Optimality, flexibility and efficiency for cell formation in group technology

Krushynskyi, Dmytro

Optimality, flexibility and efficiency for cell formation in group technology

Dmytro Krushynskyi

15th June 2012

Publisher: University of Groningen
Groningen
The Netherlands

Printed by: Ipskamp Drukkers B.V.

ISBN: 978-90-367-5625-9
978-90-367-5624-2 (e-book)

rijksuniversiteit
groningen

Optimality, flexibility and efficiency for cell formation in group technology

Proefschrift

ter verkrijging van het doctoraat in de
Economie en Bedrijfskunde
aan de Rijksuniversiteit Groningen
op gezag van de
Rector Magnificus, dr. E. Sterken,
in het openbaar te verdedigen op
donderdag 12 juli 2012
om 14:30 uur

door

Dmytro Krushynskyi

geboren op 16 september 1983
te Kiev, Oekraïne

Promotores:	Prof. dr. R.H. Teunter
	Prof. dr. B. Goldengorin
	Prof. dr. ir. J. Slomp
Beoordelingscommissie:	Prof. dr. N. Suresh
	Prof. dr. E. de Klerk
	Prof. dr. K.J. Roodbergen

Acknowledgements

The current dissertation contains the output from a period of almost four years starting in September 2008, when I was doing my PhD project at the Department of Operations of the University of Groningen and the Research School SOM. Though this thesis bears a single name on its cover, it must definitely be attributed to much more people who contributed directly or indirectly to it.

First of all, I would like to express my thanks to my supervisor, Boris Goldengorin, for giving me an opportunity to pursue this PhD and for his guidance, especially during the first years. Our discussions about the properties of the *p*-Median problem gave rise, among others, to the two longest chapters of this thesis. Secondly, I would like to express my sincere thanks to my another supervisor Jannes Slomp who was supporting, encouraging and challenging me throughout the last 3 years of my PhD. All the applied side of the dissertation results from our exciting discussions about manufacturing practice and visits to companies. I am also thankful to Ruud Teunter who assisted me in resolving certain issues and provided a valuable feedback on this thesis. Finally, I would like to thank Sven de Vries who was involved in the supervision during my very first months in Groningen.

I would like to express my gratitude to my committee members, Professors Nallan Suresh of the State University of New York at Buffalo, USA, Etienne de Klerk of the Tilburg University, The Netherlands, and Kees Jan Roodbergen of the University of Groningen, The Netherlands, for their willingness to read this dissertation and their comments.

I am also very grateful to the members of the Research School SOM who were helping me with all kinds of administrative issues. These include Rina Koning, Astrid Beerta, Arthur de Boer, Ellen Nienhuis (Ellen, thanks for your perpetual willingness to help), and former and current PhD coordinators Martin Land and Linda Toolsema, respectively. I also appreciate the assistance of the OPERA secretaries Linda Henriquez-Peterson and Marjo Mejer.

My stay in Groningen would have never been so pleasant without my colleagues and friends, such as Bertus Talsma with whom I shared the office for more than 3 years, Matthijs Streutker (thanks for helping me in finding answers to my questions), Harmen Bouma and Serra Caner who were my neighbours and companions during the LNMB courses (Harmen, thank you also for correcting the Dutch summary of the thesis), my "buddy" Ernst Osinga who helped me to get used to the new environment upon my arrival to Groninegn, Erik Soepenberg who was my officemate for the last months of my PhD, members of the PhD committee (thanks for organising yearly PhD outings and other nice events) and many others. A separate line is devoted to my russian-speaking friends: Stanislav Stakhovych, Umed Temurshoev and Ilya Voskoboynikov, and their wives Oxana, Adiba and Elona, respectively.

Finally, I would like to express my sincere thanks and gratitude to my parents who supported and guided me through my life, and have ignited my interest in science and research, to my wife Anastasia who has been taking care of me since her arrival to Groningen and stimulated me to learn Dutch, and to my baby-daughter Liza who has brought new colours to my life and was inspiring me to work more efficiently such that I could spend more time with her.

Dmitry Krushinsky
Groningen, May 2012

Contents

List of abbreviations and notations

$\mathbf{A}, C, D$	matrices
$\mathcal{B}_{C,p}(\mathbf{y})$	p-truncated pseudo-Boolean polynomial derived from costs matrix C (PMP)
CF	cell formation
CM	cellular manufacturing
Δ	differences matrix (PMP)
$G(V, E)$	undirected graph with the set of vertices V and the set of edges E
$G(V \cup U, E)$	undirected bipartite graph with partite sets of vertices V and U, and the set of edges E
GT	group technology
i, j, k, l	indices; k usually enumerates cells, clusters, subgraphs
m, n, r	dimensions of input data: m is a number of machines (CF) or potential locations (PMP), n is a number of clients (PMP) or vertices in the input graph (MINpCUT), r is a number of parts (CF)
MILP	mixed-integer linear program(ming)
MINpCUT	the minimum multicut problem aimed at decomposing the input graph into p nonempty components
MPIM	machine-part incidence matrix
MST	minimum spanning tree
Π	permutation matrix (PMP)
p	the number of cells, clusters, subgraphs
pBp	pseudo-Boolean polynomial; a polynomial of Boolean variables with real coefficients
PMP	the p-Median problem
$\mathbb{R}$	the set of real numbers
$\mathbb{R}_+$	the set of nonnegative real numbers
$v_{ik}, x_{ij}, y_i, z_{ijk}$	decision variables
$\mathbf{y}$	vector of Boolean decision variables $\mathbf{y} = (y_1, \ldots, y_m)^{\mathrm{T}}$

Chapter 1

The problem of cell formation: introduction and approaches

This thesis focuses on a development of optimal, flexible and efficient models for cell formation in group technology. By optimality we mean guaranteed quality of the solutions provided by the model[1], by flexibility – possibility of taking additional constraints and objectives into account, by efficiency – reasonable running times (e.g., taking into account that cells are reconfigured infrequently, the times of 1 sec. and 10 min. are equally acceptable). The main aim is, thus, to provide a reliable tool that can be used by managers to design manufacturing cells based on their own preferences and constraints imposed by a particular manufacturing system.

The general structure of the thesis is as follows. The first chapter contains the prerequisites, necessary for understanding the cell formation problem and the gaps in the corresponding research. Those already familiar with the problem may safely skip some sections (e.g. the one describing existing approaches). The following three chapters are focused on development of the mathematical models for cell formation, Chapter 2 being very technical and focusing on theoretical properties of a proposed model. Chapter 5 considers alternative objectives for cell formation. Finally, Chapter 6 summarises the thesis and provides directions for further research.

[1] We allow for suboptimal solutions in case they are guaranteed close to the optimum. Thus, our notion of optimality differs from the one used in mathematical programming, where optimality means that no better solution exists.

1.1 Introduction

One of the possibilities for obtaining higher profit in a manufacturing system is lowering production costs (while preserving the production volumes). This, in turn, can be achieved by minimizing flow costs that include transportation costs, idle times of machines and costs of manpower needed to deliver parts that are being processed from one machine to another. The paradigm in industrial engineering called *group technology* (GT) was first developed in the former USSR (see, e.g., Mitrofanov, 1946, 1966) and is aimed at making the manufacturing system more efficient by improving the mentioned above factors. The main idea behind group technology is that similar things should be done similarly. One of the key issues in this concept is *cell formation* (CF) that suggests grouping machines into manufacturing units (cells) and parts into product families such that a particular product family is processed mainly within one cell. Such grouping becomes possible by exploiting similarities in the manufacturing processes for different parts, and increases the throughput of the manufacturing system without sacrificing the products quality. This can be viewed as decomposing the manufacturing system into a number of almost independent subsystems that can be managed separately. Clearly, such a decomposition is beneficial from the perspective of workload control and scheduling (especially, taking into account that most scheduling problems are computationally intractable). The degree of subsystems independence corresponds to the *amount of intercell movement* – the number of parts that must be processed in more than one subsystem (by more than one manufacturing cell).

The problem of cell formation can be traced back to the works of Flanders (1925) and Sokolovski (1937) but is oftenly attributed to Mitrofanov's group technology (Mitrofanov, 1959, 1966) and Burbidge's product flow analysis (PFA, see Burbidge, 1961). Burbidge showed that it can be reduced to a functional grouping of machines based on binary machine-part incidence data. Thus, in its simplest and earliest form cell formation is aimed at the functional grouping of machines based on similarity of the sets of parts that they process. Input data for such a problem is usually given by an $m \times r$ binary machine-part incidence matrix (MPIM) $\mathbf{A} = [a_{ij}]$, where $a_{ij} = 1$ if and only if j-th part needs i-th machine at some step of its production process. In mathematical terms, the problem of cell formation was first defined as one of finding independent permutations of rows and columns that lead to a block-diagonal structure of matrix $\mathbf{A}$. For real data the perfect block-diagonal structure rarely occurs and the goal is to obtain the structure that is as close to a block-diagonal one as possible.

The problem of optimal (usually, with respect to the amount of intercell movement) cell formation has been studied by many researchers. An overview can be found in (Selim et al., 1998; Yin & Yasuda, 2006; Balakrishnan & Cheng, 2007) and in (Bhatnagar & Saddikuti, 2010). However, no tractable algorithms that guarantee optimality of the obtained solutions were reported because of computational complexity of the problem. Moreover, even worst-case performance estimates are not available for most approaches. In fact, it was only shown that they produce high quality solutions for artificially generated instances. At the same time, today's highly competitive environment makes it extremely important to increase the efficiency of manufacturing systems as much as possible. In these conditions any noticeable improvement (e.g., achieved by properly designed manufacturing cells) can provide a secure position for a company in a highly competitive market.

This chapter is organised as follows. The next section provides an overview of the cellular and alternative layouts. Section 1.3 introduces a notion of dis/similarity measure and provides an analysis of similarity and performance measures used in CF. Section 1.4 provides an overview of the existing approaches and their classification while Section 1.5 summarizes the current state-of-the-art in cell formation and presents the outline of the thesis.

1.2 Cellular layout and its counterparts

Today's highly competitive market puts a constantly increasing pressure on the manufacturing industries. Current challenges, such as increasing fraction of high variety low volume orders, short delivery times, increased complexity and precision requirements, etc., force the companies to extensively optimize their manufacturing processes by all possible means. It is not hard to understand that layout of the processing units (machines, departments, facilities) can drastically influence the productivity of the whole manufacturing system both explicitly and implicitly. The explicit impact of the layout is expressed, for example, via the material handling costs and time (spent on delivering parts from one unit to another), tooling requirements, etc. The implicit impact of the layout can be explained by the fact that smaller and well-structured systems are usually easier to manage. This provides a possibility of finding more optimal management solutions (e.g., most scheduling problems are computationally hard, and the problem structure and size substantially influence the possibility of obtaining optimal solutions), as well as additional space for improvement (e.g., possibilities for set-up time savings).

The two classical types of layout that were prevailing not so long ago (and are still used) are *job shop* (functional) and *flow shop* production line layouts. In a job shop layout, machines are grouped into functional departments based on a similarity of their functions: drilling, milling, thermal processing, cutting, storage, etc. This process-oriented layout has certain advantages, first of all, from the perspective of flexibility (with regard to a changing product mix), expertise and cross-training. Indeed, it imposes no dedication of machines to parts, so that a wide variety of parts can be manufactured in small lot sizes. In addition, as all machines in a department perform similar functions, any person able to operate one of them is able to operate other ones (sometimes after a limited additional training). Moreover, as each functional department brings together specialists in the same field, it becomes easier for them to communicate and learn from each other. However, it was shown that in job shop systems parts spend up to 95% of their manufacturing time on waiting in the machine queues (Askin & Standridge, 1993) and travelling from one machine to another. The remaining 5% of the total time is shared between setup and value adding processing time. These figures imply that functional layout is very inefficient, but it also has another drawback. When a new part is released into the shop floor, a need for rescheduling all the system may occur[2], especially if the part has a very tight deadline and cannot be processed on a FIFO basis. This substantially complicates the management. The flow shop layout, as compared to the job shop, is product-oriented and is optimized for manufacturing a small variety of parts in large volumes. This is done by grouping machines into several manufacturing lines such that there is a straight “linear” flow across each line. However, the mix flexibility in this case is assumed to be very low and adding new products may destroy the “linear” structure of the flow shop layout. Thus, in case of high-variety-low-volume orders the flow shop is very inefficient.

The *cellular* layout is intended to combine advantages of both the considered above layouts and to make the management easier by decomposing the whole manufacturing system into several almost independent subsystems. This layout can be viewed as an application of group technology and suggests that parts that need similar operations and resources should be grouped into product families such that each family is processed within an almost independent smaller size manufacturing subsystem – a cell. In case of cellular layout, machines are grouped in such a way that the physical distance between machines in a cell is small and each cell contains (almost) all the machines needed to process the corresponding part

[2] This applies only if parts are processed according to a general optimised schedule. In a common practice, however, heuristic rules are applied to choose which part will be processed next at each machine.

family. This separates the flows, similarly to the flow shop, but also preserves a certain degree of flexibility as part families are usually robust to the changes in the product mix (i.e. new parts usually fit well into present families). In other words, cells are supposed to inherit the advantages of a job shop producing a large variety of parts and a flow shop dedicated to mass production of one product (in case of cells – one family of products). It was shown in Kusiak (2000) that a reduction of 20 to 80% in material handling costs can be achieved by introducing machine cells.

The *fractal* layout (see, e.g., Tharumarajah et al., 1996) was proposed as an alternative to the other layouts in order to minimize the total part flows. It is based on an observation that the pattern of logical relations between parts usually possesses a hierarchical structure similar to the structure of a fractal. These relations between parts are of two basic types: (i) part *a* is a sub-part of part *b* (*a* needs only some operations that *b* needs) and (ii) parts *a* and *b* should be assembled together. In case (i) the set of machines needed for part *a* is a proper subset of machines needed for part *b* – machines needed for both parts *a* and *b* should be placed closer to each other, machines needed only for *b* should be placed around them. In case (ii) the sets of machines needed for *a* and *b* can be completely different – in this case the two corresponding groups of machines should be placed next to each other. There is also a somewhat different interpretation of the fractal layout (see, e.g., Montreuil et al., 1999). It suggests that a manufacturing system is decomposed into a number of cells such that each cell has machines of several types in ratios similar to those of the whole manufacturing system. This implies that each cell can produce almost any part, but some are more suited for a particular part than others. Due to its cellular structure, the fractal layout offers certain advantages similar to those of the cellular layout. Naturally, the fractal layout can be viewed as a cellular layout with some additional properties: similarity of cells and/or their hierarchical structure. On the other hand, there is a fundamental difference between the two: while the fractal layout is process-oriented, the cellular one is usually more product-oriented as each cell focuses on a production of few parts. The fractal layout is hardly possible in many manufacturing systems, especially those where most of the machines are unique. In addition, the problem of balancing the load of equivalent machines assigned to different cells may emerge.

The *random* and the *maximally distributed* (also known as *holonic*) layouts (see, e.g., Benjaafar & Sheikhzadeh, 2000) are aimed at minimizing the product flow at a condition of high mix flexibility. They suggest that machines are randomly placed on the factory floor, or machines of the same type are placed as evenly as possible within the plant, correspondingly. This ensures that for an arbitrary part the expected travelling distance between two consecutive machines is limited. Thus, these two layouts guarantee a worst case (w.r.t. product mix) moderately good performance. At the same time, these two layouts are almost unstructured that makes it quite challenging to manage such a system (or even to find a way in it for the personnel).

To sum up, in most situations, except the limiting cases (see Figure 1.1), the cellular layout is beneficial over the other ones from the perspective of part flows. As can be seen from the literature, in case of large lot sizes and low variety the flow layout is beneficial as manufacturing lines substantially reduce the handling costs and make management very easy. Only in case of very high variety and low volumes the cellular layout may not be possible and the best choice will be the functional layout. The cellular layout can be also thought of as a way of moving from the functional layout to flow lines: a decomposition into cells decreases the variety of parts processed in each cell. It should be mentioned that the condition of high variety does not itself prohibit efficiency of the cellular layout as parts within a family are assumed only to use similar sets of machines, regardless of their operational sequences. Thus, in practice the cellular layout and, therefore, the cellular manufacturing seem to be very promising as they make a rather general assumption about the structure of a manufacturing system, while the other approaches either ignore this structure (e.g., the random layout) or assume too much structure (e.g., the fractal layout) which is more likely to be absent.

The main advantages of the *cellular manufacturing* (CM) can be summarized as follows (see, e.g., Kusiak & Chow, 1988; Wemmerlov & Hyer, 1986; Vin, 2010):

- Reduction of material handling costs and time. In CM almost each part is processed in a single (small) cell. Thus, all flows are concentrated in the cells and the travelling distances are small.

- Reduction of throughput times. Reduced travelling times and transfer of each part to the next machine once it is processed reduce the total time spent in the manufacturing system.

- Reduction of setup time. A manufacturing cell produces similar parts. Thus, the settings for the parts can also be similar and the time needed to change setups is saved.

- Reduction of tooling requirements. See the previous point.

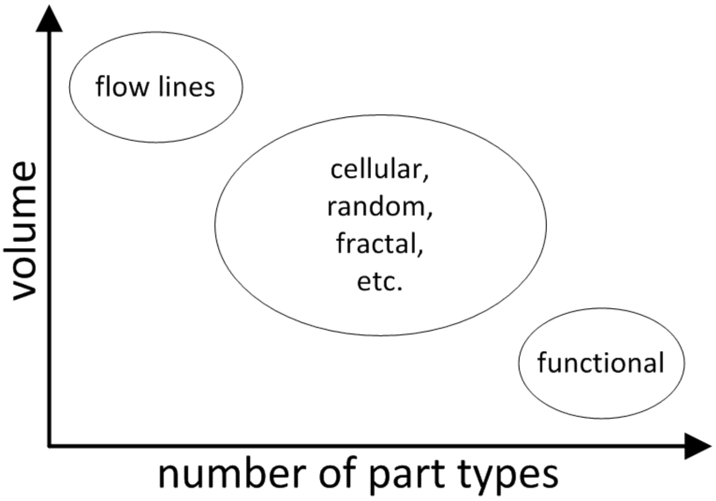


Figure 1.1. Relevance of layouts with regard to the product mix. (By the number of part types we mean the number of different processing sequences.)

- Reduction of work-in-process (WIP) and finished goods inventories. It was shown by Askin & Standridge (1993) that WIP could be reduced by 50% if the setup time is cut in half. This reduction also decreases the order delivery time.
- Reduction of space requirements. Reduced WIP and tooling requirements allow to save some space. This, in turn, can be used to shift machines closer to each other and further decrease material handling costs.
- Reduction in management efforts (scheduling, planning, etc.). Small and almost independent subsystems (cells) are substantially easier to manage than the whole large manufacturing system.
- Reduction of wasted parts percentage and improved product quality. Localized and specialized cells force the expertise to be concentrated. Small cells imply faster feedback if something goes wrong with a part.

However, CM has a number of negative side effects:

- Substantial implementation costs: identification of optimal manufacturing cells and part families, physical reorganisation (moving machines), additional cross-training of the personnel, etc.
- Difficulties in workload balancing and lack of robustness. Each machine can be important for the functioning of the whole cell, if it breaks the cell can

become inoperable. This can be partially tolerated by cross-training but the number of workers may become a constraint.

