A Branch and Bound Algorithm for a Fractional 0-1 Programming Problem

Irina Utkina, Mikhail Batsyn[✉], and Ekaterina Batsyna

Department of Applied Mathematics and Informatics,
Laboratory of Algorithms and Technologies for Network Analysis,
National Research University Higher School of Economics,
136 Rodionova Street, Niznhy Novgorod, Russia
{iutkina,mbatsyn,batcyna}@hse.ru

Abstract. We consider a fractional 0-1 programming problem arising in manufacturing. The problem consists in clustering of machines together with parts processed on these machines into manufacturing cells so that intra-cell processing of parts is maximized and inter-cell movement is minimized. This problem is called Cell Formation Problem (CFP) and it is an NP-hard optimization problem with Boolean variables and constraints and with a fractional objective function. Because of its high computational complexity there are a lot of heuristics developed for it. In this paper we suggest a branch and bound algorithm which provides exact solutions for the CFP with a variable number of cells and grouping efficacy objective function. This algorithm finds optimal solutions for 21 of the 35 popular benchmark instances from literature and for the remaining 14 instances it finds good solutions close to the best known.

Keywords: Cell formation · Biclustering · Branch and bound · Upper bound · Exact solution

1 Introduction

The first work on the Group Technology in manufacturing was written by Flanders (1925). In Russia the Group Technology was introduced by Mitrofanov (1933). The main problem in the Group Technology (GT) is to find an optimal partitioning of machines and parts into manufacturing cells, in order to maximize intra-cell processing and minimize inter-cell movement of parts. Maximization of the so-called grouping efficacy is accepted in literature as a good objective combining these two goals (Kumar and Chandrasekharan 1990). This problem is called the Cell Formation Problem (CFP) (Goldengorin et al. 2013). CFP with grouping efficacy objective function is a fractional 0-1 programming problem.

Burbidge developed Product Flow Analysis (PFA) approach to this problem and described the GT and the CFP in his book (Burbidge 1961). Ballakur and Steudel (1987) have shown that the CFP is an NP-hard problem for different objective functions. That is why there have been developed a lot of heuristic approaches

Y. Kochetov et al. (Eds.): DOOR 2016, LNCS 9869, pp. 244–255, 2016.
DOI: 10.1007/978-3-319-44914-2_20

(Goncalves and Resende 2004; James et al. 2007; Bychkov et al. 2013, Paydar and Saidi-Mehrabad 2013) and almost no exact ones for the CFP with a variable number of cells and grouping efficacy objective function.

Kusiak et al. (1993) consider one of the most simple variants of the CFP called the machine partitioning problem in which it is necessary to partition only machines into the specified number of cells minimizing the total Hamming distance between machines inside the cells. The authors present an exact A* algorithm for this variant of the CFP. They also develop a branch and bound algorithm for the CFP with a variable number of cells, a limit on the number of machines inside each cell, and maximization of the size of so-called mutually separable cells as an objective function. Spiliopoulos and Sofianopoulou (1998) and Arket et al. (2012) also present branch and bound algorithms for the machine partitioning problem.

One of the recent exact approaches for the CFP with the grouping efficacy objective function is suggested by Elbenani and Ferland (2012). These authors suggest to reduce the fractional programming CFP problem to a number of ILP problems by means of Dinkelbach approach and to solve each ILP problem with CPLEX solver. Unfortunately they consider the CFP with a fixed number of cells which is much easier. They solve 27 of the 35 popular benchmark instances, but only for a fixed number of cells. The same simplified formulation of the CFP is considered by Brusco (2015). The author develops a branch and bound algorithm and solves 31 of the 35 instances, but again only for some fixed numbers of cells. For example problem 26 is solved only for 7 cells and it requires more than 15 days of computational time.

To the best of our knowledge the only existing exact approach to the CFP with a variable number of cells and grouping efficacy objective function is by Bychkov et al. (2014) who suggested a new approach to reduce the CFP problem to a small number of ILP problems and for the first time solved to optimality 14 of the 35 popular benchmark instances from literature using CPLEX software. Zilinskas et al. (2015) considered the CFP with a variable number of cells as a bi-objective optimization problem and developed an exact algorithm which finds Pareto frontier.

In this paper we suggest an efficient branch and bound algorithm for the CFP with a variable number of cells and grouping efficacy objective function. We are able to find optimal solutions for 21 of the 35 benchmark instances. Note also that the CFP is a biclustering problem in which we simultaneously cluster machines and parts into cells. So the suggested approach can be also applied to biclustering problems arising in data mining (Busygin et al. 2008).

