin, B., Krushinsky, D., Slomp, J. (2012). Flexible PMP approach for large size cell formation. Operations Research, 60(5), 1157–1166

[31] Goldengorin B. , Krushinsky D., Pardalos P.M. Cell formation in industrial engineering: theory, algorithms and experiments, Springer Optimization and Its Applications, v.79, Springer-Verlag New York, 2013.

[32] Gonçalves J.F., Resende M.G.C(2004). An evolutionary algorithm for manufacturing cell formation. Computers & Industrial Engineering, 47(2-3),247-273.

[33] Hansen, P., Mladenović, N., Todosijević, R. and Hanafi, S. (2017). Variable neighborhood search: basics and variants. EURO Journal on Computational Optimization, 5(3), pp.423-454.

[34] Hartigan, J.A.(1972). Direct clustering of a data matrix. Journal of the American Statistical Association 67, 123–129

[35] Hartigan, J.A.(1975). Clustering Algorithms. Wiley, Chichester

[36] James, T. L., Brown, E. C., Keeling, K. B. (2007). A hybrid grouping genetic algorithm for the cell formation problem. Computers & Operations Research, 34(7), 2059–2079.

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

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

[39] Krushinsky, D., & Goldengorin, B. (1984). An exact model for cell formation in group technology. Computational Management Science, 9(3), 323-338.

[40] Kumar K. R., Kusiak A., Vannelli A. (1986). Grouping of parts and components in flexible manufacturing systems. European Journal of Operations Research, 24, 387-397.

[41] Kumar K.R., Vannelli A. (1987). Strategic subcontracting for efficient disaggregated manufacturing. International Journal of Production Research, 25(12), 1715-1728.

[42] Kumar, K. R., Chandrasekharan, M. P. (1990). Grouping efficacy: A quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. International Journal of Production Research, 28(2), 233–243.

[43] Kusiak, A. (1987). The generalized group technology concept. International Journal of Production Research, 25(4), 561–569.

[44] Mak K. L., Wong Y. S., Wang X. X. (2000). An Adaptive Genetic Algorithm for Manufacturing Cell Formation. The International Journal of Advanced Manufacturing Technology 16(7), 491-497.

[45] Mirkin, B. (1996). Mathematical Classification and Clustering, p. 448. Kluwer, Dordrecht.

[46] Mirkin, B. (2011). Approximate bicluster and tricluster boxes in the analysis of binary data. International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing, p 248-256, Springer, Berlin, Heidelberg

[47] Mitrofanov S.P.(1959). Nauchniye osnovy gruppovoi technologii.

[48] Mitrofanov, S.P.(1966). The Scientific Principles of Group Technology. Boston Spa, Yorkshire: National Lending Library Translation. Translation of Mitrofanov (1959).

[49] Mucherino, A. and Cafieri, S., 2010. A new heuristic for feature selection by consistent biclustering. arXiv preprint arXiv:1003.3279.

[50] Ng, S. (1993). Worst-case analysis of an algorithm for cellular manufacturing. European Journal of Operational Research, 69(3), 384–398.

[51] Ng, S. (1996). On the characterization and measure of machine cells in group technology. Operations Research, 44(5), 735–744.

[52] Paydar, M. M., Saidi-Mehrabad, M. (2013). A hybrid genetic-variable neighborhood search algorithm for the cell formation problem based on grouping efficacy. Computers & Operations Research, 40(4), 980–990.

[53] Pinheiro R. G. S., Martins I. C., Protti F., Ochi, L. S.; Simonetti L.G., & Subramanian A. (2016), On solving manufacturing cell formation via Bicluster Editing, European Journal of Operational Research, 254(3), 769 - 779

[54] Sarker, B. R. (2001). Measures of grouping efficiency in cellular manufacturing systems. European Journal of Operational Research, 130(3), 588–611.

[55] Solimanpur M., Saeedi S., Mahdavi I. (2010). Solving cell formation problem in cellular manufacturing using ant-colony-based optimization. The International Journal of Advanced Manufacturing Technology 50, 1135-1144.

[56] Srinivasan, G., Narendran, T.T. (1991). GRAFICS—a nonhierarchical clustering algorithm for group technology. The International Journal of Production Research, 29(3), pp.463-478.

[57] 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.

[58] Utkina, I., Batsyn, M., Batsyna, E. (2016b). A branch and bound algorithm for a fractional 0-1 programming problem. Lecture Notes in Computer Science, 9869, 244–255.

[59] Vannelli, A., Ravi Kumar, K., 1986. A method for finding minimal bottle-neck cells for grouping part-machine families. International Journal of Production Research, 24(2), pp.387-400.

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

[61] Wang, J., and Roze, C., 1997, Formation of machine cells and part families in cellular manufacturing: an experimental study. International Journal of Production Research, 35, 463-478

[62] Won, Y. (2000). Two-phase approach to GT cell formation using efficient p-median formulations. International Journal of Production Research, 38(7), 1601–1613.

[63] Zilinskas, J., Goldengorin, B., Pardalos, P. M. (2015). Pareto-optimal front of cell formation problem in group technology. Journal of Global Optimization 61(1), 91–108.

[64] http://opt-hub.com/problems/cfp

[65] https://www.researchgate.net/publication/316648108_Test_instances_and_solutions_for_the_cell_formation_problem