- Broad expertise. Each cell contains machines of different types and workers need a broader “specialization”.
- Synchronisation of parts for further assembly. Additional measures and resources (e.g., storage space) are needed to handle parts that are processed in different cells but must be assembled together.
- Lower utilization of the machines. Independence of the cells can be improved by introducing additional machines, but the load of them decreases. Moreover, if two or more cells contain equivalent machines, the load balancing problem may occur when one of such machines is underutilised while the other one is overloaded.
- Lack of flexibility. Changes of the product mix can completely destroy independence of the cells.

Despite these disadvantages, CM is assumed to improve the performance of the manufacturing system in case of high-variety-low-volume environment and, thus, is an important issue in industrial engineering. At the same time, it is clear that transition to the CM should be designed very carefully in order to reduce possible drawbacks. Finally, it is extremely important to understand that the benefits of cellular layout on its own are very restricted and it rather provides a possibility of improvement. That is why a positive effect can be achieved only if CM is complemented by proper management and planning.

1.3 A notion of (dis)similarity and performance measures

As mentioned in the introduction, CF is aimed at obtaining independent cells and this cellular decomposition becomes possible by exploiting similarities in the manufacturing processes of different parts. Thus, to construct an algorithm for solving the CF problem one usually needs to define a similarity measure for parts and or machines. The notion of similarity measure is very important for the problem under consideration. In particular, after introducing a (dis)similarity measure for machines, one can restrict himself to considering only machine-machine relations.

This substantially reduces the problem size, taking into account that the number of machines is usually quite limited, while the number of parts can be magnitudes larger. For example, Park & Suresh (2003) consider and instance with 64 machine types and 4415 parts, we experienced instances with 30. . .60 machine types and 5733. . .7563 parts in practice. Alongside with a possibility for problem size reduction, (dis)similarity measures provide flexibility to the model – they may incorporate a variety of manufacturing factors, as will be shown in the latter chapters of this thesis.

After the CF problem is solved it is necessary to estimate the effectiveness of the obtained cellular decomposition, i.e. a solution performance measure is needed. The following subsections provide an overview and analysis of the existing similarity and performance measures; a good analysis of similarities and related aspects can be found in Owsinski (2009).

1.3.1 Similarities and dissimilarities

It is not hard to understand that in case of independent cells manufacturing processes of any two parts not assigned to the same cell differ a lot, i.e. these parts do not use the same machine types. Note, that this does not automatically imply that any two parts within one cell are very similar (use mainly the same machines). To illustrate this, consider the following example.

Let the manufacturing system be represented by the following machine-part incidence matrix:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Suppose, one is interested in two cells. It is easy to see that the cells can be as follows:

cell 1: machines 1,2,3,4; parts 1,2,3,4;

cell 2: machines 5,6; parts 5,6,7.

It is easy to see that these cells are completely independent. However, parts 2 and 3 from cell 1 as well as parts 5 and 7 from cell 2 do not have similar manufacturing processes (they use completely different machines) even though they are in the same cell.

Thus, the goal of the CF problem is to maximize the dissimilarities between cells and its objective is of the general form

$$\max F(d^p(i,j) \cdot x^p_{ij} \mid i,j = 1 \ldots r) \tag{1.1}$$

where $F(.) : \mathbb{R}^{r \times r} \to \mathbb{R}$ is some functional, $d^p(i,j)$ is the dissimilarity between manufacturing processes of parts i and j, x^p_{ij} are Boolean decision variables that are equal to 1 if and only if parts i and j are in different cells. In fact, $F(.)$ can be linear in x-variables, i.e. of the form $F(d^p(i,j) \cdot x^p_{ij} \mid i,j = 1 \ldots r) = \sum_{i,j} d^p(i,j) x^p_{ij}$, due to the following lemma.

Lemma 1.1. *If one aims at the most independent cells, then the objective function of the CF problem is essentially linear.*

Proof. Let us consider a specific set of cells. Observe that the impact of each part is independent of the impacts of the other parts. This is because of the fact that if some part has to move from one cell to another this adds exactly one intercell movement, irrespective of the amount of intercell movement induced by other parts. This means that the total impact of all parts is just a sum of impacts of each part. ■

This lemma will be illustrated in the following chapters of the thesis: all the proposed models have linear objective functions, irrespective of the particular objective for the cell formation, and despite the fact that some approaches in literature use nonlinear objective functions.

Thus, the problem of cell formation can be posed as one of maximizing the sum of dissimilarities between parts. Once parts are grouped into the product families, machines can be efficiently grouped into the cells just by assigning each machine independently to the cell where it is most needed. However, this way of dealing with the problem replaces an $m \times r$ machine-part incidence matrix by an $r \times r$ part-part dissimilarity matrix, thus increasing the problem size. Yet, there exists a completely symmetric way of dealing with the cell formation problem: instead of differences between parts one can consider differences between machines. If one denotes by $d^m(i,j)$ difference between machines i and j based on the difference between sets of parts that need these machines then the objective becomes

$$\max \sum_{i,j} d^m(i,j) x^m_{ij} \tag{1.2}$$

where x_{ij}^{m} – Boolean variables equal to 1 if and only if machines i and j are in different cells.

From the CF perspective dissimilarities $d^{m}(i,j)$ must depend on the machine-part incidence matrix and, as will be shown later, on the sequence in which a part visits machines. In the simplest case when operational sequences are ignored each machine is completely characterized by a Boolean vector (a row in the machine-part incidence matrix). Thus, the dissimilarities $d^{m}(i,j)$ can be defined as some distance between the corresponding Boolean vectors (rows i and j): from Euclidean or Hamming distance to any sophisticated measure.

It should be mentioned that the problem of the form $\max \sum_{i,j} d(i,j)x_{ij}$ can be equivalently transformed:

$$\begin{aligned}
&\max \textstyle\sum_{i,j} d(i,j)x_{ij} = \\
&\textstyle\sum_{i,j} d(i,j) - \min \sum_{i,j} d(i,j)(1-x_{ij}) = \\
&\textstyle\sum_{i,j} d(i,j) + \max \sum_{i,j}(-d(i,j))(1-x_{i,j}) = \\
&c + \max \textstyle\sum_{i,j} s(i,j)(1-x_{i,j}) \simeq \\
&\min \textstyle\sum_{i,j} s(i,j)x_{i,j}
\end{aligned}$$

where the coefficients $s(i,j) = -d(i,j)$ are called *similarities* and c – some constant. Thus the problem of cell formation can be formulated as a maximization of the sum of similarities within each cell (most of the similarity-based approaches in literature use this form) or as a minimisation of similarities between cells.

Clearly, a definition of the similarity measure is ambiguous, like that of the dissimilarity measure. Even though several similarity measures were proposed in literature (an overview can be found in Shafer & Rogers (1993); Yin & Yasuda (2006); Owsinski (2009)), to the best of our knowledge for none of them there exists a strict proof of adequateness. Rather, it was shown empirically that they work well in some cases. Thus, the issue of formulating a strictly reasoned (dis)similarity measure remains open. In the following chapters several similarity measures reflecting different objectives will be proposed and explained.

1.3.2 Performance measures

As cell formation is aimed at making independent manufacturing cells, an amount of intercell movement, i.e. an amount of parts that must be processed in more than one cell is a natural performance measure of the cellular decomposition[3]. We used the term "amount of parts" to underline that one can be interested not just in minimizing the number of parts travelling between cells but also their mass or volume, etc. In case of functional grouping with a binary input matrix this amount is exactly the number of ones outside the diagonal blocks and is denoted by n_e – the *number of exceptional elements*. Thus, in the simplest case n_e can be used as a performance measure of the cellular decomposition.

Another characteristic often used to estimate the performance is the *number of voids* n_v. In terms of block diagonal matrices it is just the number of zeroes within diagonal blocks. Let us use the term *operation* to denote a single processing step of one part, i.e. processing of some part by some machine. Now, in terms of operations n_v means the number of operations that can be performed without increasing intercell movement but are not realized (are not needed).

Clearly, minimum values of n_e and n_v depend on the number of cells p and the following lemma shows an important property of these values.

Lemma 1.2. *The following two properties take place:*

(i) function $\min n_e(p)$ *is nondecreasing in* p*;*

(ii) function $\min n_v(p)$ *is nonincreasing in* p*;*

where minima are taken over all decompositions into p *nonempty cells (i.e. each cell performs at least one operation).*

Proof. We will start from part (i). Fix the input data, denote $n_e^*(p) = \min n_e(p)$ and consider two optimal (with respect to n_e) decompositions into p and $p+1$ cells, respectively. The numbers of exceptional elements of these decompositions are $n_e^*(p)$ and $n_e^*(p+1)$. Now, consider a decomposition with $p+1$ cells and merge any two cells. This leads to p cells and the number of exceptions $n_e'(p)$ such that $n_e'(p) \leq n^*(p+1)$. On the other hand, $n_e^*(p) \leq n_e'(p)$ holds, just by minimality of the latter. Thus, we have $n_e^*(p) \leq n_e'(p) \leq n_e^*(p+1)$ for arbitrary number of cells $p = 1, \ldots, m$.

A similar reasoning can be used to prove part (ii). ■

Along with n_e and n_v the proposed in literature performance measures also use the total number of operations n_1 (the total number of ones in the machine-part incidence matrix), purely for normalisation purpose. In fact, if the desired number of

[3] More precisely, the amount of parts travelling between cells. A single part may be processed in only two cells but if it has to travel several times between the cells, then the intercell movement is larger. This issue is often ignored, especially if the input data is represented by a MPIM.

cells is fixed n_e is the best performance measure as this values completely reflects the goal of cell formation – decomposition into independent cells. However, if the number of cells is also a variable then any algorithm minimizing only n_e in practical cases will produce a single cell, as usually perfect cells are not possible and the smallest amount of intercell movement equal to 0 is achieved by a single sell that contains the whole manufacturing system. In the capacitated versions of the cell formation problem constraints on the cell size, workload, etc. ensure reasonable cells. However, in the uncapacitated approaches to avoid this effect n_v was artificially introduced into the objective. Taking into account that n_v has an opposite behavoiur to n_e (see Lemma 1.2), this will force the number of cells to attain some reasonable value. As n_v is not connected to the original goal of cell formation, a number of ways of introducing it into the performance measure were proposed (an overview can be found in Sarker, 2001; Keeling et al., 2007). Yet, similarly to the situation with the (dis)similarity measures there is no strict theoretical explanation why one is better than the other.

We would like to finish this section with some examples of the performance measures most widely used in literature:

- n_e (Albadawi et al., 2005),
- $n_e + n_v$ (Bhatnagar & Saddikuti, 2010),
- $GCI = 1 - \frac{n_e}{n_1}$ – group capability index (Hsu, 1990),
- $\tau = \frac{n_1 - n_e}{n_1 + n_v}$ – grouping efficacy (Kumar & Chandrasekharan, 1990; Albadawi et al., 2005; Ahi et al., 2009),
- $\eta = \alpha \frac{n_1 - n_e}{n_1 - n_e + n_v} + (1 - \alpha) \frac{mr - n_1 - n_v}{mr - n_1 - n_v - n_e}$ – grouping efficiency (Chandrasekharan & Rajagopalan, 1986a; Ashayeri et al., 2005; Ahi et al., 2009),

where $\alpha \in [0, 1]$ – weighting factor, usually set to 0.5. Another example of a performance measure is the amount of intercell movement. It will be shown in the following chapters that, generally speaking, it is not the same as the number of exceptions n_e. Some of these measures will be used in order to compare the performance of several approaches.

An overview of performance measures and their empirical evaluation can be found in Sarker (2001); Keeling et al. (2007).

1.4 An overview of the existing approaches

As was already mentioned in the previous section there exist a great number of approaches to solving the cell formation problem. From the most general perspective these approaches can be classified as follows:

- clustering based on energy functions,
- similarity based hierarchical clustering,
- fuzzy logic methods,
- genetic algorithms and simulated annealing,
- neural networks,
- graph-theoretic approaches,
- mixed-integer linear programming (MILP).

It should be mentioned that all the groups of approaches except the last two are intrinsically heuristic, see (Miltenburg & Zhang, 1991) for a comparative study. On the contrary, the CF problem can be modelled exactly in terms of graph partitioning or MILP, but these lead to computationally intractable (NP-hard) problems, thus forcing the use of heuristic solution methods.

Below we give a brief overview of all the mentioned classes of approaches to cell formation in order to provide a reader an impression about all kinds of algorithmic tools applied to CF. The earliest iterative approaches representing ad hoc algorithms are described in detail, while for those based on standard techniques (e.g., neural networks or genetic algorithms) only the main peculiarities are mentioned. Such a level of detalisation, as we hope, will be useful especially for those not familiar with cell formation and the approached involved.

1.4.1 Bond energy analysis

The idea of using the bond energy of the cells as a criterion of clustering performance was first used by McCormick et al. (1972) in their bond energy algorithm (BEA). BEA is aimed at identifying clusters that are present in complex data arrays by permuting rows and columns of the input data matrix in such a way as to push the numerically larger elements together. The measure of clustering effectiveness (ME) used in bond energy algorithm was devised so that an array that possesses dense blocks of numerically large elements will have a large ME when compared to the same array whose rows and columns have been permuted so that its numerically large elements are more uniformly distributed. This measure is the sum of the bond strengths, where bond strength is defined as a product of a pair of nearest-neighbour elements:

$$\mathrm{ME}(A) = \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} a_{ij}(a_{i,j+1} + a_{i,j-1} + a_{i+1,j} + a_{i-1,j}) \tag{1.3}$$

where A - any $M \times N$ array with nonnegative elements. The defined in such a way ME has the following theoretical and computational advantages (McCormick et al., 1972):

- The ME is applicable to arrays of any size and shape; the only requirement is nonnegativity of elements.
- Since the vertical (horizontal) bonds are unaffected by the interchanging of columns (rows), the ME decomposes into two parts: one dependent only on row permutations and the other dependent only on column permutations. Thus, ME can be optimized in two phases by finding the optimal colums permutation and then the optimal row permutation (or vice versa).
- Since the contribution to the ME from any column (or row) is only affected by the two adjacent columns (rows), i.e. only local information is used, the ME optimization leads to a sequential suboptimal procedure.

The proposed by McCormick et al. (1972) algorithm is as follows:

1. Place one of the columns arbitrarily. Set $i = 1$.
2. Try placing individually each of the remaining $N - i$ columns in each of the $i + 1$ possible positions and compute their contributions to ME. Place the column into the position that gives the highest contribution. Increment i by 1 and repeat until $i = N$.
3. After arranging all the columns do the same procedure for rows. (This part is unnecessary in case of symmetric input matrix.)

The main characteristics of the algorithm are as follows:

- Computational time depends only on the size of input matrix and has order of $O(M^2N + N^2M)$.

- The algorithm always leads to a block diagonal form of the matrix if it can be obtained by row and column permutations.
- The final ordering is independent of the order in which rows (columns) are given and depends only on the initial row (column).

King (1980) proposed a more sophisticated algorithm that solves the particular case of binary input matrix and exploits the binary nature of input. The algorithm is as follows:

1. Consider each row of the machine-parts matrix as a binary number. Rank the rows in order of decreasing binary value. Rows with the same value should arbitrarily be ranked in the same order in which they appear in their current matrix (from top to bottom).
2. Check if the current matrix row order (numbering from top to bottom) and the rank order just calculated coincide. If yes, go to 6. If no, go to 3.
3. Rearrange the machine-part matrix starting with the first row by placing the rows in decreasing rank order. Rank columns in decreasing binary value. Columns with the same value should be arbitrarily ranked in the order in which they appear in the current matrix (reading from left to right).
4. Check if the current matrix column order and the rank order just calculated are the same. If yes, go to 6. If no, go to 5.
5. Rearrange the machine-part matrix starting with the first column by placing columns in decreasing rank order. Go to 1.
6. Stop.

King claimed that his ROC algorithm always finds a block diagonal structure if it exists and requires much less computer time than McCormick's et al. technique. However, it was shown in (Chandrasekharan & Rajagopalan, 1986b) that ROC can fail if the matrix has almost block diagonal form (2 exceptional elements in a 20×35 matrix). Another peculiarity of ROC is that it clusters machines and parts simultaneously while most other approaches first cluster machines and then derive part families.

The modification of ROC proposed by Chandrasekharan & Rajagopalan (1986b) MODROC has better performance in case of ill-structured data and consists of three stages:

Stage 1. ROC is applied on the rows and columns of the initial matrix repeatedly in two iterations. This results in an ordered matrix that has the following properties. If the first k elements are ones in the ith row then at least the first k elements are ones in the $(i-1)$th row. This is also true for columns.

Stage 2. Identification of perfect blocks (an all ones submatrix).

Stage 3. Hierarchical clustering of blocks.

The last approach based on bond energy that we consider here is used in the direct clustering algorithm (DCA) by H. M. Chan & Milner (1982). Like the previous ones, DCA iteratively permutes rows and columns such that the nonzero entries of the input matrix are grouped into dense clusters and can be outlined as follows:

1. Count the number K of nonzero cells in each column and in each row. Rearrange the machine-part matrix with columns in decreasing and rows in increasing order of K.

2. Starting with the first column in the matrix, transfer the rows which have nonzero entries in this column to the top of the matrix. Repeat the procedure with the other columns, until all the rows are rearranged.

3. Check if the matrix has changed from the previous step. If yes, go to 4. If no, go to 6.

4. Starting with the first row of the matrix, transfer columns that have nonzero entries in this row to the left-most position in the matrix. Repeat the procedure for all the other rows, until all the columns are rearranged.

5. Check if the matrix has changed from the previous step. If yes, go to 2. If no, go to 6.

6. Stop.

This algorithm can work with any starting form of the machine-part matrix. The iterative procedure of DCA converges after a limited number of iterations (H. M. Chan & Milner, 1982) and unlike all the mentioned above approaches the result is always the same, irrespectively of the initial permutation of rows and columns.

The above mentioned algorithms are reasonably fast but involve intuitive procedures that cannot guarantee optimality.

1.4.2 Iterative approaches based on similarity measures

As follows from the title, all these approaches need some similarity measure $S(.,.)$ to be defined for any pair of machines (and parts). Several similarity measures have been considered and a particular choice was usually made either based on experimental evaluation of possible candidates or on the desired properties of the manufacturing cells to be obtained.

One of the first papers considering an iterative hierarchical clustering approach based on similarity measures is by McAuley (1972). He used a single linkage clustering algorithm (SLC) in which the similarity measure between two clusters is defined as the maximum of the machine similarities between machine pairs where machines of the pair are in different clusters. In a formalized form, the measure of similarity $S(K_1, K_2)$ between two clusters K_1 and K_2 is defined as:

$$S(K_1, K_2) = \max_{i_1 \in K_1, i_2 \in K_2} S(i_1, i_2) \tag{1.4}$$

The idea of the hierarchical clustering algorithm is very simple. At the beginning each machine is considered as one separate cluster, then iteratively two clusters with the highest similarity are merged into one bigger cluster. Usually, a threshold value is introduced and merges occur only if similarity between a particular pair of clusters exceeds this threshold. The result of such clustering algorithms can be represented in a form of dendogram (tree), nodes of which represent machine cells at different levels of detail. One of the main disadvantages of such procedure is the so-called chaining effect: clusters that have low similarity for most of the machine pairs can be merged if there exist a single pair of machines that are similar enough. By its essence hierarchical clustering is equivalent to the approach based on the minimum spanning tree problem (to be discussed in Section 1.4.6).