2 Formulation

The objective of the CFP is to find an optimal partitioning of machines and parts into groups (production cells, or shops) in order to minimize the inter-cell movement of parts from one cell to another and to maximize intra-cell processing operations. The input data for this problem is matrix A which contains zeroes and ones. The size of this matrix is $m \times p$ which means that it has m machines and

p parts. The element a_{ij} of the input matrix is equal to one if part j should be processed on machine i. The objective is to minimize the number of zeroes inside cells and the number of ones outside cells. There have been suggested several objective functions which combine these two goals. The objective function which provides a good combination of these goals and is widely accepted in literature is the grouping efficacy suggested by Kumar and Chandrasekharan (1990):

$$f = \frac{n_1^{in}}{n_1 + n_0^{in}} \to \max, \tag{1}$$

where n_1 is the number of ones in the input matrix, n_1^{in} is the number of ones inside cells, n_0^{in} is the number of zeroes inside cells.

The mathematical programming model for the CFP is the following (see also Bychkov et al. (2014)).
Decision variables:

$$x_{ik} = \begin{cases} 1 & \text{if machine } i \text{ is assigned to cell } k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$y_{jk} = \begin{cases} 1 & \text{if part } j \text{ is assigned to cell } k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Objective function:

$$\max \frac{n_1^{in}}{n_1 + n_0^{in}} \tag{4}$$

Constraints:

$$n_1^{in} = \sum_{k=1}^{c} \sum_{i=1}^{m} \sum_{j=1}^{p} a_{ij} x_{ik} y_{jk} \tag{5}$$

$$n_0^{in} = \sum_{k=1}^{c} \sum_{i=1}^{m} \sum_{j=1}^{p} (1 - a_{ij}) x_{ik} y_{jk} \tag{6}$$

$$\sum_{k=1}^{c} x_{ik} = 1 \quad \forall i = 1, \ldots, m \tag{7}$$

$$\sum_{k=1}^{c} y_{jk} = 1 \quad \forall j = 1, \ldots, p \tag{8}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{p} x_{ik} y_{jk} \geq \sum_{i=1}^{m} x_{ik} \quad \forall k = 1, \ldots, c \tag{9}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{p} x_{ik} y_{jk} \geq \sum_{j=1}^{p} y_{jk} \quad \forall k = 1, \ldots, c \tag{10}$$

Here $c = \min(m, p)$ is the maximum possible number of cells. Constrains (7) and (8) require that every machine and every part is assigned to exactly one cell. Constrains (9) and (10) require that there are no cells having only machines without parts or only parts without machines.

3 Branch and Bound Algorithm

3.1 Branching

Because of the biclustering structure of the CFP our branching goes by two parameters. The suggested algorithm has branching on machines and parts sequentially changing each other: machines-parts-machines-... We use vectors $M(1 \times m)$ and $P(1 \times p)$ for this purpose. Element M_i contains the cell to which machine i is assigned and element P_j contains the cell to which part j is assigned. For example M = [1231] and P = [11321] mean that cell 1 contains machines 1, 4 and parts 1, 2, 5, cell 2 contains machine 2 and part 4, and cell 3 contains machine 3 and part 3.

Branching on machines makes changes in vector M. It starts from assigning the first machine to cell 1. Let k be the number of cells in the current partial solution. When the algorithm branches on machines, it takes the first machine which is not assigned to any cell and tries to assign it to the existing cells with numbers from 1 to k or creates a new cell $(k + 1)$ for this machine.

Branching on parts makes changes in vector P. It starts with all zeroes inside P which means that no parts are assigned to any cell. When the algorithm branches on parts it takes the first part which is not assigned to any cell and tries to assign it to the existing cells from 1 to k or to a new cell $(k + 1)*$ (star means that the number of the cell can be $k + 1$ or greater) if there are some unassigned machines which can be also added later to this new cell. We assume that the number of parts is greater than the number of machines.

The algorithm branches on parts and machines successively. It starts with $M = [100 \ldots 0]$ and $P = [00 \ldots 0]$. Next it changes vector P, then - vector M and so on. This way the algorithm builds the search tree. The leaves of the search tree contain complete solutions and other nodes contain partial solutions. The complete search tree depends only on the number of machines and parts. It contains all feasible solutions as its leaves.

To provide an efficient branching, before choosing a branch we calculate an upper bound for each branch and choose the branch with the greatest value of the upper bound. This branching strategy allows us to find good solutions earlier.

3.2 Upper Bound

To obtain an upper bound for a given partial solution we relax the original CFP problem and suggest a polynomial algorithm to calculate an optimal solution or an upper bound for the relaxed problem. The relaxed problem is formulated as follows. We are given a partial solution in which some of the machines and parts are already assigned to some cells. For example in Table 1 machines 1, 2 with parts 1, 2, 3, 4 are assigned to cell 1, and machine 3 with part 5 is assigned to cell 2. The objective is to assign the remaining machines independently on each other to the existing cells or to a new cell, and assign the remaining parts to the existing cells taking into account only the rows already assigned in the given partial solution. In the relaxed problem we allow an independent assignment of machines and parts to cells. In this case the best assignment for machine 4 will be to put it to cell 1 with parts 1, 2, 3, 4, 7. This will bring 4 ones and 1 zero inside cells. The best assignment for machine 5 will be to put it to a new cell 3 with parts 7, 8. This will bring 2 ones and 0 zeroes inside cells. The best assignment for parts 6 and 7 which takes into account only rows 1, 2, 3 will be to put it to cell 2 (with machine 3). The best assignment for part 8 which takes into account only rows 1, 2, 3 will be to put it to cell 1 (with machines 1, 2). This optimal solution for the relaxed problem is shown in Table 2. This solution is infeasible for the original CFP problem because independent assignment of machines and parts is allowed and as a result we obtain non-rectangular cells which can also intersect by columns. Since it is an optimal solution to the relaxed problem it provides an upper bound to the original problem. In our example for the partial solution we have $f = \frac{8}{21+1} \approx 0.36$ and the solution of the relaxed problem gives us an upper bound to the complete solution of the CFP equal to $UB = \frac{8+10}{21+1} \approx 0.82$.

In our example from Table 1 it is not obvious whether the chosen alternative $(a_1, b_1) = (4, 1)$ (putting 4 ones and 1 zero inside cells) is better than alternative $(a_2, b_2) = (1, 0)$ for machine 3. To choose between two alternatives we use the following theorem.

Theorem 1. *If the unknown maximum value of the objective function $\frac{a}{b}$ for the relaxed CFP problem without assignment of machine i (considering all its ones and zeroes to be outside cells) can be estimated as $\frac{a}{b} \in [l, u]$, $b \in [b_l, b_u]$, then alternative (a_1, b_1) for machine i is better than alternative (a_2, b_2) if:*

$$b_l \left(l - \frac{\Delta a}{\Delta b} \right) \geq b_1 \frac{\Delta a}{\Delta b} - a_1 \tag{11}$$

Table 1. A partial solution for the CFP

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

Table 2. Optimal solution for the relaxed CFP

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

and is worse than (a_2, b_2) *if:*

$$b_u\left(u - \frac{\Delta a}{\Delta b}\right) \leq b_1\frac{\Delta a}{\Delta b} - a_1 \tag{12}$$

Here $\Delta a = a_2 - a_1, \Delta b = b_2 - b_1 > 0$ *(if* $\Delta b < 0$ *we can always swap the alternatives).*

Proof. Alternative (a_1, b_1) is better than alternative (a_2, b_2) if:

$$\frac{a + a_1}{b + b_1} \geq \frac{a + a_2}{b + b_2} \tag{13}$$

Multiplying this inequality by the positive denominators and making simple transformations (here we need $\Delta b > 0$) we get the equivalent inequality:

$$b\left(\frac{a}{b} - \frac{\Delta a}{\Delta b}\right) \geq b_1\frac{\Delta a}{\Delta b} - a_1 \tag{14}$$

Using the given estimations for the unknown maximum value of the objective function $\frac{a}{b}$ we have the following bounds for the left-hand side of this inequality:

$$b_l\left(l - \frac{\Delta a}{\Delta b}\right) \leq b\left(\frac{a}{b} - \frac{\Delta a}{\Delta b}\right) \leq b_u\left(u - \frac{\Delta a}{\Delta b}\right)$$

So if inequality (11) is true then we immediately have that inequalities (14) and (13) are true, which means that alternative (a_1, b_1) is better than alternative (a_2, b_2). Otherwise if inequality (12) is true then we immediately have that the opposite inequalities are true:

$$b\left(\frac{a}{b} - \frac{\Delta a}{\Delta b}\right) \leq b_1\frac{\Delta a}{\Delta b} - a_1, \quad \frac{a + a_1}{b + b_1} \leq \frac{a + a_2}{b + b_2}$$