Numerous modifications were proposed to avoid chaining effect. These include complete linkage clustering (CLC), average linkage clustering (ALC) and linear clustering algorithm (LCC). The main difference between all of them is the definition of similarity measure for clusters. In case of CLC the similarity $S(K_1, K_2)$ between two clusters K_1 and K_2 is defined as

$$S(K_1, K_2) = \min_{i_1 \in K_1, i_2 \in K_2} S(i_1, i_2) \tag{1.5}$$

while for ALC it is given as

$$S(K_1, K_2) = \frac{1}{|K_1||K_2|} \sum_{i_1 \in K_1, i_2 \in K_2} S(i_1, i_2) \qquad (1.6)$$

where $|K_1|$ and $|K_2|$ are cardinalities of the corresponding clusters. The LCC algorithm is slightly more sophisticated and can be described as follows (Wei & Kern, 1989):

Step 1. Select the highest similarity value that has not yet been considered in the clustering process. Assume, it is the similarity for machine pair (i_1, i_2). One of four cases occurs:

(a) Neither machine i_1 nor i_2 has yet been assigned to a machine cell. In this case a new cell is created containing these two machines.

(b) One of the machines is already assigned to some cell. In this case the second machine is added to the same cell.

(c) Machines i_1 and i_2 are already assigned to the same cell. Nothing needs to be done.

(d) Machines i_1 and i_2 are already assigned to different cells. The similarity between them implies that the two cells can be merged in later processing. This pair is marked.

Step 2. Repeat Step 1. Go to Step 3 when all machines have been assigned to cells.

Step 3. Steps 1 and 2 create the maximum number of clusters that would fit the situation defined by the input matrix. If there are no bottleneck parts created by this clustering then the solution is optimal. However, this solution may contain more cells than it is desired. If so, go to Step 4.

Step 4. Starting with the largest commonality score that was marked at Step 1d start joining the cells. If at some step the resulting cell is too large or does not conform with some other requirements then do not perform the join operation.

Step 5. Repeat Step 4 until all predefined constraints on number of cells, their size, etc. are satisfied.

The authors claim (Wei & Kern, 1989) that the algorithm has linear complexity.

Likewise the previous group of algorithms, iterative approaches are reasonably fast but involve intuitive procedures that cannot guarantee optimality.

1.4.3 Fuzzy logic approaches

The main assumption behind all the above mentioned approaches and those based on graph partitioning and mathematical programming is that the part families are mutually exclusive and collectively exhaustive, i.e. each part can only belong to one part family. The absence of ideal block diagonal structure of the machine-part incidence matrix for most real manufacturing systems, as well as uncertainties (e.g., about future part demands) and ambiguities (e.g. some part has half of operations within one cell and other half in another) had lead to an idea of using fuzzy logic (instead of classical one) in cell formation. From a qualitative point of view such changes mean that classical Boolean decisions (e.g. some machine is either included into a particular cell or not) are replaced by fuzzy ones (a machine is likely to be included into a particular cell with some likelihood coefficient $\mu \in [0;1]$). The most important fact is that fuzzy arithmetic can be plugged into any existing algorithm for cell formation by replacing Boolean variables by continuous ones defined on the interval $[0;1]$ and classical logic operations by fuzzy ones (e.g., conjunction can be replaced by min, disjunction by max and negation $\neg x$ by $1 - x$). At the output, for any machine (part) a vector of inclusion coefficients for any cell (part family) is obtained and the machine (part) is assigned to the cell (part family) that corresponds to the highest coefficient. We do not give a detailed description of successful implementations of fuzzy logic approach for the sake of shortness as fuzzy logic operations lead to quite extensive notations. Instead, we would like to refer the reader to Xu & Wang (1989); Chu & Hayya (1991); Gindy et al. (1995); Narayanaswamy et al. (1996) where relevant algorithms are explained in detail and examples are given. Of certain interest is a paper by Suresh et al. (1999) where fuzzy logic approach is used within a framework of neural networks.

1.4.4 Genetic algorithms and simulated annealing

Genetic algorithms (GA) and simulated annealing (SA) are quite general metaheuristics that proved to be useful in a wide variety of optimization problems (including clustering and classification). Thus, their application to a field of cell formation is quite natural.

Authors usually start with a non-linear objective function and some initial configuration of machine cells, and then apply an evolutionary procedure to optimize the value of the objective (Adil & Rajamani, 2000; Xambre & Vilarinho, 2003). For example, Adil & Rajamani (2000) use an objective function that contains two non-linear terms reflecting intra- and intercell movement costs. The SA that they use has the following main steps. Initially, the number of cells is set equal to the number of machines and each machine is assigned to a separate cell. This is an initial configuration. At each subsequent iteration, one machine is moved from the current cell to another in order to get a new machine assignment. The machine to be moved and the cell for it are chosen randomly and after the movement the objective value is updated for the new configuration. The generated solution is accepted if the objective value is improved. If the objective value is not improved then the solution is accepted with some probability depending on a temperature that is high at the beginning and decreases during the execution of the algorithm. Such setting ensures that a large proportion of generated solutions are accepted at the beginning and local optima can be avoided at early stages. Decreasing temperature allows the algorithm to stabilize in a vicinity of some local (and, hopefully, global) optimum. At each cooling temperature many moves are tried and the algorithm stops when predefined conditions are met.

Genetic algorithms are applied to cell formation in the same spirit and differ from SA only in the details of the evolutionary optimization procedure. A typical genetic algorithm starts with an initial population (pool) containing a predefined number of feasible solutions to the cell formation problem (decompositions into cells). Then, at each iteration some fixed number of the worst solutions are deleted from the population and the same number of new solutions is added. These new solutions are obtained in one of two ways: by small modifications of some solution already present in the current population (mutation) or by joining parts of two solutions (crossover). This procedure is repeated iteratively until some stopping criterion is met. The size of an initial population, the proportion of deleted solutions, probabilities of mutation and crossover are parameters of the GA. It should be mentioned that there are no provably good approaches for finding optimal values of these parameters (the same is true for the parameters of SA) and in practice they are found on a trial-and-error basis. Applications of GA to the cell formation can be found in (Mak et al., 2000; Filho & Tiberti, 2006).

1.4.5 Neural network approaches

Flexibility and universality of artificial neural networks (ANN), as well as presence of a rich choice of architectures and learning rules, inspired their application to the cell formation problem. All the ANN-based approaches can be classified into two groups: those using supervised and unsupervised learning. The first group typically uses either feed-forward perceptrons and BackPropagation learning rule (see, e.g., Kao & Moon, 1991) or Hopfield-like feed-back network (see, e.g., Liang & Zolfaghari, 1999). This group needs some learning set to be defined, i.e. a typical representative of each machine cell and part family should be chosen. Respectively, the desired number of cells should be known. Typical representatives are usually found by some heuristic procedure.

The neural networks from the second group are capable of finding the cluster structure of the input data without any additional knowledge about typical representatives and some architectures do not even need the number of clusters to be given. This group of ANN-based approaches includes the Carpenter-Grossberg neural network (Kaparthi & Suresh, 1992), so-called self-organising maps (Guerrero et al., 2002), competitive neural networks (Malave & Ramachandran, 1991; Venugopal & Narendran, 1994) and adaptive resonance theory (ART) networks (Suresh et al., 1999; Yang & Yang, 2008).

ANN-based approaches also differ in the type of input data they use: some deal directly with binary machine-part relations, while others perform clustering based on similarities.

1.4.6 Graph-theoretic approaches

One of the examples of applying graph theoretic approach to the cell formation can be found in (Rajagopalan & Batra, 1975). For a given manufacturing system authors construct a graph with each vertex representing a machine. An arc between two machines i and j represents the "strength" of the relationship between the machines. These "strengths" can be defined as similarity coefficients used in approaches from Section 1.4.2. Given this weighted graph, cliques can be found (a clique is a maximal complete subgraph) and these cliques are merged into production cells such that the relationship within a cell is "strong" (a sum of all pairwise similarities within a cell is large) and intercell relationships are "weak". Once production cells are formed, a set of rules are used to assign parts to cells. This approach works especially well if the number of machines is small, because the number of possible cliques increases exponentially as the number of vertices in a graph (a number of machines) increases.

In (Gower & Ross, 1969) an algorithm based on the minimum spanning tree (MST) problem is considered. As in the previous case, cell formation instance can be encoded in a weighted graph where each vertex corresponds to some machine and weights are dissimilarities between corresponding machines. Suppose, an MST is found in such a graph. Deleting $K-1$ heaviest edges from the MST produces K subtrees that can be interpreted as manufacturing cells. Such a procedure ensures that dissimilarity between machines within a cell is minimal. In (Ng, 1993) a worst case analysis of the MST approach is performed. Ng (1991) uses the bond energy formulation of the problem and then shows that it can be transformed into the rectilinear Travelling Salesman Problem (TSP) and also provides a worst case bound.

Finally, one of the most widely used graph-theoretic approaches to cell formation is based on the p-Median problem (PMP). This approach is closely related to the two approaches mentioned above. While the first approach suggests decomposition of the graph into cliques and the second one – into spanning trees, the approach based on the PMP seeks for the optimal decomposition of the graph into trees of depth one, i.e. trees consisting of a root and some leaves without internal nodes. A detailed description of the approach can be found in (Wang & Roze, 1997; Deutsch et al., 1998; Ashayeri et al., 2005; Won & Lee, 2004; Won & Currie, 2006) and in the next chapter.

The use of similarity measures is typical for this group of approaches. For a more detailed overview of graph-theoretic approaches to cell formation we refer the reader to (Chandra et al., 1993).

1.4.7 MILP based approaches

Mixed-integer linear programming (MILP) is quite a broad and well studied area. Many optimisation problems, including those of cell formation, have been translated into the MILP format due to a simple and quite general structure of the latter. In its most general form a MILP problem can be expressed as:

$$\min\left\{c^{\mathrm{T}}x \mid Ax \le b;\ x \in \mathbb{R}^n_+;\ x_i \in \mathbb{Z},\ i \in U\right\},$$

where x is a vector of variables, A is a real-valued matrix, b and c are real-valued vectors; the dimensions of A, b and c must be such that all the multiplications make sense. U is an index set for integer variables.

A number of researchers start from an explicit mixed-integer programming or mixed-integer linear programming formulation of the cell formation problem (and linearise the formulation if necessary). For example, Chen & Heragu (1999) define the problem as follows:

$$\sum_i\sum_j\sum_k c_j a_{ij} x_{jk}(1-y_{ik}) + \sum_i\sum_j\sum_k d_{ij}(1-a_{ij}) x_{jk} y_{ik} \to \min \tag{1.7}$$

$$\begin{aligned}
s.t. \quad & \sum_k x_{jk} = 1 \quad \forall j && (1.8)\\
& \sum_k y_{ik} = 1 \quad \forall i && (1.9)\\
& y_{sk} + y_{tk} \leq 1 \quad \forall k,\ (s,t) \in S_1 && (1.10)\\
& y_{sk} - y_{tk} = 0 \quad \forall k,\ (s,t) \in S_2 && (1.11)\\
& M_{min} \leq \sum_i y_{ik} \leq M_{max} \quad \forall k && (1.12)\\
& x_{jk} \in [0;1] \quad \forall j,k && (1.13)\\
& y_{ik} \in \{0,1\} \quad \forall i,k && (1.14)
\end{aligned}$$

where $a_{ij} = 0$ if machine i is not required for part j and $0 < a_{ij} \leq 1$ otherwise, c_j – intercell movement costs for part j, d_{ij} – cost of part j not utilizing machine i, M_{min} – minimum number of machines in a cell, M_{max} – maximum number of machines in a cell, S_1 - set of machine pairs that cannot be located in the same cell, S_2 – set of machine pairs that must be located in the same cell. Decision variables x_{jk} and y_{ik} have the following meaning:

$$\begin{array}{ll} x_{jk} = 0 & \text{part } j \text{ is not processed in cell } k \\ 0 < x_{jk} \leq 1 & \text{part } j \text{ is processed in cell } k \end{array} \qquad (1.15)$$

$$y_{ik} = \begin{cases} 1, & \text{machine } i \text{ is in cell } k \\ 0, & \text{otherwise} \end{cases} \qquad (1.16)$$

The first term in the objective function (1.7) represents the total costs of intercell movement and the second term represents the total costs of resource underutilization. Constraint (1.8) ensures allocation of each part to a cell. Constraints (1.9) and (1.14) ensure that each machine can only be assigned to one cell. Constraint (1.10) states that the machine pairs included in S_1 cannot be placed in the same cell. Similarly, constraint (1.11) forces machine pairs from S_2 to be placed in the same cell. Finally, constraint (1.12) specifies the minimum and maximum number of machines allowed in any cell. As the mentioned model has a nonlinear objective function (1.7), it was linearized by introducing new variables $z_{ijk} = x_{jk}y_{ik}$ and Chen and Heragu's MILP model is:

$$\begin{aligned}
& \sum_i \sum_j \sum_k c_j a_{ij} x_{jk} + \sum_i \sum_j \sum_k (d_{ij}(1 - a_{ij}) - c_j a_{ij}) z_{ijk} && (1.17)\\
& s.t. && (1.18)
\end{aligned}$$

$$(1.8) - (1.14) \tag{1.19}$$

$$z_{ijk} \leq x_{jk} \quad \forall i, j, k \tag{1.20}$$

$$z_{ijk} \leq y_{ik} \quad \forall i, j, k \tag{1.21}$$

$$z_{ijk} \in [0; 1] \quad \forall i, j, k \tag{1.22}$$

It should be mentioned that the number of integer (Boolean) variables in the model depends on the number of machines, parts and cells to be made. This means that for realistic instances having hundreds of parts the formulation becomes huge and hardly solvable.

Another MILP formulation that includes a wider range of practically motivated constraints can be found in (Slomp et al., 2005), however due to its size it is hardly tractable for moderate and large-size instances. Note also that all the models based on graph theory can be (and usually are) reformulated in terms of MILP. This is done in order to avoid the need of developing special algorithms for handling the model.

As MILP is computationally intractable (NP-hard) in general, heuristic methods were used to solve the obtained problems. However, in the next chapter we will show that there exist a compact MILP formulation based on the p-Median problem that can be solved exactly by a general-purpose MILP solver just due to its compactness (small size in terms of the number of variables and constraints).

We would like to conclude this section by saying that most classes of approaches (except MILP) rely algorithms for which addition of constraints is problematic. For example, for genetic algorithms it is quite easy to check if the generated solution satisfies additional constraints but generating a feasible solution may be challenging (the algorithm makes sense only if some feasible solutions are present in the pool). Thus, alongside with tractability and optimality, the issue of flexibility makes practical applicability of many approaches questionable.

1.5 Conclusions and outline of the thesis

Cellular decomposition of the manufacturing system has a substantial impact on its efficiency. This is caused by both explicit and implicit factors. First of all, cellular layout explicitly improves the products flow, reduces handling and cross training costs and delivery times. At the same time, the implicit impact of cellular layout is due to the fact that smaller systems are easier to manage. For example, taking into account NP-hardness of most scheduling problems, switching from one big manufacturing system to few small subsystems can make the difference between impossibility and possibility of making an optimal schedule.

Due to vast benefits proposed by cellular manufacturing, the cell formation problem has been extensively studied for more than 50 years. This resulted in a wide variety of approaches as well as modifications of the problem. However, to the best of our knowledge there have been very few attempts of solving the problem to optimality and almost all the proposed models for CF problem are either of intuitive (heuristic) nature or are solved by heuristic procedures. This means that the obtained solutions incorporate two types of errors: an intrinsic error of modelling and a computational error induced by a heuristic solution procedure. In fact, for an overwhelming majority of the existing approaches no worst case performance guarantees are available and it was only shown that they give satisfactory results for some artificial instances (as optimal solutions to the real life instances are usually not known). Moreover, not only most solution algorithms are lacking a strict theoretical analysis but also such basic concepts as dis/similarity and performance measures. One may conclude that despite its long history the theoretical and applied sides of the CF problem have certain gaps that we are going to fill in the following chapters.

The main research theme can be formulated as follows: **design of an applicable in practice approach (model) for solving the cell formation problem**. By practical applicability we mean that the approach (model) must satisfy certain criteria:

- guaranteed solution quality;
- reasonable running times for real-life instances;
- flexibility: possibility of adding additional constraints and/or objectives.

Taking these requirements into account, the methodological grounds of this thesis are as follows. Based on the observation that there is a prominent imbalance between the number of machines and parts (dozens vs. thousands) we conclude that an efficient model uses a (dis)similarity measure and works with machine-machine relations first making machine cells and then assigning parts to the cells made. In order to comply with the flexibility requirement, we will consider the models expressed in terms of mixed-integer linear programs. Having quite a general and simple form, MILP models can be extended by any number of linear constraints without affecting the general structure of the problem. This choice of a model format can be further motivated by the fact that MILP is a well studied area and there exist a number of commercial (e.g., Cplex, Xpress-MP) and non-commercial (e.g., GLPK) solvers. Contemporary solvers are very efficient and able to handle instances with thousands of variables and constraints. Furthermore, the use of available solvers makes the implementation of the models much easier by avoiding the need of developing special algorithms and programming them.

In the following chapters we propose two new models based on the p-Median and multicut problems. The first model is an efficient heuristic having a restricted modelling error and a zero computational error. The second model solves the problem exactly; however, due to its computational complexity only instances of a moderate size (in terms of the number of machines) can be handled. Yet, we demonstrate the applicability of this model by an industrial case. Besides the two models, we propose several similarity measures exactly reflecting possible objectives of cell formation.

The rest of the thesis is organised as follows. Chapter 2 provides an insight into the p-Median problem (PMP) and its properties. An efficient model based on the pseudo-Boolean formulation of the PMP is presented; its computational possibilities are discussed and demonstrated by means of extensive experiments.

Chapter 3 is focused on the PMP-based model for cell formation. It is shown that PMP-based models, though being an approximation to the cell formation problem, provide high-quality solutions and outperform other contemporary heuristics. At the same time, if an efficient PMP formulation (like the one discussed in Chapter 2) is used, the computing times are negligibly small even for the largest CF instances occurring in practice.

Chapter 4 deals with an exact model for cell formation. It is shown that the latter is equivalent to the minimum multicut problem (that we abbreviate as MINpCUT). Two MILP formulations are also presented and their effectiveness is demonstrated by means of computational experiments with real industrial data. Further, it is shown that a reasonable similarity measure corresponds to the amount of parts travelling directly between a pair of machines, therefore, sequencing information is of particular importance for optimal cell formation.

Chapter 5 discusses appropriateness of the standard objective (minimisation of parts flow between cells) and considers other possible objectives for cell formation, as well as the ways of their introduction into the proposed models (first of all, via the similarity measure).

Finally, Chapter 6 summarises the major results of the thesis and provides concluding remarks.

Chapter 2

The p-Median problem

2.1 Introduction

The p-Median problem (PMP) is a well-known NP-hard problem which was originally defined by Hakimi (1964; 1965) and involves location of p facilities on a network in such a manner that the total weighted distance of serving all demands is minimized. It has been widely studied in literature and applied in cluster analysis, quantitative psychology, marketing, telecommunications industry (Brusco & Köhn, 2008), sales force territories design (Mulvey & Beck, 1984), political districting (Belenky, 2008), optimal diversity management (Briant & Naddef, 2004), cell formation in group technology (Won & Lee, 2004), vehicle routing (Koskosidis & Powell, 1992), and topological design of computer communication networks (Pirkul, 1987).