This means that alternative (a_2, b_2) is better in this case. □

Note that in case $\Delta b = 0$ it is obvious which of the two alternatives is better. It is also not difficult to estimate the unknown maximum value of the objective function $\frac{a}{b}$ for the relaxed CFP problem without assignment of machine i (considering all its ones and zeroes to be outside cells). Let a_c, b_c be the current values of the objective function numerator $a_c = n_1^{in}$ and denominator $b_c = n_1 + n_0^{in}$ for the current partial solution. Then in the worst case we can get $\frac{a}{b} = \frac{a_c}{b_c}$ because every unassigned machine can be put to a new cell putting some ones and no zeroes inside it, and every part can be left unassigned. So the lower bound is $l = \frac{a_c}{b_c}$. In the best case we can put inside cells no zeroes and all the ones except $\bar{n}_1^{out}$ ones which lie in the already assigned area and cannot get inside cells (see gray area (the darkest area on black-and-white printing) in Table 1) and except n_1^i ones which lie in row i. So the upper bound is $u = (n_1 - \bar{n}_1^{out} - n_1^i)/b_c$. The value of the denominator b can be estimated as: $b_c \leq b \leq n_1 + n_0 - \bar{n}_0^{out} - n_0^i$. Here n_1 and n_0 are the number of ones and zeroes in the input matrix, $\bar{n}_0^{out}$ is the number of zeroes which lie in the already assigned area (gray area in Table 1) and n_0^i is the number of zeroes in row i.

Below we present a polynomial algorithm of calculating the suggested upper bound as an optimal solution of the relaxed CFP problem, if we can always choose the best alternative for every machine and part, or as an upper bound to this solution otherwise. We illustrate the algorithm on the instance shown in Table 3.

Algorithm 1. Algorithm to choose between two alternatives

```
function CompareAlternatives($a_1, b_1, a_2, b_2, n_1, n_0, n_1^{in}, n_0^{in}, \bar{n}_1^{out}, \bar{n}_0^{out}, n_1^i, n_0^i$)
    $\Delta a \leftarrow a_2 - a_1, \Delta b \leftarrow b_2 - b_1$                ▷ $\Delta b$ should be non-negative
    if ($\Delta b = 0$) then
        if ($\Delta a < 0$) then
            return 1
        else if ($\Delta a > 0$) then
            return 2
        else
            return 0
    $a_c \leftarrow n_1^{in}, b_c \leftarrow n_1 + n_0^{in}, b_l \leftarrow b_c, b_u \leftarrow n_1 + n_0 - \bar{n}_0^{out} - n_0^i, l \leftarrow \frac{a_c}{b_c}, u \leftarrow \frac{n_1 - \bar{n}_1^{out} - n_1^i}{b_c}$
    if $b_l \left(l - \frac{\Delta a}{\Delta b}\right) \geq b_1 \frac{\Delta a}{\Delta b} - a_1$ then
        return 1
    if $b_u \left(u - \frac{\Delta a}{\Delta b}\right) \leq b_1 \frac{\Delta a}{\Delta b} - a_1$ then
        return 2
    return -1
```

Table 3. Example for upper bound calculation

	1	2	3	4	5	6	7	8	9
1	1	1	1	1	1	0	0	0	0
2	1	1	1	1	0	0	0	0	1
3	0	0	0	0	0	1	1	0	0
4	0	1	1	0	0	0	0	1	1
5	0	0	0	0	1	1	0	0	1

1. Calculate the number of ones n_1^{in} and zeroes n_0^{in} inside the cells of the given partial solution and the number of ones $\bar{n}_1^{out}$ and zeroes $\bar{n}_0^{out}$ which cannot get inside cells in any solution (see gray area (the gray area with black zeroes on black-and-white printing) in Table 3). The total number of ones n_1 and zeroes n_0 are constant. From these values we get the numerator $a_c = n_1^{in}$ and the denominator $b_c = n_1 + n_0^{in}$ for the efficacy $f = a_c/b_c$ of the current partial solution. For the example in Table 3 we have: $n_1^{in} = 11, n_0^{in} = 1, \bar{n}_1^{out} = 0, \bar{n}_0^{out} = 9, n_1 = 19, n_0 = 26, a_c = 11, b_c = 20$.