The basic PMP model that has remained almost unchanged during recent 30 years is the so called ReVelle and Swain integer linear programming formulation (ReVelle & Swain, 1970; Church, 2008). Note that this formulation contains Boolean decision variables and, hence, this is a Boolean linear programming formulation. Since then, the PMP has been the subject of considerable research involving the development of some different types of adjusted model formats (Rosing et al., 1979; Cornuejols et al., 1980; Dearing et al., 1992; Church, 2003), and recently by Church (2008), AlBdaiwi et al. (2009), and Elloumi (2010), as well as the development of advanced solution approaches (Reese (2006) and references within) and some recent

This chapter is based on the paper (Goldengorin & Krushinsky, 2011a).

publications by Senne et al. (2005), Beltran et al. (2006), Avella et al. (2007), Brusco & Köhn (2008). For a comprehensive list of references to the PMP we address the reader to Reese (2006), Mladenovic et al. (2007) and a bibliographical overview by ReVelle et al. (2008).

A *Boolean linear programming formulation of the PMP* can be defined on a weighted bipartite graph $G = (V, A, C)$ with the set of vertices $V = I \cup J$, the set of arcs $A \subseteq I \times J$, and non-negative weights $C = \{c_{ij} : c_{ij} \geq 0, (i, j) \in A\}$ as follows. For the given sets $I = \{1, 2, \ldots, m\}$ of sites at which plants (cluster centres) can be located, $J = \{1, 2, \ldots, n\}$ of clients (cluster points) with unit demand at each client site, a matrix $C = [c_{ij}]$ of non-negative costs (distances, or some other dissimilarity measure) of supplying each $j \in J$ from each $i \in I$, the number p of plants to be opened, the PMP can be written as

$$\min \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, \ldots, n \tag{2.2}$$

$$x_{ij} \leq \overline{y}_i, \quad i = 1, \ldots, m; \quad j = 1, \ldots, n \tag{2.3}$$

$$\sum_{i=1}^{m} \overline{y}_i = p \tag{2.4}$$

$$\overline{y}_i \in \{0, 1\}, \quad i = 1, \ldots, m \tag{2.5}$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, \ldots, m; \quad j = 1, \ldots, n. \tag{2.6}$$

For any feasible solution $(x_{ij}, \overline{y}_i)$, $\overline{y}_i = 1$ if plant i is open, and $\overline{y}_i = 0$, otherwise; $x_{ij} = 1$ if client j is assigned to plant i, and $x_{ij} = 0$, otherwise. Constraints (2.2) assign each client to exactly one plant, constraints (2.3) forbid the assignment of a client to a closed plant, constraint (2.4) fixes the number of opened plants to p.

A PMP instance is described by an $m \times n$ matrix $C = [c_{ij}]$ and the number $1 \leq p \leq |I|$. We assume that the entries of C are nonnegative and finite, i.e. $C \in \mathbb{R}_{+}^{mn}$. If $I = J$, we have the classic ReVelle and Swain's PMP model (ReVelle & Swain, 1970) with n^2 Boolean decision variables.

Further progress with improvements of ReVelle and Swain's PMP model was made by Rosing et al. (1979), Cornuejols et al. (1980), Dearing et al. (1992), Church (2003), Church (2008), and recently by AlBdaiwi et al. (2009) and Elloumi (2010). All of them have incorporated in different ways the following properties of PMP:

(i) based on an ordering of the distances c_{ij} with respect to a given demand point they have either reduced the number of clients or have excluded from (2.1)–(2.6) a repetition of decision variables x_{ij} and x_{kj} corresponding to the equal costs $c_{ij} = c_{kj}$ for some $j \in J$;

(ii) the $mn + m$ Boolean decision variables are replaced by m Boolean decision variables and mn non-negative decision variables, i.e. (2.6) is replaced by

$$x_{ij} \geq 0, \quad i = 1, \ldots, m; \quad j = 1, \ldots, n. \tag{2.7}$$

To the best of our knowledge there is no PMP model that adjusts the numbers of non-negative decision variables and corresponding linear constraints depending on the number p of medians.

In this chapter we start our study of the classic p-median model represented as a Boolean linear programming model by posing the following questions: (i) what are the optimal numbers of decision variables partitioned into Boolean and non-Boolean variables; (ii) what is the optimal number of constraints; (iii) are the above mentioned numbers of decision variables and constraints dependent on the PMP input data, more specifically on the number p of medians.

This chapter proposes a new model formulation for the PMP that contains all previously suggested improvements which we have incorporated in a concise and simplified notation including our adjustment of decision variables and corresponding linear constraints depending on the number p of medians. This new p-median formulation is called a *Mixed Boolean Pseudo-Boolean Model* for the PMP. We show that our model can result in a substantially smaller mixed Boolean linear programming formulation for a given application of the PMP and can be used either to find a global optimum by means of general-purpose MILP solvers or to develop new exact and approximate algorithms based on the well known methods in mixed integer programming (see, e.g., Wolsey, 2008).

Some of above mentioned improvements were separately done for the PMP without taking into account the ongoing progress with model formulations for another common model within minisum location-allocation problems, namely the Simple Plant Location Problem (SPLP), often referred to as the Uncapacitated Facility Location Problem (UFLP) (see Cornuejols et al., 1990) or the warehouse location problem (see, e.g., ReVelle et al., 2008). The SPLP is similar to the PMP, and the methods used to solve one are often adapted to solve the other. The objective function of the SPLP is one of determining the cheapest method of meeting the demands of a set of clients $J = \{1, \ldots, n\}$ from plants that can be located at some candidate sites $I = \{1, \ldots, m\}$. The costs involved in meeting the client demands include the fixed cost of setting up a plant at a given site, and the per unit transportation cost of supplying a given client from a plant located at a given site. Both PMP and SPLP are defined on bipartite graphs and differ in the following details. First, the SPLP involves a fixed cost for locating a facility at a given vertex while the PMP does not. Second, unlike the PMP, SPLP does not have a constraint on the number of opened facilities. Typical SPLP formulations separate the set of potential facilities (sites location, cluster centres) from the set of demand points (clients). In the PMP these sets are identical, i.e. $I = J$. Such problems are well known in cluster analysis (see, e.g., Brusco & Köhn, 2008). Both problems form underlying models in several combinatorial problems, like set covering, set partitioning, information retrieval, simplification of logical Boolean expressions, airline crew scheduling, vehicle dispatching (see Christofides, 1975), assortment (see, e.g., Goldengorin et al., 2003; Pentico, 2008), and are subproblems of various location analysis problems (see ReVelle et al., 2008).

An instance of the SPLP has an optimal solution in which each client is satisfied by exactly one plant. A similar observation is valid for the PMP. In Hammer (1968) (see also Dearing et al., 1992) this fact is used to derive a pseudo-Boolean representation of the SPLP. The pseudo-Boolean polynomial (pBp) developed in that work has terms that contain both a literal and its complement. At the end of (Hammer, 1968) it is shown by means of an example that only linear monomials can have negative coefficients. Subsequently, in Beresnev (1973) a different pseudo-Boolean form has been developed in which each term contains only literals or only their complements. We have found this form easier to manipulate, and hence adjusted Beresnev's formulation of the SPLP to the PMP in (AlBdaiwi et al., 2009) and (Goldengorin & Krushinsky, 2011a).

The purpose of this chapter is twofold. First, we design a new model for the p-Median problem and show that the number of nonnegative decision variables and corresponding constraints depend on the number of p-medians and will be adjusted in our model. Moreover, these numbers are minimal within the class of Mixed Integer Linear Programs for the PMP. Second, we show that our new model allows solving by means of a general-purpose solver on a PC some PMP benchmark instances previously intractable by both general-purpose solvers and the state-of-the-art exact algorithms, as well as handling smaller instances more efficiently.

In order to demonstrate the properties of the PMP and compare performance of the formulations we used benchmark instances from the four most popular libraries: OR, TSP, ODM, RW. The first one, the OR library, was introduced by Beasley

(1985) and is available at (OR Library, 1990). Every node is both a potential location and a client, and the costs are the lengths of the shortest paths between the corresponding nodes.

The TSP library was originally proposed for the travelling salesman problem (TSP) and is available at (TSP Library, 1995). TSP instances are defined as sets of points in a two dimensional plane. Every point is considered both a potential location and a client, and the costs are simply Euclidean distances.

Instances from the next library that we studied are based on the optimal diversity management (ODM) problem. For the description of this problem and instances see Briant & Naddef (2004).

Finally, we considered instances proposed by Resende & Werneck (2003). These problems are defined on random distance matrices. In every case the number of potential facilities m is equal to the number of clients n and distances are integers taken uniformly at random from the interval $[1, n]$. The library contains five instances with $n = 100, 200, 250, 500, 1000$.

The chapter is organized as follows. Section 2.2 focuses on the pseudo-Boolean formulation of the PMP and its basic properties. In Section 2.3 we analyse the size reduction techniques applicable to the p-Median problem. Next, in Section 2.4 we present our new MBpBM formulation, discuss its minimality and provide results of numerical experiments. Sections 2.5 and 2.6 provide two applications of the pseudo-Boolean formulation: estimation of instance data complexity and characterization of equivalent instances. Finally, Section 2.7 concludes the chapter with a summary and future research directions.

2.2 The pseudo-Boolean representation

Recall that given sets $I = \{1, 2, \dots, m\}$ of sites in which plants can be located, $J = \{1, 2, \dots, n\}$ of clients, a matrix $C = [c_{ij}]$ of transportation costs (supplying costs, distances, similarities, etc.) for each $j \in J$ from each $i \in I$, the number p of plants to be opened, and a unit demand at each client site, the p-Median Problem (PMP) is one of finding a set $S \subseteq I$ with $|S| = p$, such that the total cost

$$f_C(S) = \sum_{j \in J} \min\{c_{ij} \mid i \in S\} \tag{2.8}$$

of satisfying all unit demands is minimized. Note that non-unit demands $d_j \neq 1$ can be scaled by $c'_{ij} = c_{ij} d_j$, and the number of served clients by each plant is unbounded (the so called *uncapacitated* location problem, see e.g., Reese (2006) and ReVelle et al. (2008)). An instance of the problem is described by an $m \times n$ matrix $C = [c_{ij}]$ and the number $1 \leq p \leq |I|$. We assume that entries of C are nonnegative and finite, i.e. $C \in \mathbb{R}^{mn}_{+}$. The *Combinatorial Formulation of PMP* is to find

$$S^{\star} \in \arg\min\{f_C(S) : \emptyset \subset S \subseteq I,\ |S| = p\}. \tag{2.9}$$

It is possible to reformulate the objective function $f_C(S)$ of PMP (2.8) in terms of a pseudo-Boolean polynomial (see Hammer (1968), Beresnev (1973)). It is enough to find a pseudo-Boolean representation for each addend $\min\{c_{ij} \,|\, i \in S\}$, and sum up addends for all $j \in J$. In the rest of this section we will use the following notions. Mappings $f : \{0,1\}^n \to$ R are called pseudo-Boolean functions. All pseudo-Boolean functions can be uniquely represented as *multi-linear polynomials* of the form (see, e.g., Boros & Hammer, 2002)

$$f(\mathbf{x}) = \sum_{S \subseteq I} \alpha_S \prod_{i \in S} x_i. \tag{2.10}$$

The expressions $\alpha_S \prod_{i \in S} x_i$ and $\prod_{i \in S} x_i$ are called a *monomial* and a *term*, respectively. In this paper multi-linear polynomials are called *pseudo-Boolean polynomials* and monomials with the same term are called *similar monomials*. For example, the following pairs of monomials $2x_1x_5$ and $5x_1x_5$; $3x_3x_4x_7$ and $3x_3x_4x_7$ are similar monomials. We say that a pseudo-Boolean polynomial is in *the reduced form* if it contains no similar monomials. In other words, the algebraic summation of similar monomials is called *reduction*. Representation of the cost function (2.8) in terms of a pseudo-Boolean polynomial needs two additional notions: an *ordering matrix* and a *differences matrix*.

An $m \times n$ *ordering matrix* $\Pi = [\pi_{ij}]$ is a matrix with each column $\Pi_j = (\pi_{1j}, \ldots, \pi_{mj})^{\mathrm{T}}$ defining a permutation of $1, \ldots, m$ that if being applied to the corresponding column of the costs matrix makes its entries sorted in a non-decreasing order. There may exist several ordering matrices for a given instance of the PMP. Given a matrix C, the set of all ordering matrices Π such that $c_{\pi_{1j}j} \leq c_{\pi_{2j}j} \leq \cdots \leq c_{\pi_{mj}j}$ for $j = 1, \ldots, n$ is denoted by $perm(C)$.

Consider the expression $\min\{c_{ij} \,|\, i \in S\}$ for some fixed $j \in J$. Clearly, the minimum is attained if $S = I$, i.e. the smallest value is chosen among all entries c_{ij} for a fixed column j. It is clear that the unit demand of column j cannot be satisfied cheaper than this smallest value. Assume that this smallest value is attained at an entry $c_{\pi_{1j}j}$ of column j such that π_{1j} indicates the number of the row containing this smallest entry $c_{\pi_{1j}j}$ in column j. In terms of the original PMP, if the site numbered by π_{1j} is open, then the unit demand of client j will be satisfied by costs $c_{\pi_{1j}j}$, otherwise (if the site π_{1j} is closed, but all other sites in $I \setminus \{\pi_{1j}\}$ are opened) the cheapest way to satisfy the unit demand of client j is by the value of a second smallest entry $c_{\pi_{2j}j}$. The value of a second smallest entry $c_{\pi_{2j}j}$ can be represented as follows: $c_{\pi_{2j}j} = c_{\pi_{1j}j} + [c_{\pi_{2j}j} - c_{\pi_{1j}j}]$. Similarly, if both sites π_{1j}, π_{2j} are closed and all other sites are opened, then the unit demand of client j will be satisfied by the value of a third smallest entry $c_{\pi_{3j}j} = c_{\pi_{1j}j} + [c_{\pi_{2j}j} - c_{\pi_{1j}j}] + [c_{\pi_{3j}j} - c_{\pi_{2j}j}]$, etc. In other words, depending on the set of opened and closed sites from I the corresponding smallest value of $\min\{c_{i,j} \mid i \in S\}$ can be represented by the sum of the smallest values of entries in column j and the corresponding differences of ordered entries in column j. By introducing a Boolean variable $y_{\pi_{1j}} = 0$ if the site π_{1j} is opened, and $y_{\pi_{1j}} = 1$ if the site π_{1j} is closed, we are able to express, for example, the costs of satisfying the unit demand j depending on whether the site π_{1j} is opened or closed (if π_{1j} is closed then we assume that π_{2j} is open, i.e. $y_{\pi_{2j}} = 0$), as follows: $c_{\pi_{2j}j} = c_{\pi_{1j}j} + [c_{\pi_{2j}j} - c_{\pi_{1j}j}]y_{\pi_{1j}}$.

To illustrate this idea, let us consider the first column C^1 of matrix C, namely $C^1 = (c_{11}, c_{21}, c_{31}, c_{41})^{\mathrm{T}} = (7, 10, 16, 11)^{\mathrm{T}}$. After ordering its entries in a non-decreasing order $7 < 10 < 11 < 16$ we have that the corresponding permutation is $\Pi^1 = (1, 2, 4, 3)^{\mathrm{T}}$. If the Boolean vector $(y_1, y_2, y_3, y_4)^{\mathrm{T}}$ reflects an opened (closed) plants at cite $i = 1, 2, 3, 4$, then depending on the set of opened plants $S \subseteq \{1, 2, 3, 4\}$ we have $\min\{c_{i1} \mid i \in S\} = [7 + 3y_1 + 1y_1y_2 + 5y_1y_2y_4]$. For example, if $S = \{2, 4\}$, then $\mathbf{y} = (1, 0, 1, 0)^{\mathrm{T}}$, and $\min\{c_{i1} \mid i \in \{2, 4\}\} = 7 + 3 \times 1 + 1 \times 1 \times 0 + 5 \times 1 \times 0 \times 0 = 10$.

Corresponding to an ordering matrix $\Pi = [\pi_{ij}]$, a differences matrix $\Delta = \delta_{ij}$ containing differences between the transportation costs for each $j \in J$ is uniquely defined as follows:

$$\begin{aligned} \delta_{1k} &= c_{\pi_{1k}k} \\ \delta_{rk} &= c_{\pi_{rk}k} - c_{\pi_{(r-1)k}k} \text{ for } 2 \le r \le m. \end{aligned} \qquad (2.11)$$

Defining

$$y_i = \begin{cases} 0 & \text{if } i \in S \\ 1 & \text{otherwise,} \end{cases} \quad \text{for each } i = 1, \ldots, m \qquad (2.12)$$

we can indicate any solution S by a vector $\mathbf{y} = (y_1, y_2, \ldots, y_m)^{\mathrm{T}}$. Its total cost is given by the following pseudo-Boolean polynomial:

$$\mathcal{B}_{C,\Pi}(\mathbf{y}) = \sum_{j=1}^{n} \left\{ \delta_{1j} + \sum_{k=2}^{m} \delta_{kj} \cdot \prod_{r=1}^{k-1} y_{\pi_{rj}} \right\}. \tag{2.13}$$

Note, this pseudo-Boolean polynomial is different from those used by Hammer (1968) and Dearing et al. (1992) containing both variables and their complements.

We call a pseudo-Boolean polynomial $f(\mathbf{y})$ (2.10) a *Hammer-Beresnev polynomial* if there exists a PMP instance C and $\Pi \in perm(C)$ such that $f(\mathbf{y}) = \mathcal{B}_{C,\Pi}(\mathbf{y})$ for each $\mathbf{y} \in \{0,1\}^m$, since this representation of the total cost was first presented in the context of uncapacitated facility location problems independently in Hammer (1968) and Beresnev (1973). The following theorem from AlBdaiwi et al. (2011) gives necessary and sufficient conditions for this.

Theorem 1. *A general pseudo-Boolean polynomial is a Hammer-Beresnev polynomial if and only if all its coefficients are nonnegative.*

Proof. The "if" statement is trivial. In order to prove the "only if" statement, consider a PMP instance defined by the cost matrix C, an ordering matrix $\Pi \in perm(C)$, and a Hammer-Beresnev polynomial $\mathcal{B}_{C,\Pi}(\mathbf{y})$ in which there is a monomial of degree k with a negative coefficient. Since monomials in $\mathcal{B}_{C,\Pi}(\mathbf{y})$ are contributed by the elements of C only, a monomial with a negative coefficient implies that $\delta_{k,j}$ is negative for some $j \in 1, \ldots, n$. But this contradicts the fact that $\Pi \in perm(C)$. ■

In AlBdaiwi et al. (2009) it is shown that the total cost function (2.13) is identical for all permutations in $perm(C)$. Hence, we can remove the Π in $\mathcal{B}_{C,\Pi}(\mathbf{y})$ without introducing any confusion. We denote a Hammer-Beresnev function corresponding to a given PMP instance C by $\mathcal{B}_C(\mathbf{y})$ and define it as

$$\mathcal{B}_C(\mathbf{y}) = \mathcal{B}_{C,\Pi}(\mathbf{y}), \tag{2.14}$$

where $\Pi \in perm(C)$.