Table 4. Solution providing the upper bound

	1	2	3	4	5	6	7	8	9
1	1	1	1	1	1	0	0	0	0
2	1	1	1	1	0	0	0	0	1
3	0	0	0	0	0	1	1	0	0
4	0	1	1	0	0	0	0	1	1
5	0	0	0	0	1	1	0	0	1

2. For every unassigned machine (row) using Algorithm 1 we compare all possible alternatives of adding it to one of the existing cells or to a new cell.
   For our example we have 3 alternatives for machine $i = 4$: 1) $(4, 3)$ - add it to cell 1 with parts 1, 2, 3, 4, 5, 8, 9 putting 4 ones and 3 zeroes inside this cell; 2) $(2, 2)$ - add it to cell 2 with parts 6, 7, 8, 9 putting 2 ones and 2 zeroes inside; 3) $(2, 0)$ - add it to a new cell 3 with parts 8, 9 putting 2 ones and 0 zeroes inside. Obviously alternative 3 is better than alternative 2. So we need to compare only two alternatives $(a_1, b_1) = (2, 0)$ and $(a_2, b_2) = (4, 3)$. We have: $n_1^i = 4, n_0^i = 5, \Delta a = 2, \Delta b = 3, l = a_c/b_c = 11/20, u = (n_1 - \bar{n}_1^{out} - n_1^i)/b_c = 15/20, b_l = 20, b_u = n_1 + n_0 - \bar{n}_0^{out} - n_0^i = 31$. And the values we need to apply Algorithm 1 are:

$$b_1 \frac{\Delta a}{\Delta b} - a_1 = -2, \quad b_l \left(l - \frac{\Delta a}{\Delta b} \right) = -\frac{7}{3}, \quad b_u \left(u - \frac{\Delta a}{\Delta b} \right) = \frac{31}{12}$$

   So neither of the conditions in Algorithm 1 is satisfied and we cannot determine which alternative is better (Algorithm 1 returns -1). In this case we build an alternative $(\max(a_1, a_2), \min(b_1, b_2))$, which is better than both incomparable alternatives, and use it to obtain an upper bound on the solution of the relaxed CFP problem. In our example it is alternative $(4, 0)$.
   Now for machine $i = 5$ we have: $n_1^i = 3, n_0^i = 6, l = 11/20, u = 16/20, b_l = 20, b_u = 30$. There are 3 alternatives $(2, 4)$, $(2, 1)$, and $(1, 0)$. It is clear that alternative $(2, 4)$ is worse than $(2, 1)$. For $(a_1, b_1) = (1, 0)$ and $(a_2, b_2) = (2, 1)$ we have:

$$b_1 \frac{\Delta a}{\Delta b} - a_1 = -1, \quad b_l \left(l - \frac{\Delta a}{\Delta b} \right) = -9, \quad b_u \left(u - \frac{\Delta a}{\Delta b} \right) = -6$$

   So $b_u \left(u - \frac{\Delta a}{\Delta b} \right) \leq b_1 \frac{\Delta a}{\Delta b} - a_1$ and Algorithm 1 returns alternative $(a_2, b_2) = (2, 1)$. Thus $(2, 1)$ is the best alternative for machine 5.
3. For every unassigned part (column) in the same way using Algorithm 1 we compare all possible alternatives of adding it to one of the existing cells or leaving it unassigned. However in this case we take into account only ones and zeroes which lie in the rows already assigned in the given partial solution (blue area (the darkest area with white digits on black-and-white printing) in Table 3).
   In our example part 8 has only zeroes in this area and so it is better not to add it to any cell. For part 9 ($j = 9$) we have 3 alternatives: (1) $(1, 1)$ - add

it to cell 1 putting 1 one and 1 zero inside; (2) $(0,1)$ - add it to cell 2 putting 0 ones and 1 zero inside; (3) $(0,0)$ - do not add it to any cell. It is clear that $(0,1)$ is a bad alternative and we need to compare only $(a_1,b_1)=(0,0)$ and $(a_2,b_2)=(1,1)$. We have $n_1^j=1, n_0^j=2, l=a_c/b_c=11/20, u=(n_1-\bar{n}_1^{out}-n_1^j)/b_c=18/20, b_l=20, b_u=n_1+n_0-\bar{n}_0^{out}-n_0^j=34$. The values needed to apply Algorithm 1 are:

$$b_1\frac{\Delta a}{\Delta b}-a_1=0, \quad b_l\left(l-\frac{\Delta a}{\Delta b}\right)=-9, \quad b_u\left(u-\frac{\Delta a}{\Delta b}\right)=-3.4$$

So $b_u\left(u-\frac{\Delta a}{\Delta b}\right)\leq b_1\frac{\Delta a}{\Delta b}-a_1$ and Algorithm 1 returns alternative $(a_2,b_2)=(1,1)$ as the best alternative for part 9.