A solution $\mathbf{y}$ is feasible if $\sum_{i=1}^{m} y_i = m - p$. Thus, every product of more than $m - p$ variables is 0 for any feasible solution. This observation allows excluding monomials of high degree from the objective function and we call this procedure *truncation of the Hammer-Beresnev polynomial*. The polynomial subjected to truncation and summation of similar monomial is denoted by $\mathcal{B}_{C,p}(\mathbf{y})$ and has the following form:

$$\mathcal{B}_{C,p}(\mathbf{y}) = \sum_{j=1}^{n} \left\{ \delta_{1j} + \sum_{k=2}^{m-p+1} \delta_{kj} \cdot \prod_{r=1}^{k-1} y_{\pi_{rj}} \right\}. \tag{2.15}$$

It should be mentioned that the truncated Hammer-Beresnev polynomial $\mathcal{B}_C(\mathbf{y})$ usually contains less than $(m-p) \times n$ monomials as presence of equal entries in columns of the cost matrix leads to zero differences δ_{kj} and similar monomials can be subjected to algebraic summation (e.g., constants δ_{1j} can be always summed up into one value). Further, we will denote the truncated Hammer-Beresnev polynomial with reduced similar monomials by $\mathcal{B}_{C,p}(\mathbf{y})$, it can be expressed as:

$$\mathcal{B}_{C,p}(\mathbf{y}) = \sum_{r=0}^{k} \alpha_r \prod_{i \in T_r} y_i = \sum_{r=0}^{k} \alpha_r \mathcal{T}_r, \tag{2.16}$$

where T_r is a set of Boolean variables y_i included in term $\mathcal{T}_r$, k is the number of non-constant monomials in $\mathcal{B}_{C,p}(\mathbf{y})$.

We can reformulate (2.9) in terms of Hammer-Beresnev polynomials as the *pseudo-Boolean formulation of PMP*:

$$\mathbf{y}^\star \in \arg\min\{\mathcal{B}_{C,p}(\mathbf{y}) : \mathbf{y} \in \{0,1\}^m, \sum_{i=1}^{m} y_i = m - p\}. \tag{2.17}$$

Example. Consider a PMP instance borrowed from Elloumi (2010) with $m = 4$, $n = 5$, $p = 2$ and

$$C = \begin{bmatrix} 1 & 6 & 5 & 3 & 4 \\ 2 & 1 & 2 & 3 & 5 \\ 1 & 2 & 3 & 3 & 3 \\ 4 & 3 & 1 & 8 & 2 \end{bmatrix}. \tag{2.18}$$

A possible ordering matrix for this problem is given by

$$\Pi = \begin{bmatrix} 1 & 2 & 4 & 1 & 4 \\ 3 & 3 & 2 & 2 & 3 \\ 2 & 4 & 3 & 3 & 1 \\ 4 & 1 & 1 & 4 & 2 \end{bmatrix} \tag{2.19}$$

and the differences matrix is

$$\Delta = \begin{bmatrix} 1 & 1 & 1 & 3 & 2 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 2 & 3 & 2 & 5 & 1 \end{bmatrix} \tag{2.20}$$

The Hammer-Beresnev polynomial representing the total cost function for this instance in the form (2.13) is

$$\begin{aligned} \mathcal{B}_C(\mathbf{y}) = \quad & [1 + 0y_1 + 1y_1y_3 + 2y_1y_2y_3] + \\ & [1 + 1y_2 + 1y_2y_3 + 3y_2y_3y_4] + \\ & [1 + 1y_4 + 1y_2y_4 + 2y_2y_3y_4] + \\ & [3 + 0y_1 + 0y_1y_2 + 5y_1y_2y_3] + \\ & [2 + 1y_4 + 1y_3y_4 + 1y_1y_3y_4]. \end{aligned} \tag{2.21}$$

Taking into account that $p = 2$, after truncation and reduction of similar monomials in (2.21) we obtain the following pseudo-Boolean representation of the instance:

$$\begin{aligned} \mathcal{B}_{C,p=2}(\mathbf{y}) = \quad & 8 + 1y_2 + 2y_4 + 1y_1y_3 + 1y_2y_3 + 1y_2y_4 + 1y_3y_4 \to \min \\ & s.t. \\ & y_1 + y_2 + y_3 + y_4 = m - p = 2 \\ & \mathbf{y} \in \{0,1\}^m \end{aligned} \tag{2.22}$$

◁

It is easy to see that the objective function in 2.22 contains only 7 non-zero coefficients while the initial costs matrix (2.18) has 20 entries. This implies that the pseudo-Boolean representation allows reduction of the memory needed to store the PMP instance data. Of course, not only coefficients but also terms must be stored, however, these overheads in most cases will be overwhelmed by the substantial reduction of the polynomial.

2.3 Reduction techniques

The pseudo-Boolean representation of a PMP instance has very attractive properties, which we are going to consider in the rest of this chapter. First of all, a pseudo-Boolean polynomial can be subjected to several quite straightforward types of reductions.

2.3.1 Reduction of the number of monomials in the pBp

Recall that given a variable vector $\mathbf{y} = (y_1, y_2, \ldots, y_m)^T$, the expressions $\mathcal{T} = \prod_{i \in T} y_i$ and $\alpha\mathcal{T} = \alpha \prod_{i \in T} y_i$ ($T \subseteq \{1, \ldots, m\}$, $\alpha \in \mathbb{R}$) are called a *term* and a *monomial*, respectively. We also call two monomials *similar* if their terms are identical. Finally, by *reduction of monomials* we mean algebraic summation of similar monomials.

Reduction of the number of monomials in pBp consists of three stages. First, as some locations may have equal distance to several clients, the corresponding entries in the differences matrix are zero and the number of terms in the polynomial is usually less than mn (see column #T in Table 2.1). This reduction is similar to the one introduced by many authors (Beresnev, 1973; Cornuejols et al., 1980; Elloumi, 2010; Church, 2003; Dearing et al., 1992) and can be illustrated by the following small example: let $m = 4$, $n = 5$ and the costs matrix is

$$C = \begin{bmatrix} 7 & 15 & 10 & 7 & 10 \\ 10 & 17 & 4 & 11 & 22 \\ 16 & 7 & 6 & 18 & 24 \\ 11 & 7 & 6 & 12 & 8 \end{bmatrix}. \tag{2.23}$$

A possible permutation matrix and the corresponding difference matrix are

$$\Pi = \begin{bmatrix} 1 & 3 & 2 & 1 & 4 \\ 2 & 4 & 3 & 2 & 1 \\ 4 & 1 & 4 & 4 & 3 \\ 3 & 2 & 1 & 3 & 2 \end{bmatrix} \tag{2.24}$$

and

$$\Delta = \begin{bmatrix} 7 & 7 & 2 & 7 & 8 \\ 3 & 0 & 2 & 4 & 2 \\ 1 & 8 & 0 & 1 & 4 \\ 5 & 2 & 4 & 6 & 8 \end{bmatrix}. \tag{2.25}$$

Thus, the pBp is $\mathcal{B}_C = [7 + 3y_1 + 1y_1y_2 + 5y_1y_2y_4] + [7 + 0y_3 + 8y_3y_4 + 2y_1y_3y_4] + [4+2y_2+0y_2y_3+4y_2y_3y_4]+[7+4y_1+1y_1y_2+6y_1y_2y_4]+[8+2y_4+4y_1y_4+8y_1y_3y_4]$. As there are two zeroes in the differences matrix, the initial (in contrast to reduced and truncated) pBp has $mn - 2 = 18$ nonzero terms (we will denote this characteristic by $\#T$).

Second, the pBp can be subjected to reducing similar monomials (by its essence, it corresponds to the second reduction rule from Elloumi (2010), p. 11). In the considered example this procedure leads to a polynomial $33 + 7y_1 + 2y_2 + 2y_4 + 2y_1y_2 + 8y_3y_4 + 4y_1y_4 + 11y_1y_2y_4 + 10y_1y_3y_4 + 4y_2y_3y_4$ with 10 monomials. We denote the number of monomials in such pBp with reduced similar monomials by $\#T_r$.

Finally, as shown in (AlBdaiwi et al., 2009), for any feasible solution $\mathbf{y}$ the value of truncated polynomial $\mathcal{B}_{C,p}$ obtained from $\mathcal{B}_C$ by deleting all terms of degree higher that $(m - p)$ is equal to the value of the initial pBp. For example, $\mathcal{B}_C = 33+7y_1+2y_2+2y_4+2y_1y_2+8y_3y_4+4y_1y_4+11y_1y_2y_4+10y_1y_3y_4+4y_2y_3y_4$ with $p = 2$, i.e. $\mathcal{B}_{C,2} = 33 + 7y_1 + 2y_2 + 2y_4 + 2y_1y_2 + 8y_3y_4 + 4y_1y_4$ has just seven monomials. This makes it possible for the particular problem with fixed number of medians to truncate the polynomial thus reducing its size to at most $(m - p) \cdot n$.

In order to determine the effect of the mentioned above techniques, a number of experiments with instances from the four libraries were carried. Results of pseudo-Boolean formulation and reduction of similar monomials for typical representatives of each library are given in Table 2.1. We computed reduction (see the rightmost column of the table) as $(mn - \#T_r)/mn \times 100\%$. As can be seen from the table, instances from OR library allow the highest reduction of the number of terms in the pBp. For example, for the instance pmed40 the size of the polynomial is about 4% of the number of entries in the costs matrix. So, from the point of view of our notion of complexity, these instances are the easiest ones. Instances from TSP and RW libraries also allow compact representation of the polynomial, while ODM instances are the most complex ones and allow only minor reduction of the number of terms.

Table 2.1. Reduction of the pBp for benchmark instances

			Entries in			reduction
library	instance	m	matrix C	$\#T$	$\#T_r$	(%)
OR	pmed1	100	10,000	7,506	6,722	32.78
OR	pmed15	300	90,000	20,182	17,428	80.64
OR	pmed26	600	360,000	29,963	25,694	92.86
OR	pmed40	900	810,000	36,326	31,642	96.09
ODM	BN48	42	411	411	329	19.95
ODM	BN1284	1284	88,542	88,447	85,416	3.53
ODM	BN3773	3773	349,524	348,063	341,775	2.22
ODM	BN5535	5535	666,639	665,577	654,709	1.79
TSP	rd100	100	9,900	9,394	9,243	6.63
TSP	D657	657	430,992	368,233	367,355	14.77
TSP	fl1400	1400	1,958,600	838,110	836,557	57.29
TSP	pcb3038	3038	9,226,406	5,763,280	5,759,404	37.58
RW	rw100	100	10,000	6,357	6,232	37.68
RW	rw200	200	40,000	25,351	25,099	37.25
RW	rw250	250	62,500	39,542	39,228	37.24
RW	rw500	500	250,000	158,007	157,362	37.06
RW	rw1000	1000	1,000,000	631,805	630,543	36.95

Of certain interest is a relation between instance size and the achieved reduction (rightmost column in Table 2.1). For OR and TSP libraries this factor tends to increase for larger problems implying that pseudo-Boolean representation is efficient for large instances from these classes. However, for ODM library the situation is opposite, so from this point of view ODM instances are also hard. With randomized graphs from RW library the reduction ratio is almost constant, so these instances are somewhere in between the previous two groups.

Despite the differences in performance between the above mentioned libraries, truncation of the polynomial has similar impact on the required space for all the considered instances (resulting in at most $(m - p) \cdot n$ entries). We have observed that nonzero entries are uniformly distributed over the rows of the differences matrix Δ (in other words, the numbers of nonzero monomials of different degrees are approximately the same). It means that with increasing p the number of monomials in the truncated polynomial $\mathcal{B}_{C,p}$ decreases in a linear fashion from $\#T_r$ to 1 (if $p = m$ the polynomial is just a constant). Moreover, if we denote by $p^* \leq m$ the (minimum) number of rows that contain all minima in columns, then the polynomial reduces to a constant for $p \geq p^*$.

2.3.2 Reduction of the number of clients (columns)

In order to show why the reduction of the number of clients (columns) is possible we have to give the following definitions.

Definition 1. *Two PMP instances defined on costs matrices C and D are called* equivalent *if C and D are of the same size (number of rows) and* $\mathcal{B}_{C,p}(\mathbf{y}) = \mathcal{B}_{D,p}(\mathbf{y})$.

Definition 2. *Having an* $m \times n$ *costs matrix C, by* aggregation of clients *(columns) we mean construction of such* $m \times n'$ *matrix D that* $\mathcal{B}_{C,p}(\mathbf{y}) = \mathcal{B}_{D,p}(\mathbf{y})$ *and* $n' < n$.

This means that if there exist some costs matrix D that leads to the same polynomial as C and D has fewer columns, then the p-Median problem defined on C can be substituted by the problem defined on D. So, the question is: given a costs matrix C and the number of medians p, find such a matrix D that corresponds to the same truncated polynomial as C and has the minimum possible number of columns.

The idea behind this type of processing is as follows. Each chain of embedded terms in a pBp corresponds to some permutation and a column of differences. At the same time, over the terms of the polynomial it is possible to define a relation of partial order, that, in turn, can be represented by the Hasse diagram. It is clear that all the terms can be covered by n chains that correspond to n columns of the differences matrix. It means that all vertices of the Hasse diagram can be covered by n (internally) vertex disjoint chains. However, observation that for some instances reduction of similar monomials leads to a substantial decrease in their number suggests a possibility that all terms can be covered by fewer chains. Having a chain of embedded terms it is possible to reconstruct a permutation and a row of the differences matrix. Thus, reduced number of chains covering all terms implies reduced number of clients in the aggregated matrix and the problem of finding the smallest n' is reduced to finding the minimum number of chains that cover all terms of the polynomial (or all vertices of the corresponding Hasse diagram). According to the well-known Dilworth's decomposition theorem (see, e.g., Theorem 14.2 in Schrijver (2003), p.218), this minimal number of chains is equal to the maximum size of an antichain (in our case it is the maximum number of non-embedded terms).

In order to compute the minimum number of chains we used the MINLEAF algorithm described in (Gutin et al., 2008) that constructs a minimum leaf outbranching. (MINLEAF is a polynomial-time algorithm and is essentially based on finding the maximum cardinality matching.) Having such an outbranching it is possible to reconstruct the chains such that the number of chains is equal to the number of leaves in the outbranching. After that, an equivalent matrix, each column of which is induced by one of the obtained chains, can be restored. As in the formulation of the p-Median problem each column of the costs matrix corresponds to a client whose demand is to be satisfied, existence of the equivalent matrix with smaller number of columns implies that in the initial instance some clients can be aggregated.

Within the mentioned above small example (2.23) this procedure leads to the following. The reduced pseudo-Boolean polynomial $\mathcal{B}_C(\mathbf{y}) = 33 + 7y_1 + 2y_2 + 2y_4 + 2y_1y_2 + 8y_3y_4 + 4y_1y_4 + 11y_1y_2y_4 + 10y_1y_3y_4 + 4y_2y_3y_4$ corresponds to the following Hasse diagram:

$$\begin{tikzcd}
 & y_2 \arrow[r] & y_1y_2 \arrow[r] & y_1y_2y_4 \arrow[dr] & \\
const \arrow[r] \arrow[ur] \arrow[dr] & y_1 \arrow[r] \arrow[ur] & y_1y_4 \arrow[r] \arrow[ur] & y_1y_3y_4 \arrow[r] & y_1y_2y_3y_4 \\
 & y_4 \arrow[r] \arrow[ur] & y_3y_4 \arrow[r] \arrow[ur] & y_2y_3y_4 \arrow[ur] &
\end{tikzcd} \tag{2.26}$$

It is easy to check that the size of the maximum antichain is 3, so all the terms of $\mathcal{B}_C(\mathbf{y})$ can be covered by three chains and the aggregated matrix has three columns. Below are the chains (each being presented as a column), permutation and differences matrices:

$$\begin{array}{lll}
y_2 & y_1 & y_4 \\
y_1y_2 & y_1y_4 & y_3y_4 \\
y_1y_2y_4 & y_1y_3y_4 & y_2y_3y_4 \\
y_1y_2y_3y_4 & y_1y_2y_3y_4 & y_1y_2y_3y_4
\end{array} \tag{2.27}$$

$$\Pi' = \begin{bmatrix} 2 & 1 & 4 \\ 1 & 4 & 3 \\ 4 & 3 & 2 \\ 3 & 2 & 1 \end{bmatrix} \quad \Delta' = \begin{bmatrix} 0 & 0 & 33 \\ 2 & 7 & 2 \\ 2 & 4 & 8 \\ 11 & 10 & 4 \end{bmatrix} \tag{2.28}$$

Having these two matrices it is possible to restore the costs matrix D of the aggregated instance:

$$D = \begin{bmatrix} 2 & 0 & 47 \\ 0 & 21 & 43 \\ 15 & 11 & 35 \\ 4 & 7 & 33 \end{bmatrix} \tag{2.29}$$

Experiments

As was mentioned above, the minimum number of aggregated clients (columns) does not exceed n. On the other hand, it cannot be smaller then the maximum number of terms with same degree in the reduced polynomial. In particular, for the case of instances from OR library this leads to the following. As the costs matrix for such instances has a zero diagonal, the minimal element of ith column is located in the ith row and the first row of the permutation matrix contains no equal entries. This means that the (reduced) pBp contains n linear terms and cannot be covered by less then n chains. So, the OR instances, if considered "as is", do not allow any aggregation of clients. This result brought us to an idea of considering the corrected instances without zeroes on the diagonal (it is filled by some positive numbers during application of the Floyd's algorithm). Further we mark such instances with an asterisk (e.g., pmed1*). As all the other considered libraries are free of the mentioned "hardness", they can be directly used for experiments with aggregation of clients.

In our experiments we considered truncated polynomials and determined the minimum number of aggregated columns (n') for all values of p from 1 to $m-1$ (if $p = m$, the truncated polynomial is just a constant and it can be covered by one chain). Let us denote by p'' the smallest number of medians at which the truncated polynomial can be covered by less then n chains.

The results for typical representatives from each library are given in Figures 2.1 and 2.2. As can be seen from the figures, for corrected OR and ODM problems $p'' = 0$ and even a non-truncated polynomial can be covered by $n-1$ chains, thus making it possible to aggregate one client. At the same time, for TSP and RW instances any aggregation becomes possible only as p gets very close to m.

Thus, we can summarize that the reduction of the number of clients is negligible for all the considered benchmark libraries.

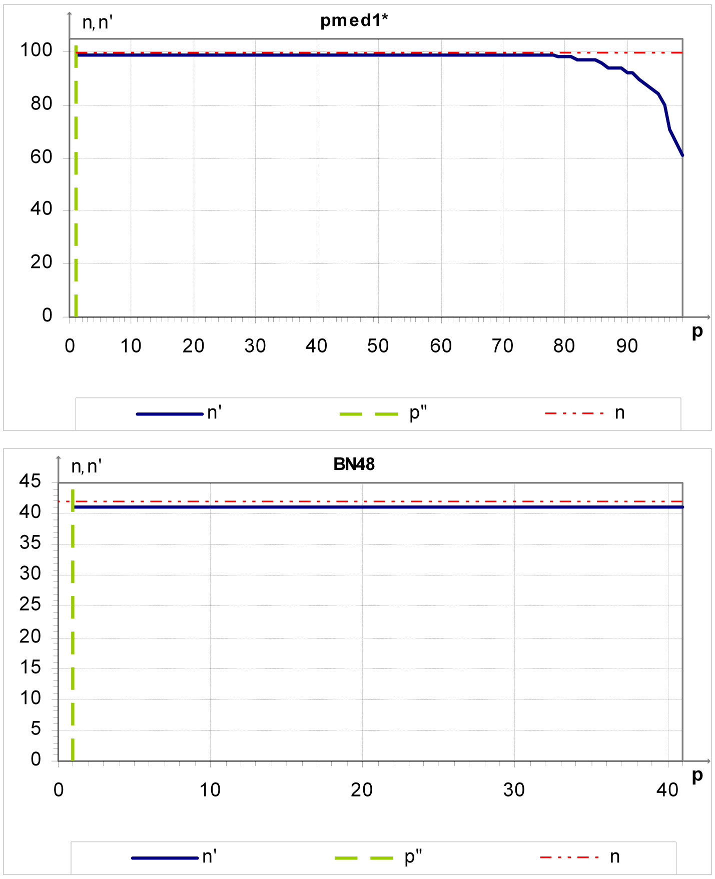


Figure 2.1. Aggregation of clients for benchmark instances from OR and ODM libraries.

2.3.3 Preprocessing

The essence of preprocessing that we consider is to find such locations that can be excluded from consideration as they are not contained in some optimal solution. At the same time, the technique considered in this section is independent of any solution algorithm (e.g., it does not use upper and lower bounds) and is based purely on the structural properties of the input costs matrix.

Let us define the p-truncation operation applied separately to each column of the costs matrix as setting p largest entries to the value of the smallest of them. This procedure ensures that the pBp of the p-truncated matrix is equal to the truncated pBp of the initial matrix. The following theorem (see AlBdaiwi et al. (2009), Theorem 4), provides a direct suggestion for preprocessing based on p-truncation.

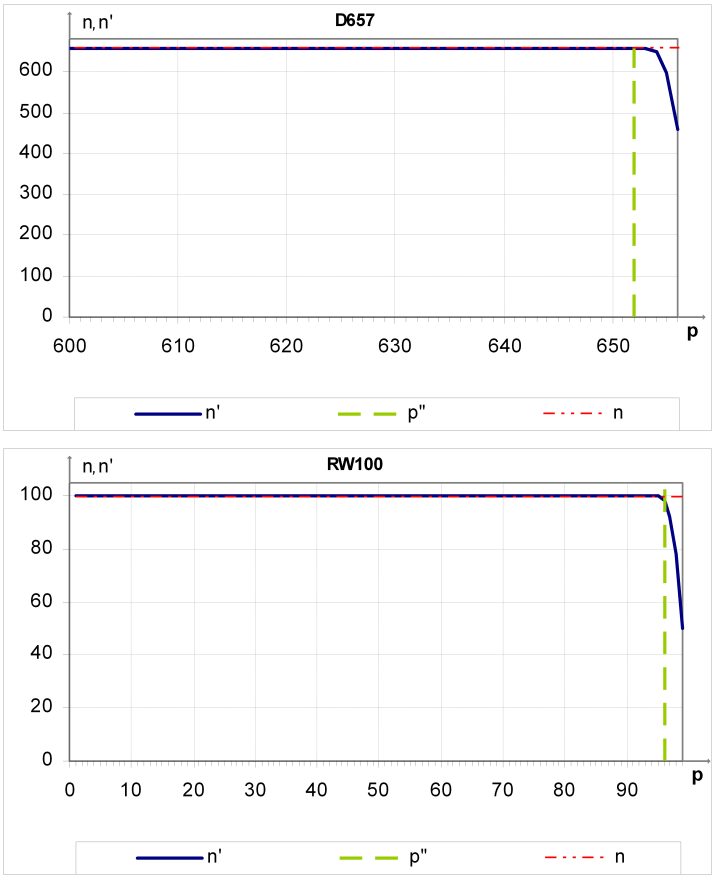


Figure 2.2. Aggregation of clients for benchmark instances from TSP and RW libraries.

Theorem 2. *Assume that in a given PMP instance with $p < m$ some row i in the costs matrix C contains all the columns maxima after p-truncation operations are performed on all columns of C. Then there exist an optimal solution $\mathbf{y}^*$ to the instance with $y_i^* = 1$.*

Proof. The fact that row i in a p-truncated matrix contains all columns maxima implies that location i is among the $m - p$ most expensive locations for every client. This, in turn, means that in a feasible solution each client j can be served cheaper from a different location. Thus, location i can be excluded from consideration and the corresponding y-variable can be fixed to 1. ■

In other words, the theorem means that if some variable y_i is not contained in the truncated polynomial, then there exists an optimal solution $\mathbf{y}^*$ with $y_i^* = 1$. In order to illustrate this we would like to consider the following example. Let the costs matrix be defined as (the rightmost column of numbers enumerates rows of the matrix):

$$C = \begin{bmatrix} 1 & 3 & 9 \\ 2 & 5 & 3 \\ 9 & 7 & 8 \\ 5 & 9 & 7 \\ 4 & 4 & 5 \end{bmatrix} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \tag{2.30}$$

Also, let p be $p = \lceil m/2 \rceil = 3$ (that corresponds to the hardest case from a combinatorial point of view). The p-truncated matrix $C_{p=3}$ is:

$$C_{p=3} = \begin{bmatrix} 1 & 3 & 7 \\ 2 & 5 & 3 \\ 4 & 5 & 7 \\ 4 & 5 & 7 \\ 4 & 4 & 5 \end{bmatrix} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \tag{2.31}$$

The objective function can be represented by the pseudo-Boolean polynomial $\mathcal{B}_C(\mathbf{y}) = 7 + 2y_1 + 2y_2 + 2y_1y_2 + 1y_1y_5 + 2y_2y_5 + 3y_1y_2y_5 + 1y_2y_4y_5 + 4y_1y_2y_4y_5 + 2y_1y_2y_3y_5 + 1y_2y_3y_4y_5$. After truncation one obtains $\mathcal{B}_{C,p=3}(\mathbf{y}) = 7 + 2y_1 + 2y_2 + 2y_1y_2 + 1y_1y_5 + 2y_2y_5$. As can be seen, the truncated pBp does not contain two variables y_3, y_4, so they can be set to 1 as this does not affect the value of $\mathcal{B}_{C,p=3}(\mathbf{y})$. This means that the initial matrix C given by (2.30) can be reduced to matrix D with fewer rows:

$$D = \begin{bmatrix} 1 & 3 & 7 \\ 2 & 5 & 3 \\ 4 & 4 & 5 \end{bmatrix} \begin{array}{c} 1 \\ 2 \\ 5 \end{array} \tag{2.32}$$

It should be noticed, that if one sets all other variables y_1, y_2, y_5 to 0, this immediately gives the optimal solution. Thus, for this small example the problem can be solved just by data preprocessing.

However, with large instances this technique does not always allow solving the problem. Given a PMP costs matrix, we studied how the possibility of preprocessing depends on the value of p. As the value of p grows, the number of entries in any column whose values are revised increases. So, the higher the value of p, the greater the chance that a row of C is eliminated due to Theorem 2. This explains why PMP instances with $p = p_0$, $p_0 < m/2$, are more difficult to solve than instances on the same costs matrix with $p = m - p_0$, even though the number of feasible solutions for both cases are identical. Let p' be the smallest value of p for which p-truncation eliminates at least one row in C. Let us also denote by p^* the minimum number of rows that contain the minimum entry of each column of C. Then, the PMP instance defined on C with $p > p^*$ has open facilities that do not serve any client and increasing the value of p over p^* does not improve the objective value.

Table 2.2 presents a characterization of benchmark instances introduced in Table 2.1 in terms of p' and p^*. As can be seen from the table, preprocessing becomes possible only for large number of medians as $p' > m/2$ holds for all the considered benchmark instances. One may notice that the benchmark libraries are arranged in order of increasing difficulty: value of p' are getting closer to m. The values of p^* are very close to m for all the considered instances implying that all benchmark libraries contain no degenerate instances and most of the rows can be potentially included into an optimal solution.

2.3.4 Minimality of the pseudo-Boolean representation

In the previous sections we described a number of reductions that are based on the pseudo-Boolean formulation of the PMP and substantially reduce the amount of data that unambiguously describes the instance. However, there emerges a natural question: can one do better by using a different approach? The following lemma gives an answer to this question.

Theorem 3. *The pseudo-Boolean formulation (2.17) of PMP allows the most compact representation of its instance.*

Proof. The intuition is as follows. Take the reduced and truncated pseudo-Boolean polynomial and consider a monomial $\alpha\mathcal{T}$ with a nonzero coefficient that corresponds to an entry of the costs matrix c_{ij} that does not contribute to any optimal solution. This means that there exist a client j that cannot be assigned to location i. There can be several causes for this:

1. For any subset S of p opened locations there always exists a location $i' \in S$ such that $c_{i'j} < c_{ij}$. In this case client j is never served from location i.

2. For client j location i can be replaced by location i', i.e. there exist some location i' such that $c_{ij} = c_{i'j}$. In this case client j can be served from location i'

Table 2.2. Values of p' and p^* for benchmark instances

library	instance	m	p'	p^*
OR	pmed1*	100	90	93
OR	pmed15*	300	180	285
OR	pmed26*	600	452	581
OR	pmed40*	900	644	882
ODM	BN48	42	27	35
ODM	BN1284	1284	653	1211
ODM	BN3773	3773	3385	3742
ODM	BN5535	5535	2179	5503
TSP	rd100	100	97	97
TSP	D657	657	477	653
TSP	fl1400	1400	1177	1395
TSP	pcb3038	3038	3026	3033
RW	rw100	100	90	95
RW	rw200	200	186	193
RW	rw250	250	241	243
RW	rw500	500	489	492
RW	rw1000	1000	978	992

instead of i.

3. For some subset of locations S client j is equivalent to some client j'. (By equivalence of clients with regard to the set of locations S we mean that sorting locations from S by distance from j and j' gives two equal sequences). In this case these two clients can be viewed as one with aggregate serving costs $c_{ij} + c_{ij'}$ for all $i \in S$.

The latter two cases are symmetric: case 2 means that from the point of view of client j locations i and i' are the equally distant, while case 3 means that from the point of view of the set of locations S clients j and j' are equal. In case 1 the coefficient α is set to 0 during the truncation. For the second case we have zero coefficient as the difference $\Delta[., j] = c_{ij} - c_{i'j}$ is zero. Finally, for the third case equivalent clients are eliminated by reduction of similar monomials.

Next, consider the number of coefficients in the truncated and reduced Hammer-Beresnev polynomial $\mathcal{B}_{C,p}(\mathbf{y})$. Suppose, there exist a model with one less coefficient. This implies that some monomial $\alpha_r \prod_{i \in T_r} y_i$ can be deleted from $\mathcal{B}_{C,p}(\mathbf{y})$ to obtain a new polynomial $\mathcal{B}'_{C,p}(\mathbf{y})$. However, it is always possible to select an input matrix C such that

$$f_C(S) = \mathcal{B}_{C,p}(\mathbf{y}^S)$$

and

$$\min\{\mathcal{B}'_{C,p}(\mathbf{y}), \sum_{i=1}^{m} y_i = m - p\} = \min\{\mathcal{B}_{C,p}(\mathbf{y}), \sum_{i=1}^{m} y_i = m - p\} - \alpha_r.$$

Taking into account that $\alpha_r > 0$ (by definition of $\mathcal{B}_{C,p}(\mathbf{y})$), the optimal values of the two formulations are different and we have a contradiction. ■

Theorem 3 has important consequence for applicability of the pseudo-Boolean formulation. Let us consider an arbitrary model of the PMP within the class of mixed-Boolean linear programming (LP) models. The size of a mixed-Boolean LP model is determined by the following four factors:

- number of Boolean variables
- number of continuous variables
- number of constraints (and number of terms in each constraint)
- number of monomials in the objective function

We claim that the minimum mixed-Boolean LP (MBLP) model for PMP can be derived from its pseudo-Boolean representation, as demonstrated in the next section.

2.4 A compact mixed-Boolean LP model

In order to obtain a mixed-Boolean LP model we have linearised all nonlinear terms in 2.16 by introducing additional variables $z_r = \prod_{i \in T_r} y_i$. Since $\alpha_r \geq 0$ and PMP is a minimization problem (2.17) one can replace each nonlinear equality $\prod_{i \in T_r} y_i = z_r$ by an equivalent nonlinear inequality $\prod_{i \in T_r} y_i \leq z_r$ which is equivalent to the following system of linear inequalities: $\sum_{i \in T_r} y_i - |T_r| + 1 \leq z_r$ and $z_r \geq 0$. In any optimal PMP solution the variable z_r is set to 0 if and only if at least one variable $y_i = 0$, and $z_r = 1$ if and only if all $y_i = 1$, i.e. $z_r \in \{0, 1\}$. Now the pseudo-Boolean formulation of PMP with a nonlinear objective function (2.16) can be presented as the following mixed Boolean linear programming model:

$$\text{minimize} \quad \left\{ \alpha_0 + \sum_{r=1}^{m} \alpha_r y_r + \sum_{r=m+1}^{k} \alpha_r z_r \right\} \tag{2.33}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} y_i = m - p; \tag{2.34}$$

$$\sum_{i \in T_r} y_i - |T_r| + 1 \leq z_r, \;\; r = m+1, \dots, k; \tag{2.35}$$

$$z_i \geq 0, \quad i = m+1, \dots, k. \tag{2.36}$$

$$y_i \in \{0, 1\}, \;\; i = 1, \dots, m. \tag{2.37}$$

The objective function (2.33) is split into three parts: the first part α_0 is the sum of all smallest entries δ_{1j} per column (client) j; the second part reflects the penalties incurred by the next to the smallest entries δ_{2j}, and the third part represents all other penalties corresponding to δ_{ij} for $1 < i \leq m - p$.

We call the formulation (2.33)–(2.37) a Mixed Boolean pseudo-Boolean model (MBpBM) for PMP. In case of the example costs matrix defined by (2.18) the MBpBM formulation can be easily derived from (2.22):

$$\min 8 + y_2 + 2y_4 + z_5 + z_6 + z_7 + z_8 \tag{2.38}$$

$$\text{s.t.} \quad z_5 + 1 \geq y_1 + y_3, \tag{2.39}$$

$$z_6 + 1 \geq y_2 + y_3, \tag{2.40}$$

$$z_7 + 1 \geq y_2 + y_4, \tag{2.41}$$

$$z_8 + 1 \geq y_3 + y_4, \tag{2.42}$$

$$\sum_{i=1}^{4} y_i = m - p = 2, \tag{2.43}$$

$$z_i \geq 0, \;\; i = 5, \dots, 8; \tag{2.44}$$

$$y_i \in \{0, 1\}, \;\; i = 1, \dots, 4. \tag{2.45}$$

In the following Lemma 2.1 we explain how to reduce the number of Boolean variables y_i involved in the restrictions (2.35). If for $T_{r_1} \neq \emptyset$ we have that $T_{r_1} \subset T_{r_2}$, then the number of variables corresponding to the inequality with z_{r_2} in (2.35) might be reduced as follows

$$z_{r_1} + \sum_{i \in T_{r_2} \setminus T_{r_1}} y_i - |T_{r_2} \setminus T_{r_1}| + 1 \leq z_{r_2} \tag{2.46}$$

Lemma 2.1. *Let $\emptyset \neq T_{r_1} \subset T_{r_2}$ be a pair of embedded sets of Boolean variables y_i. Thus,* *two following systems of inequalities*

$$\sum_{i \in T_{r_1}} y_i - |T_{r_1}| + 1 \leq z_{r_1} \tag{2.47}$$

$$\sum_{i \in T_{r_2}} y_i - |T_{r_2}| + 1 \leq z_{r_2} \tag{2.48}$$

$$z_{r_1} \geq 0, \quad z_{r_2} \geq 0 \tag{2.49}$$

and

$$\sum_{i \in T_{r_1}} y_i - |T_{r_1}| + 1 \leq z_{r_1} \tag{2.50}$$

$$z_{r_1} + \sum_{i \in T_{r_2} \setminus T_{r_1}} y_i - |T_{r_2} \setminus T_{r_1}| \leq z_{r_2} \tag{2.51}$$

$$z_{r_1} \geq 0, \quad z_{r_2} \geq 0 \tag{2.52}$$

are equivalent.

Proof. Our proof will be done if we show that the following inequalities $\sum_{i \in T_{r_2}} y_i - |T_{r_2}| + 1 \leq z_{r_2}$ and $z_{r_1} + \sum_{i \in T_{r_2} \setminus T_{r_1}} y_i - |T_{r_2} \setminus T_{r_1}| \leq z_{r_2}$ are equivalent. Note that $z_{r_2} = 1$ if and only if $y_i = 1$ for all $i \in T_{r_2}$ which implies that $z_{r_1} = 1$ since $T_{r_1} \subset T_2$, but $z_{r_2} = 0$ if and only if at least one variable $y_i = 0$ for some $i \in T_{r_2}$. If $i \in T_{r_1}$, then $z_{r_1} = 0$ implies $z_{r_2} = 0$, even if $y_i = 1$ for all $i \in T_{r_2} \setminus T_{r_1}$, otherwise $y_i = 0$ for some $i \in T_{r_2} \setminus T_{r_1}$ implies $z_{r_2} = 0$ even if $z_{r_1} = 1$. ■

In the following theorem we indicate that our new problem formulation (2.33)–(2.37) is equivalent to the pseudo-Boolean formulation (2.17).

Theorem 4. *PMP formulations (2.33)–(2.37) and (2.17) are equivalent.*

Proof. The "if" statement is trivial and we start with "only if" part, i.e we are going to show that any feasible solution to (2.33)–(2.37) is feasible to (2.17). Constraints (2.35) ensure that for any subset of opened sites within T_r the corresponding penalties in both objective functions will be zero. Otherwise (if all sites within T_r are closed), the same penalty value will be added to the objective functions of (2.33)–(2.37) and (2.17). ■

It is clear that the restriction (2.35) for z_r can be expressed by means of embedded terms with different degrees such that $T = \left\{ \cup_{t=1}^{k} T_{r_t} \right\} \subseteq T_r$.

Based on the compact representation of a PMP instance within pseudo-Boolean formulation (2.17) one may conclude that this formulation has extracted only essential information to represent the PMP from optimization point of view. In particular, for each client j only sites with p-truncated and pairwise different distances are essential for an optimal PMP solution. These distances form the objective function of our mixed Boolean linear programming formulation (MBpBM) as well as the set of linear constraints. Since each linear constraint (2.35) represents a non-linear monomial in the objective function of pseudo-Boolean formulation (2.17) we have incorporated in the MBpBM the number p of medians as follows: for larger values of p our MBpBM has less non-negative variables and corresponding constraints induced by non-linear monomials. It means that we are in a position to check whether our MBpBM is an optimal model within the class of Mixed Boolean Linear Programming models. If we will be able to show that the matrix of all linear constraints in our MBpBM induced by non-linear monomials contains the smallest number of non-zero entries, then taking into account that the objective function of our MBpBM has the smallest number of non-zero coefficients one may conclude that our MBpBM is an optimal one. Unfortunately, in general it is not the case. It is not difficult to show that the problem of finding a constraint matrix with the smallest number of non-zero entries is at least as hard as the *classic set covering problem* (see, e.g., Garey & Johnson, 1979). Let us consider a partially ordered set with subsets of cardinality at least two corresponding to all non-linear monomials in the truncated and reduced pBp. Take any set F with the largest cardinality. We say that the set F is *covered* by its subsets $F_i \subset F$ if $|F \setminus (\cup_{i \in L} F_i)| \leq 1$ and F is *covered by a single subset* if $|F \setminus F_r)| = 1$. It is clear that the following linear constraint corresponding to a single covering

$$z_{F_r} + y_r - 1 \leq z_F$$

has the smallest number of non-zero entries. In general case even for a single set F to find the best covering by subsets embedded in F is an NP-hard problem (see, e.g., Garey & Johnson, 1979), but our problem is more difficult since we are looking for an optimal covering not only for the set F but also for all its subsets minimizing the number of non-zero entries of all corresponding linear constraints. In other words, we have shown that to find an optimal model within the class of Mixed Boolean Linear Programming models is an NP-hard problem, even if the number of corresponding linear constraints is a linear function on the number of all non-negative decision variables.