4. We calculate the upper bound by putting inside all the ones and zeroes corresponding to the best alternatives chosen for all unassigned machines and parts. For our example the corresponding solution which gives an upper bound to the relaxed CFP problem (and thus to the original CFP problem also) is shown in Table 4. For this example we have $UB=\frac{11+4+2+1}{19+0+1+1}=\frac{18}{21}\approx 0.86$.

4 Results

The suggested branch and bound algorithm has been able to solve 21 of 35 popular benchmark instances from the literature exactly and to find good solutions for the remaining 14 instances. The results are presented in Table 5. All computations were run on Intel Core i7 with 16 Gb RAM. Note that the algorithm was run without any initial solution while in the results reported by Bychkov et al. (2014) the best-known solutions were used as initial.

The results show that the developed algorithm is more efficient than the approach suggested by Bychkov et al. (2014). Bychkov et al. (2014) approach is able to solve to optimality only instances 1–13, where it is much slower (up to 1000 times) than our new algorithm, and also instance 22 which is very simple and is solved withing 0.01 s by both approaches. The suggested algorithm is able to solve to optimality 7 instances more (instances 14, 15, 16, 17, 20, 23, 24) which are very hard and have never been solved before to the best of our knowledge. All instances marked with footnote ‘b’ in Table 5 (21 instance) have not been solved to optimality by Bychkov et al. (2014). As it is noted in this footnote in this approach the problem is divided into a number of IP subproblems and a time limit of 300 s is set for every subproblem. That is why all these instances have different times after which the computation has been stopped without reaching an optimal solution. All instances marked with footnote ‘a’ in Table 5 (14 instances) have not been solved to optimality by the new approach. The time limit was set to 100000 s for all these instances.

Table 5. Results

#	Name	Size	Best-known solution	f	Time, s	Bychkov et al. (2014), Time, s
1	King and Nakornchai (1982)	5 × 7	0.8235	0.8235	0.00	0.63
2	Waghodekar and Sahu (1984)	5 × 7	0.6957	0.6957	0.00	2.29
3	Seifoddini (1989)	5 × 18	0.7959	0.7959	0.00	5.69
4	Kusiak (1987)	6 × 8	0.7692	0.7692	0.00	1.86
5	Kusiak and Chow (1987)	7 × 11	0.6087	0.6087	0.00	9.14
6	Boctor (1991)	7 × 11	0.7083	0.7083	0.00	5.15
7	Seifoddini and Wolfe (1986)	8 × 12	0.6944	0.6944	0.00	13.37
8	Chandrasekharan and Rajagopalan (1986a)	8 × 20	0.8525	0.8525	0.00	18.33
9	Chandrasekharan and Rajagopalan (1986b)	10 × 10	0.5872	0.5872	0.19	208.36
10	Mosier and Taube (1985a)	10 × 15	0.7500	0.7500	0.00	6.25
11	Chan and Milner (1982)	10 × 15	0.9200	0.9200	0.00	2.93
12	Askin and Subramanian (1987)	14 × 24	0.7206	0.7206	2.89	259.19
13	Stanfel (1985)	14 × 24	0.7183	0.7183	5.51	259.19
14	McCormick et al. (1972)	16 × 24	0.5326	0.5326	97117.43	[b]20829.38
15	Srinivasan et al. (1990)	16 × 30	[c]0.6899	0.6899	837.93	[b]13719.99
16	King (1980)	16 × 43	0.5753	0.5753	7045.64	[b]24930.93
17	Carrie (1973)	18 × 24	0.5773	0.5773	5668.25	[b]13250.01
18	Mosier and Taube (1985b)	20 × 20	0.4345	[a]0.4211	100000.00	[b]43531.77
19	Kumar et al. (1986)	20 × 23	0.5081	[a]0.4697	100000.00	[b]33020.13
20	Carrie (1973)	20 × 35	0.7791	0.7791	88.62	[b]11626.98
21	Boe and Cheng (1991)	20 × 35	0.5798	[a]0.5615	100000.00	[b]33322.08
22	Chandrasekharan and Rajagopalan (1989)	24 × 40	1.0000	1.0000	0.00	0.00
23	Chandrasekharan and Rajagopalan (1989)	24 × 40	0.8511	0.8511	33.70	[b]6916.24
24	Chandrasekharan and Rajagopalan (1989)	24 × 40	0.7351	0.7351	86007.93	[b]14408.88
25	Chandrasekharan and Rajagopalan (1989)	24 × 40	0.5329	[a]0.5185	100000.00	[b]34524.47
26	Chandrasekharan and Rajagopalan (1989)	24 × 40	0.4895	[a]0.4648	100000.00	[b]41140.94
27	Chandrasekharan and Rajagopalan (1989)	24 × 40	0.4726	[a]0.4468	100000.00	[b]44126.76
28	McCormick et al. (1972)	27 × 27	0.5482	[a]0.5017	100000.00	[b]22627.28
29	Carrie (1973)	28 × 46	0.4706	[a]0.4569	100000.00	[b]71671.08
30	Kumar and Vannelli (1987)	30 × 41	0.6331	[a]0.5942	100000.00	[b]22594.20
31	Stanfel (1985)	30 × 50	0.6012	[a]0.5789	100000.00	[b]31080.82
32	Stanfel (1985)	30 × 50	0.5083	[a]0.4860	100000.00	[b]48977.01
33	King and Nakornchai (1982)	30 × 90	0.4775	[a]0.4684	100000.00	[b]99435.64
34	McCormick et al. (1972)	37 × 53	0.6064	[a]0.5680	100000.00	[b]47744.04
35	Chandrasekharan and Rajagopalan (1987)	40 × 100	0.8403	[a]0.8403	100000.00	[b]24167.76