Note that the number of non-zero coefficients in the objective function of MBpBM is minimal because the number of non-zero coefficients corresponding to non-linear terms are minimal (just by means of contradiction it can be easily shown that if we assume that there is an objective function with strictly less number of non-zero coefficients then there is either a feasible or an optimal solution to the PMP for which the objective function value defined on the corresponding solution is strictly less than the objective function computed on the given PMP instance). Since the number of linear constraints (2.35) is equal to the number of non-zero coefficients α_r for $r = m+1,\dots,k$ one may conclude that both numbers, namely the number of non-zero coefficients in (2.33) and the number of linear constraints (2.35) are minimal. These considerations are formalized in the following theorem.

Theorem 5. *The numbers of coefficients in the objective, variables and constraints in MBpBM are minimal within the class of mixed-Boolean LP models for PMP.*

Proof. First of all note that the number of Boolean variables cannot be smaller than m as otherwise some potential locations are not taken into account. Next, minimality of the number of coefficients in the objective immediately follows from the minimality of the pseudo-Boolean representation (Theorem 3). This implies minimality of the number of nonnegative variables (corresponding to nonlinear monomials) as the number of Boolean variables is fixed to m and is closely related to the number of linear monomials. This, in turn, implies minimality of the number of constraints as it is exactly the number of nonnegative variables and each nonnegative variable needs at least one constraint to be biased with Boolean variables. ■

Theorem 5 can be illustrated by the following example using the costs matrix (2.18). The MBpBM can be easily derived from the pseudo-Boolean formulation (2.22) and looks like:

$$\begin{aligned}
& f(\mathbf{y},\mathbf{z}) = 8 + y_2 + 2y_4 + z_5 + z_6 + z_7 + z_8 \to \min \\
& \text{s.t.} \\
& y_1 + y_2 + y_3 + y_4 = 2 \\
& z_5 \geq y_1 + y_3 - 1 \\
& z_6 \geq y_2 + y_3 - 1 \\
& z_7 \geq y_2 + y_4 - 1 \\
& z_8 \geq y_3 + y_4 - 1 \\
& y_i \in \{0,1\},\ i = 1,\dots,4 \\
& z_i \geq 0,\ i = 5,\dots,8
\end{aligned} \tag{2.53}$$

The obtained model has 4 Boolean y-variables, 4 nonnegative z-variables, 5 constraints and 6 terms in the objective function. For Elloumi's Mixed Integer Linear Programming (MILP) model (Elloumi, 2010) there numbers are 4, 17, 23 and 12, respectively.

Properties of our model, such as decreasing number of variables and constraints, give some insight into the properties of the polytope of feasible solutions to the PMP. The fact that the total number of variables in MBpBM is always less then $(m-p)\cdot n$ implies that the p-Median polytope never has a full dimension of $m\cdot n$, i.e. some dimensions are either fixed (by p-truncation) or are duplicate (these are removed by combining similar monomials in the Hammer-Beresnev polynomial). At the same time MBpBM allows measuring the actual dimension of the polytope and implies that this dimension decreases with increasing p as more and more dimensions become fixed (more and more assignments of clients to facilities become prohibited).

2.4.1 Further reductions

Even though MBpBM is a very compact model, we can further reduce it by involving upper and lower bounds on the cost of optimal solutions (see Goldengorin et al., 2003). Suppose, we know from some heuristic a (global) upper bound f^{UB} on the cost of optimal solutions. This can be even a virtual upper bound, i.e. without a feasible solution. Let us now consider some term $\mathcal{T}_r = \sum_{i\in T_r} y_i$ from the pseudo-Boolean polynomial and define a vector $\mathbf{y}^r$ in the following way: for any $i \in T_r$ set $y_i^r = 1$ and set all other elements of $\mathbf{y}^r$ to zero. It is easy to see that $\mathcal{B}_{C,p}(\mathbf{y}^r)$ is a valid lower bound for the subspace of solutions with all locations from T_r closed. If we denote this lower bound by f_r^{LB} then the following holds

$$f_r^{LB} = \mathcal{B}_{C,p}(\mathbf{y}^r) \tag{2.54}$$

The essence of the reduction that we consider here can be expressed by the following lemma.

Lemma 2.2. *If for some term $\mathcal{T}_r = \prod_{i\in T_r} y_i$ of a truncated and reduced Hammer-Beresnev polynomial holds $f_r^{LB} > f^{UB}$ then for every optimal solution $\mathbf{y}^*$ holds $\mathcal{T}_r(\mathbf{y}^*) = 0$.*

Proof. Taking into account that we have a truncated polynomial, any term r has a degree of at most $m-p$ and $|T_r| \le m-p$. This means that to keep $\mathcal{T}_r$ equal to 1 at least p sites are to be opened, implying that the cost any feasible solution with sites from T_r closed is at least $\mathcal{B}_{C,p}(\mathbf{y}^r)$ (by closing additional sites we can only increase the objective value). By condition of the lemma we have that there exist a cheaper feasible solution, thus any feasible solution $\mathbf{y}$ with $y_i = 1$ for any $i \in T_r$ is not optimal. ■

The following counter-example shows that the inequality in Lemma 2.2 should be strict.

Example. Consider an instance defined by the following costs matrix C:

$$C = \begin{bmatrix} 0 & 6 & 6 \\ 1 & 0 & 8 \\ 2 & 9 & 9 \\ 5 & 4 & 0 \end{bmatrix} \tag{2.55}$$

A possible permutation and differences matrices are

$$\Pi = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 4 & 1 \\ 3 & 1 & 2 \\ 4 & 3 & 3 \end{bmatrix} \quad \Delta = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 4 & 6 \\ 1 & 2 & 2 \\ 3 & 3 & 1 \end{bmatrix} \tag{2.56}$$

In case $p = 2$ the Hammer-Beresnev polynomial is

$$\mathcal{B}_{C,p=2}(\mathbf{y}) = 1y_1 + 4y_2 + 6y_4 + 1y_1y_2 + 3y_1y_4 + 2y_2y_4 \tag{2.57}$$

Considering the first term $\mathcal{T}_1 = y_1$, we have $T_1 = \{1\}$, $\mathbf{y}^1 = (1,0,0,0)$ and $f_1^{LB} = \mathcal{B}_{C,p=2}(\mathbf{y}^1) = 1$. Suppose, the upper bound is the same $f^{UB} = 1$. If the inequality in Lemma 2.2 was non-strict then this would imply that for any optimal solution $y_1 = 0$. However, it can be checked that for the unique optimal solution to this instance $\mathbf{y}^* = (1,0,1,0)$ this does not hold. Note that $y_3^* = 1$ due to Theorem 2 ◁

Now we can check the condition of Lemma 2.2 for every term of the pseudo-Boolean polynomial, starting from terms with the lowest degree. Such order of checking terms is beneficial because if some term is found to be zero in any optimal solution (let us call such terms *null terms*) then all terms of higher degree containing it are also zero. Here we would like to point out two possibilities of dealing with null terms within our MBpBM model.

The first and the most straightforward approach is to set the variable that corresponds to a null term to zero throughout the formulation. This eliminates one term from the objective function, but preserves the number of constraints. At the same time the number of non-zero entries in the constraints matrix can increase as elimination of some term (or corresponding z-variable) reduces the possibility of applying Lemma 2.1. We call the model reduced according to this approach based on bounds MBpBMb. The following example shows how this reduction works.

Consider the costs matrix C (2.18), $p = 2$ and an MBpBM model (2.53). One can compute the global upper bound f^{UB}, for example, by greedy heuristics that works as follows. It starts with all locations opened (i.e. $\mathbf{y} = (0,0,0,0)$) and at each step closes such location (sets such y_i to 1) that results in the smallest increase in the value of the objective function. The procedure is repeated until $m - p$ locations are closed ($m - p$ entries of $\mathbf{y}$ are set to 1). For the costs matrix given by (2.18) this procedure gives $f^{UB} = 9$. Then, for every term $\mathcal{T}_r$ of the objective function we construct a vector $\mathbf{y}^r$ and compute the lower bound to the unknown optimal value $\mathcal{B}_{C,p=2}(\mathbf{y}^r)$:

$$\begin{array}{llrl}
\mathcal{T}_1 = y_2 & \mathbf{y}^1 = (0,1,0,0) & \mathcal{B}_{C,p=2}(\mathbf{y}^1) = \ 9 & \\
\mathcal{T}_2 = y_4 & \mathbf{y}^2 = (0,0,0,1) & \mathcal{B}_{C,p=2}(\mathbf{y}^2) = \ 10 & > f^{UB} \\
\mathcal{T}_3 = z_5 = y_1y_3 & \mathbf{y}^3 = (1,0,1,0) & \mathcal{B}_{C,p=2}(\mathbf{y}^3) = \ 9 & \\
\mathcal{T}_4 = z_6 = y_2y_3 & \mathbf{y}^4 = (0,1,1,0) & \mathcal{B}_{C,p=2}(\mathbf{y}^4) = \ 10 & > f^{UB} \\
\mathcal{T}_5 = z_7 = y_2y_4 & \mathbf{y}^5 = (0,1,0,1) & \mathcal{B}_{C,p=2}(\mathbf{y}^5) = \ 12 & > f^{UB} \\
\mathcal{T}_6 = z_8 = y_3y_4 & \mathbf{y}^6 = (0,0,1,1) & \mathcal{B}_{C,p=2}(\mathbf{y}^6) = \ 11 & > f^{UB}
\end{array} \tag{2.58}$$

By comparing the obtained values with the computed upper bound we have that in any optimal solution $\mathcal{T}_2$, $\mathcal{T}_4$, $\mathcal{T}_5$ and $\mathcal{T}_6$ are zero, i.e. in our MBLP model we can fix variables y_4, z_6, z_7 and z_8 to zero:

$$8 + y_2 + z_5 + 0z_6 + 0z_7 + 0z_8 \to \min \tag{2.59}$$

$$\text{s.t.} \quad y_1 + y_2 + y_3 = 2, \tag{2.60}$$

$$z_5 + 1 \geq y_1 + y_3, \tag{2.61}$$

$$z_6 + 1 \geq y_2 + y_3, \tag{2.62}$$

$$0 + 1 \geq y_2 + 0, \tag{2.63}$$

$$0 + 1 \geq y_3 + 0, \tag{2.64}$$

$$y_4 = 0 \tag{2.65}$$

$$z_i \geq 0, \ i = 5, \ldots, 8; \tag{2.66}$$

$$y_i \in \{0,1\},\ i = 1,\dots,4. \tag{2.67}$$

By substituting the fixed values into all constraints and the objective function and observing that some constraints (2.63) and (2.64) became redundant we obtain the reduced model:

$$8 + y_2 + z_5 \to \min \tag{2.68}$$

$$\text{s.t.} \quad z_5 + 1 \geq y_1 + y_3, \tag{2.69}$$

$$1 \geq y_2 + y_3, \tag{2.70}$$

$$y_1 + y_2 + y_3 = 2, \tag{2.71}$$

$$y_4 = 0, \tag{2.72}$$

$$z_5 \geq 0; \tag{2.73}$$

$$y_i \in \{0,1\},\ i = 1,\dots,4. \tag{2.74}$$

Further we will call this variation of MBpBM with reduction based on bounds – MBpBMb.

Another approach to dealing with null terms allows not only to reduce the size of the objective function, but also the number of constraints. Its essence is expressed by the following lemmas.

Lemma 2.3. *Increasing a coefficient at a null term does not affect the cost of optimal solutions.*

Proof. Straightforward from the definition of null terms. If some term is found to be equal to 0 in any optimal solution then increasing the coefficient of the corresponding monomial will not result in new optimal solutions. ■

Lemma 2.4. *If for some term r_1 in the Hammer-Beresnev polynomial there exist an embedded term r_0 with large enough coefficient α_{r_0}, then r_1 can be given a zero coefficient without affecting the cost of optimal solutions.*

Proof. The cost of any feasible solutions for which term r_0 evaluates to 1 is bounded from below by α_{r_0}. If there exist a feasible solution of cost less than α_{r_0}, all terms containing r_0 evaluate to 0 in any optimal solution. This, in turn, implies that their coefficients are irrelevant and can be set to 0. ■

Thus, if a null term is found, it can be given a large enough coefficient (exceeding a cost of an arbitrary feasible solution) and all terms for which it is embedded can be eliminated from the Hammer-Beresnev polynomial without a need in additional constraints. We call the model with this reduction MBpBMb1. Even though it allows smaller reduction of the objective function, it can benefit from the reduced number of constraints. For the considered above instance with costs matrix C (2.18) and $p = 2$ the MBpBMb1 model is as follows:

$$8 + y_2 + 1000y_4 + z_5 + 1000z_6 \rightarrow \min \tag{2.75}$$

$$\text{s.t.} \quad z_5 + 1 \geq y_1 + y_3, \tag{2.76}$$

$$z_6 + 1 \geq y_2 + y_3, \tag{2.77}$$

$$y_1 + y_2 + y_3 + y_4 = 2, \tag{2.78}$$

$$y_4 = 0, \tag{2.79}$$

$$z_i \geq 0, \; i = 5, \ldots, 8; \tag{2.80}$$

$$y_i \in \{0, 1\}, \; i = 1, \ldots, 4. \tag{2.81}$$

Here we have set large coefficients to 1000 for the sake of clarity, while in practice the value of $f^{UB} + 1 = 10$ suffices for this purpose. It is also clear that if we find a linear null term then we fix the corresponding y-variable to 0 and eliminate this term from the objective function (instead of raising its coefficient to infinity). Moreover, if for some null term there are no terms into which it is embedded, then it is beneficial to eliminate it and add a corresponding constraint. If these exceptions are introduced into MBpBMb1 then for the considered example both formulations (MBpBMb and MBpBMb1) become equal, although the mechanisms by which some constraints were dropped are different. While in MBpBMb third and fourth constraints became redundant as most of the variables in them were set to 0, in MBpBMb1 these constraints did not exist at all because corresponding terms were eliminated from the pseudo-Boolean polynomial.

Finally, it should be mentioned that instead of $f_r^{LB} = \mathcal{B}_{C,p}(\mathbf{y}^r)$ a somewhat stronger lower bound can be used.

Lemma 2.5. *ϕ_r^{LB} defined as*

$$\phi_r^{LB} = f_C(\overline{T}_r) + \min_{k_i \in \overline{T}_r} \sum_{i=1}^{|\overline{T}_r| - p} [f_C(\overline{T}_r \setminus \{k_i\}) - f_C(\overline{T}_r)] \tag{2.82}$$

is a valid lower bound for the subspace of feasible solutions having locations from T_r closed, where $f_C(.)$ – cost function of the PMP, i.e. $f_C(\overline{T}_r) = \mathcal{B}_{C,p}(\mathbf{y}^r)$, and $\overline{T}_r$ denotes the complement of T_r, i.e. $\overline{T}_r = \{1, \ldots, m\} \setminus T_r$.

Proof. As the cost function of the PMP $f_C(.)$ is a supermodular function (Goldengorin et al., 2003), the following holds for any $S \subseteq T \subseteq \{1, \ldots, m\}$

$$f_C(S) \geq f_C(T) + \sum_{k \in T \setminus S} [f_C(T \setminus \{k\}) - f_C(T)] \tag{2.83}$$

In our case S is unknown restricted ($S \subseteq T$ must hold) optimal solution (set of opened locations) of cardinality $|S| = p$ implying that

$$f_C(S) \geq f_C(T) + \min_{k_i \in T} \sum_{i=1}^{|T|-p} [f_C(T \setminus \{k_i\}) - f_C(T)] \tag{2.84}$$

where $\overline{T} = \{1, \ldots, m\} \setminus T$. If we now set $T = \overline{T}_r$ then the proof is completed. ■

In the computational experiments reported in the following sections (and involving MBpBM and its modifications) we used lower bounds given by (2.82).

2.4.2 Computational experiments

In order to show the applicability of our compact MBpBM formulation, a number of computational experiments were held. We used benchmark instances from two of the most widely used libraries: J. Beasley's OR-library and randomly generated RW instances by Resende and Werneck (see, e.g., Elloumi, 2010). The common class of benchmark instances included in almost all publications devoted to the PMP itself is just the OR-Library instances. Since the main purpose of our experiments is to show that our model for PMP is one of the best currently known models (see Church (2008) and Elloumi (2010)) which could be used to solve PMP instances to optimality based on general-purpose software, we have used Xpress-MP and 15 largest instances (see Tables 2.3 and 2.5) from OR-Library (OR Library, 1990). This problem library contains 40 different PMP instances, each representing a graph of n vertices, each being a client and a potential facility, and a specific value of p. Graphs are complete and range in size from 100 (with 10000 arcs) to 900 (with 810000 arcs) nodes. The distance c_{ij} between two nodes is the length of a shortest path linking them.

We have conducted our experiments on a Personal Computer with Intel 2.33 GHz processor 1.95 GB RAM and Xpress-MP[1] as an MILP solver.

[1] This choice of the solver is governed purely by the availability of software at my working place. CPLEX might provide better results.

Tables 2.3 and 2.4 summarize the computational results obtained for the largest 15 OR instances and random RW instances, correspondingly. The first three columns contain the name of instance, the number of m nodes and the number p of medians. The next three columns are related to the running times (in seconds) for the considered above variations of our model: the initial MBpBM formulation and its modifications incorporating reductions based on bounds. The last column reflects computing times for Elloumi's NF model that we implemented and tested within the same environment as our models so that to ensure consistent comparison of performance.