[a] The problem is not solved to optimality by our algorithm within the time limit of 100000 s.
[b] The problem is not solved to optimality by Bychkov et al. (2014). In this approach the problem is divided into a number of IP subproblems and a time limit of 300 s is set for every subproblem.
[c] A greater value is reported in some papers on heuristics probably due to an incorrect input matrix.

Acknowledgment. This research is supported by Laboratory of Algorithms and Technologies for Network Analysis, NRU HSE.

References

Arkat, J., Abdollahzadeh, H., Ghahve, H.: A new branch and bound algorithm for cell formation problem. Appl. Math. Model. **36**, 5091–5100 (2012)

Askin, R.G., Subramanian, S.P.: A cost-based heuristic for group technology configuration. Int. J. Prod. Res. **25**(1), 101–113 (1987)

Ballakur, A., Steudel, H.J.: A within cell utilization based heuristic for designing cellular manufacturing systems. Int. J. Prod. Res. **25**, 639–655 (1987)

Boctor, F.F.: A linear formulation of the machine-part cell formation problem. Int. J. Prod. Res. **29**(2), 343–356 (1991)

Boe, W., Cheng, C.H.: A close neighbor algorithm for designing cellular manufacturing systems. Int. J. Prod. Res. **29**(10), 2097–2116 (1991)

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

Burbidge, J.L.: The new approach to production. Prod. Eng. **40**(12), 3–19 (1961)

Busygin, S., Prokopyev, O., Pardalos, P.M.: Biclustering in data mining. Comput. Oper. Res. **35**(9), 2964–2987 (2008)

Bychkov, I., Batsyn, M., Sukhov, P., Pardalos, P.M.: 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, vol. 59, pp. 43–69. Springer, New York (2013)

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

Carrie, S.: Numerical taxonomy applied to group technology and plant layout. Int. J. Prod. Res. **11**, 399–416 (1973)

Chan, H.M., Milner, D.A.: Direct clustering algorithm for group formation in cellular manufacture. J. Manuf. Syst. **1**(1), 64–76 (1982)

Chandrasekharan, M.P., Rajagopalan, R.: MODROC: an extension of rank order clustering for group technology. Int. J. Prod. Res. **24**(5), 1221–1233 (1986a)

Chandrasekharan, M.P., Rajagopalan, R.: An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. Int. J. Prod. Res. **24**(2), 451–464 (1986b)

Chandrasekharan, M.P., Rajagopalan, R.: ZODIAC: an algorithm for concurrent formation of part families and machine cells. Int. J. Prod. Res. **25**(6), 835–850 (1987)

Chandrasekharan, M.P., Rajagopalan, R.: Groupability: analysis of the properties of binary data matrices for group technology. Int. J. Prod. Res. **27**(6), 1035–1052 (1989)

Elbenani, B., Ferland, J. A. Cell Formation Problem Solved Exactly with the Dinkelbach Algorithm. Montreal, Quebec. CIRRELT-2012-07, 1–14(2012)

Flanders, R.E.: Design manufacture and production control of a standard machine. Trans. ASME **46**, 691–738 (1925)

Goldengorin, B., Krushinsky, D., Pardalos, P.M.: Cell Formation in Industrial Engineering. Theory, Algorithms and Experiments. Springer Optimization and Its Applications, vol. 79, p. 206. Springer, New York (2013)

Goncalves, J.F., Resende, M.G.C.: An evolutionary algorithm for manufacturing cell formation. Comput. Ind. Eng. **47**, 247–273 (2004)

James, T.L., Brown, E.C., Keeling, K.B.: A hybrid grouping genetic algorithm for the cell formation problem. Comput. Oper. Res. **34**(7), 2059–2079 (2007)

King, J.R.: Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm. Int. J. Prod. Res. **18**(2), 213–232 (1980)

King, J.R., Nakornchai, V.: Machine-component group formation in group technology: review and extension. Int. J. Prod. Res. **20**(2), 117–133 (1982)

Kumar, K.R., Kusiak, A., Vannelli, A.: Grouping of parts and components in flexible manufacturing systems. Eur. J. Oper. Res. **24**, 387–397 (1986)

Kumar, K.R., Chandrasekharan, M.P.: Grouping efficacy: a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology. Int. J. Prod. Res. **28**(2), 233–243 (1990)

Kumar, K.R., Vannelli, A.: Strategic subcontracting for efficient disaggregated manufacturing. Int. J. Prod. Res. **25**(12), 1715–1728 (1987)

Kusiak, A.: The generalized group technology concept. Int. J. Prod. Res. **25**(4), 561–569 (1987)

Kusiak, A., Chow, W.S.: Efficient solving of the group technology problem. J. Manuf. Syst. **6**(2), 117–124 (1987)

Kusiak, A., Boe, J.W., Cheng, C.: Designing cellular manufacturing systems: branch-and-bound and A* approaches. IIE Transactions **25**(4), 46–56 (1993)

McCormick, W.T., Schweitzer, P.J., White, T.W.: Problem decomposition and data reorganization by a clustering technique. Oper. Res. **20**(5), 993–1009 (1972)

Mitrofanov, S.P.: Nauchnye osnovy gruppovoy tekhnologii. Lenizdat, Leningrad, Russia, 435 pages (1933). (in Russian)

Mosier, C.T., Taube, L.: The facets of group technology and their impact on implementation. OMEGA **13**(6), 381–391 (1985a)

Mosier, C.T., Taube, L.: Weighted similarity measure heuristics for the group technology machine clustering problem. OMEGA **13**(6), 577–583 (1985b)

Paydar, M.M., Saidi-Mehrabad, M.: A hybrid genetic-variable neighborhood search algorithm for the cell formation problem based on grouping efficacy. Comput. Oper. Res. **40**(4), 980–990 (2013)

Seifoddini, H.: A note on the similarity coefficient method and the problem of improper machine assignment in group technology applications. Int. J. Prod. Res. **27**(7), 1161–1165 (1989)

Seifoddini, H., Wolfe, P.M.: Application of the similarity coefficient method in group technology. IIE Transactions **18**(3), 271–277 (1986)

Spiliopoulos, K., Sofianopoulou, S.: An optimal tree search method for the manufacturing systems cell formation problem. Eur. J. Oper. Res. **105**, 537–551 (1998)

Srinivasan, G., Narendran, T.T., Mahadevan, B.: An assignment model for the part-families problem in group technology. Int. J. Prod. Res. **28**(1), 145–152 (1990)

Stanfel, L.: Machine clustering for economic production. Eng. Costs Prod. Econ. **9**, 73–81 (1985)

Waghodekar, P.H., Sahu, S.: Machine-component cell formation in group technology MACE. Int. J. Prod. Res. **22**, 937–948 (1984)

Zilinskas, J., Goldengorin, B., Pardalos, P.M.: Pareto-optimal front of cell formation problem in group technology. J. Global Optim. **61**(1), 91–108 (2015)