Table 2.3. Comparison of computing times for our and Elloumi's NF formulations (15 largest OR-library instances)

instance	m	p	MBpBM	MBpBMb	MBpBMb1	Elloumi
pmed26	600	5	163.84	194.08	111.81	180.31
pmed27	600	10	27.59	41.00	21.31	43.73
pmed28	600	60	2.48	8.63	2.13	3.61
pmed29	600	120	1.78	6.50	1.31	2.91
pmed30	600	200	1.50	5.56	0.78	4.81
pmed31	700	5	153.22	132.91	57.05	90.95
pmed32	700	10	33.13	53.17	43.39	37.64
pmed33	700	70	3.09	10.11	2.69	4.73
pmed34	700	140	3.72	8.03	1.97	7.11
pmed35	800	5	70.30	233.66	154.41	514.72
pmed36	800	10	2256.83	2014.70	4252.13	6737.25
pmed37	800	80	3.91	12.61	3.08	7.00
pmed38	900	5	1328.34	368.73	2041.28	307.00
pmed39	900	10	572.81	713.59	444.08	473.95
pmed40	900	90	5.39	15.53	4.02	8.42

Our computational experiments with OR and RW instances can be summarized as follows. Our basic MBpBM formulation outperforms Elloumi's New Formulation in most of the tested cases, especially for larger numbers of medians p. At the same time the reduction based on bounds (see column MBpBMb) has comparatively poor performance in general. This can be explained by an increased number of non-zero coefficients in a constraints matrix (for large RW instances this formulation cannot be handled by Xpress-MP due to memory limitations). However, there exist instances (e.g., pmed36 and pmed38) for which MBpBMb performs better then other variations of the formulation based on a pseudo-Boolean polynomial and for the instance pmed36 has three times smaller computing times comparatively to NF. Better performance of larger models can be explained by the fact that increased number of variables and coefficients provides more options for branching and thus may lead to better pivoting of the MILP solution procedure. Finally, the revised reduction based on bounds MBpBMb1 outperformed other considered models in almost all cases except pmed32, pmed36, pmed38 (for pmed36 it is better than NF but is worse than MBpBM). We would also like to mention one instance from TSP library (TSP Library, 1995), namely fl1400, with $p = 400$ which is unsolved in Avella et al. (2007) and has been solved to optimality by our MBpBM in 598.5 sec. Note that Beltran's et al. (Beltran et al., 2006) advanced semi-Lagrangean approach based on Proximal-Analytic Center Cutting Plane Method has not solved the instance fl1400 with $p = 400$ to optimality as well and returns an approximation within 0.11% in 678 sec.

2.5 Application of the pseudo-Boolean approach: Instance data complexity

2.5.1 Data complexity and problem size reduction

Problem size reduction is a very common technique in integer programming and combinatorial optimization that can be used to find a compact representation of PMP instances. It is aimed at constructing an instance of a smaller size that is assumed to be easier to solve and provides an optimal solution to the initial instance. Moreover, it is straightforward that if the procedure of size reduction is as time-consuming as the procedure for solving the initial problem, it has no sense. These considerations lead to the following definition:

Definition 3. *We will call instance $\mathcal{D}$ a* reduced version *of instance $\mathcal{C}$ ($\mathcal{D} = red(\mathcal{C})$) if it satisfies the following conditions:*

1. $\emptyset \subset opt.solutions(\mathcal{D}) \subseteq opt.solutions(\mathcal{C})$;
2. $size(\mathcal{D}) \leq size(\mathcal{C})$;
3. *$\mathcal{D}$ can be obtained from $\mathcal{C}$ in polynomial time.*

The first requirement guarantees that by solving $\mathcal{D}$ to optimality one immediately obtains an optimal solution to $\mathcal{C}$ (here we assume the feasibility of $\mathcal{C}$), while the second one is related to the reduction itself. Finally, the last requirement is needed to make the definition useful in practice: if for some NP-hard problem computing the reduced instance $\mathcal{D}$ is as hard as solving $\mathcal{C}$ then such a reduction is senseless. Based on this definition of a reduced instance we define complexity of the instance data in the following way:

Definition 4. *By* complexity *of the instance data $\mathcal{C}$ (relative to a particular problem) we mean the minimum capacity of the storage needed to be able to obtain an optimal solution to the initial instance:*
$comp(\mathcal{C}) = \min\{size(\mathcal{D}) : \mathcal{D} = red(\mathcal{C})\}.$

It should be noticed that without a reference to a particular problem (in our case – the p-Median problem) this definition is meaningless. However, even when the problem is fixed, it provides neither a direct way to constructing a compact representation of the data, nor even for determining the minimum required space. Further we briefly describe existing approaches to reducing the problem size and thus to obtaining upper bounds of instance data complexity.

As the costs matrix of a PMP instance has $m \times n$ elements, it is clear that this value is the most trivial upper bound for $comp(\mathcal{C})$. This value is achieved by the classical ILP representation (see ReVelle & Swain, 1970) of the p-Median problem with its objective function defined as:

$$\sum_{j \in J} \sum_{i \in I} c_{ij} x_{ij} \tag{2.85}$$

Here c_{ij} denote entries of the costs matrix and x_{ij} are decision variables ($x_{ij} = 1$ if jth client is served from ith location, otherwise $x_{ij} = 0$). Cornuejols *et al.* Cornuejols et al. (1980) introduced an alternative formulation of the problem. For any client j, let K_j be the number of different distances from j to any location. It follows that $K_j \leq m$. Let $D_j^1 < D_j^2 < \ldots < D_j^{K_j}$ be these distances, sorted. For each client j it is possible to define a hierarchy of neighbourhoods V_j^k such that each V_j^k is a set of locations within the distance D_j^k from client j. Naturally, in an optimal solution a client j is assigned to its neighbourhood with the smallest D_j^k containing the opened location. Thus, instead of x_{ij} this formulation uses variables z_j^k such that $z_j^k = 1$ if and only if there are no opened locations in V_j^k. The objective function in this case is defined as:

$$\sum_{j \in J} \left(D_j^1 + \sum_{k=1}^{K_i - 1} (D_i^{k+1} - D_i^k) z_i^k \right) \tag{2.86}$$

Informally, this representation implies that only different elements in each column of the costs matrix are meaningful and the problem size can be reduced by storing only the pairwise different elements from each column. The further reduction is proposed in Elloumi (2010). It states that if for some j, k, j', k' holds $V_j^k = V_{j'}^{k'}$, then for any feasible solution $z_j^k = z_{j'}^{k'}$ and some terms in (2.86) can be merged. Several reductions are also presented in Church (2003), but they are similar to those described above. There are also some papers aimed at reduction of the number of constraints in the ILP formulation of the problem (see, e.g., Avella & Sforza, 1999; Avella & Sassano, 2001); however, the number of coefficients in the objective function remains the same.

It should be noticed that most of the reduction techniques described in literature are based on ILP formulation of the p-Median problem and apply artificial tricks exploiting some features of the instance. On the contrary, as we showed in Section 2.3, a pseudo-Boolean representation itself naturally leads to several reductions that allow obtaining better estimates of the instance data complexity and include all known reductions. (Note that due to the fact that the construction of the pBp and all its reductions can be done in polynomial time, the third condition of Definition 3 is satisfied.)

2.5.2 Complex benchmark instances

In this section we consider the aspects of constructing complex benchmark instances that can be used for testing solution algorithms and introduce our library of such instances.

To have maximum possible complexity, a PMP instance defined on an $m \times n$ costs matrix should not be amenable to any of the reductions described above. Thus, first of all, the entries of the differences matrix should be non-zero, such that all monomials in the pBp have non-zero coefficients, or, equivalently:

Claim 1. *The most complex instances have pairwise different and nonzero entries in every column of the costs matrix (assuming that sizes of the costs matrix are fixed).*

However, as explained below, these two restrictions on the entries of the differences matrix are not sufficient to ensure the complexity. Suppose, for some column j the difference between the minimal and second minimal element $\Delta c[1, j]$ is comparable to the (unknown) costs of optimal solution. In this case the location (row), at which the minimum for jth client (column) is attained, will be included into any optimal solution. Such additional structure can be exploited by the solution algorithms and thus reduces the complexity of the instance. This particular case can be generalized in the following way. Suppose, the (truncated) pseudo-Boolean polynomial contains a monomial $\alpha\mathcal{T} = \alpha \prod_{i\in T} y_i$ with a large enough coefficient α that exceeds the costs of the optimal solution (or its somehow computed upper bound). Then, clearly, for any optimal solution $\mathbf{y}$ holds $\mathcal{T}(\mathbf{y}) = 0$, implying that at least one of the variables in $\mathcal{T}$ must be set to zero and at least one of the corresponding locations is opened. In fact, this condition can be made even stronger if one considers not only the coefficient α at $\mathcal{T}$, but the sum of α and coefficients of all monomials with terms embedded in $\mathcal{T}$. It is also quite straightforward that this test is more likely to fail as the range of the entries of differences matrix (or, equivalently, coefficients of the pseudo-Boolean polynomial) becomes smaller, up to the limit case when they all are equal. These considerations lead to the following claim.

Claim 2. *Instances that lead to the pseudo-Boolean polynomial with all coefficients equal (except a constant - monomial of degree zero) are the most complex ones (assuming that the number of monomials is fixed).*

Once we know how to construct a “complex” pseudo-Boolean polynomial, we are interested in maximizing the number of monomials in it. To achieve this, there should be no similar monomials in the pBp representation of the problem. It should be mentioned that constants obtained from pseudo-Boolean representation of all the columns can be reduced into one monomial, so every PMP instance has a complexity of at most

$$comp(\mathcal{C}) \leq mn - (n-1) = n(m-1) + 1. \tag{2.87}$$

To ensure that only constants can be aggregated, the permutation matrix Π must conform with the following requirement: the sets of first k entries of columns Π^j in Π should be pairwise different for any $k : 1 \leq k \leq m$. This requirement can be expressed in an alternative form. Let us consider a Hasse diagram defined over the subsets of $\{1, \ldots, m\}$. It is easy to see that each permutation $\Pi^j = (\pi_{1j}, \pi_{2j}, \ldots, \pi_{mj})^{\mathrm{T}}$ corresponds to a chain of embedded subsets $\{\pi_{1j}\} \subset \{\pi_{1j}, \pi_{2j}\} \subset \ldots \subset \{\pi_{1j}, \ldots, \pi_{mj}\}$ that, in turn, corresponds to a $\emptyset - \{1, \ldots, m\}$ path in the Hasse diagram. Now the requirement can be formulated as follows:

Claim 3. *In order to prohibit reduction of similar monomials, the permutation matrix should correspond to a collection of internally vertex-disjoint $\emptyset - \{1, \ldots, m\}$ paths in the Hasse diagram defined on subsets of $\{1, \ldots, m\}$.*

Taking into account that there are at most m such paths, for PMP instances with $n > m$ it is always possible to reduce at least $n - m$ linear monomials in the pBp, so for instances in our benchmark library holds $n \leq m$. Due to these considerations it is possible to formulate the problem of constructing a permutation matrix that leads to a complex instance as a problem of finding n vertex disjoint paths in a graph obtained from the Hasse diagram. Though this problem is known to be polynomially solvable (Robertson & Seymour, 1995), the fact that the complete Hasse diagram has 2^m vertices makes the procedure very time consuming for large m. However, there exists a trivial solution:

$$\pi_{ij} = (i + j) \bmod m + 1. \tag{2.88}$$

In case $n = 4$, $m = 5$ this solution leads to the following permutation matrix Π:

$$\Pi = \begin{bmatrix} 3 & 4 & 5 & 1 \\ 4 & 5 & 1 & 2 \\ 5 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \end{bmatrix}$$

Based on a representation of the monomials as a collection of chains it is possible to estimate the complexity of a PMP instance (the maximum number of monomials in the pBp). Consider a complete Hasse diagram that contains all subsets of $\{1, \dots, m\}$. Clearly, the maximum length of a chain of embedded non-constant terms is $m - 1$, as it is the maximum possible degree of the pBp. The number of chains of this maximum length is exactly $m = \binom{m}{1}$ as there exit m linear terms (as well as m terms of degree $m - 1$). Each of these chains uses exactly one term of each degree from 1 to $m - 1$. Once all maximum length chains are used, the next available maximum length of a chain is $m - 3$ (terms of degree $2, \dots, m - 2$). The number of such chains is $\binom{m}{2} - \binom{m}{1}$ which is exactly the number of quadratic terms that were not included in chains of length $m - 1$. If we have enough columns to use all these chains (i.e. n is sufficiently large) then we switch to chains of length $m - 5$ (terms of degree $3, \dots, m - 3$) and there are $\binom{m}{3} - [\binom{m}{2} - \binom{m}{1}] - \binom{m}{1} = \binom{m}{3} - \binom{m}{2}$ such chains, which is exactly the number of cubic terms not used by chains of lengths $m - 2$ and $m - 1$. We continue picking the longest possible chains until we have n of them. It is not hard to understand that the number of terms of some degree k ($1 \leq k \leq m - 1$) in such a collection of n longest chains is bounded by n and, at the same time, cannot exceed $\binom{m}{k}$. Figure 2.3 gives a graphical representation of how the number of terms in such a collection of chains of maximum length can be calculated.

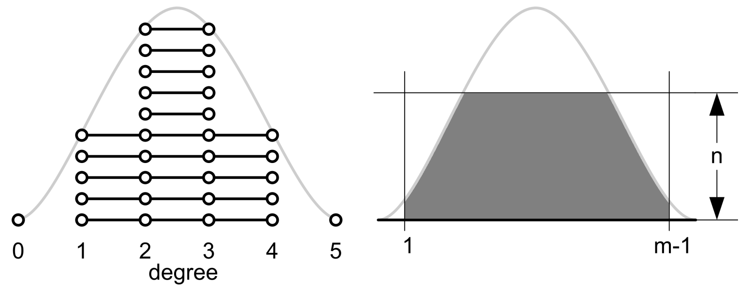


Figure 2.3. Estimating the maximum number of nonzero terms in a pBp.

In the left part of Figure 2.3 an example for $m = 5$ is shown. Circles denote terms of a pBp that are arranged in such a way that terms of same degree are within one column. Lines correspond to possible chains of embedded terms. If one is aimed at having n chains containing the maximum number of terms then he picks n longest chains starting from the lower part of the picture. In particular, it can be seen that it is impossible to get more than 5 full chains for the given example. For instance, if $n = 6$ then at least one linear monomial will be reduced. For arbitrary m and n the maximum number of monomials in the reduced pBp corresponds to the area of the shaded region in the right part of Figure 2.3. Thus the complexity (equivalently, the number of monomials in the corresppondingpBp) of a PMP instance $\mathcal{C}$ defined by an $m \times n$ costs matrix is bounded by

$$comp(\mathcal{C}) \leq \sum_{i=1}^{m-1} \min\{n, \binom{m}{i}\} + 1. \tag{2.89}$$

The main peculiarity is that for a number of clients n exceeding the number of locations m addition of new clients has progressively smaller impact on the complexity of the instance that is always less than $n(m-1)+1$, while for $n \leq m$ there exist instances of complexity $n(m-1)+1$.

It is easy to see that in case $n < m$ all the minima are contained in at most n rows (i.e. all minima are achieved on at most $n < m$ locations) and preprocessing will eliminate at least $m - n$ variables (rows of the costs matrix). This leads to a conclusion that instances with $n = m$ are, potentially, the most complex ones (provided the entries of the costs matrix satisfy the considered above requirements).

Possibility of truncation of the pseudo-Boolean polynomial depends only on the number of medians and is not affected by the values of the costs matrix. Thus, we cannot negate this reduction by adjustment of the costs matrix and if p is fixed (2.89) can be improved in the following way:

$$comp(\mathcal{C}) \leq \sum_{i=1}^{m-p} \min\{n, \binom{m}{i}\} + 1. \tag{2.90}$$

Our benchmark library contains complex (in terms of possibility of problem size reduction) PMP instances defined on square matrices of different size. As costs matrices are dense, they are stored explicitly in files named "XmatrY, Z.txt", where X is 't' if the permutation matrix Π is defined by (2.88) and X is 'r' if Π is a randomized permutation matrix obtained as a solution to the disjoint paths problem mentioned above. Y reflects the values of m and n (in our instances $n = m$), and costs are selected in such a way that entries of the differences matrix Δ are uniformly distributed random integers from $\{1, \ldots, Z\}$. Due to Claim 2 instances with smaller Z are harder to solve. For example, the file named "tmatr4,1.txt" defines the following instance:

$$C = \begin{bmatrix} 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{bmatrix} \tag{2.91}$$

It is easy to check that the permutation matrix is the same as costs matrix C and the difference matrix has all unit entries.

The structure of the files is as follows. The first line contains the numbers of clients and potential locations (columns and rows of the costs matrix), next all entries of the costs matrix are explicitly listed row by row. The library is available at *http://www.hse.ru/en/org/persons/22927115* or *http://go.to/dkrush* (under 'Science'-¿'p-Median problem').

We would like to finish description of our complex instances by mentioning that under certain conditions optimal objective values can be computed by a simple formula (see Lemma 2.6 below). This property is especially useful for the developers of heuristic methods and makes it possible to estimate the quality of the generated solutions.

Lemma 2.6. *If the $m \times n$ costs matrix of a PMP instance satisfies the following conditions:*

(i) $m = n$,

(ii) a permutation matrix is defined by (2.88),

(iii) all entries of the differences matrix are equal to some constant d,

then the optimal objective value can be computed as:

$$d(n'+1)\left[\frac{n'p}{2}+(n\ mod\ p)\right], \tag{2.92}$$

where $n' = \lfloor n/p \rfloor$.

Proof. Conditions of the lemma ensure that each row (and each column) of the costs matrix can be obtained from $(d, 2d, \ldots, md)$ by a cyclic shift, i.e. each multiple of d is contained in each row exactly once. This implies that at most p clients can be served at a cost d. As well, at most p clients can be served at costs $2d$, $3d$, etc. Thus, the minimum can be obtained by serving first p clients at cost d, the next $\min\{n-p, p\}$ clients at cost $2d$, the next $\min\{n-2p, p\}$ clients at a cost $3d$, etc., until we serve all n clients. By a simple combinatorial reasoning, the total costs in this case can be computed as

$$d\left[\frac{n'(n'+1)}{2}p+(n'+1)(n \bmod p)\right] = d(n'+1)\left[\frac{n'p}{2}+(n \bmod p)\right] \tag{2.93}$$

This minimum is achieved by the p locations (rows of the costs matrix) that fall within the following pattern:

$$\begin{array}{cccccccccccc}
(d & 2d & \ldots & pd & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & md) \\
(\ldots & \ldots & \ldots & md & d & 2d & \ldots & pd & \ldots & \ldots & \ldots) \\
(\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & md & d & 2d & \ldots) \\
\ldots
\end{array} \tag{2.94}$$

■

Lemma 2.6 and its constructive proof has an important corollary. It can be checked that in the instances satisfying the condition of the lemma each location is open in p optimal solutions and the number of optimal solutions is n (does not depend on the number of medians p). This means that these instances are degenerate and may be easily solvable, irrespectively of their size.

In order to check the properties of our instances we held a number of computational experiments. For the sake of comparison we used two formulations of PMP: our MBpBM and Elloumi's NF (Elloumi, 2010) (which is the most compact MILP

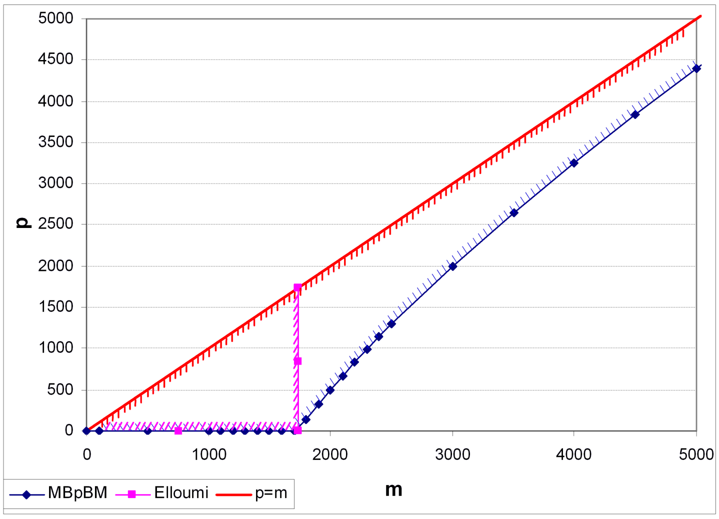


Figure 2.4. Ranges of complex instances data size for which our MBpBM and Elloumi's NF can be loaded by Xpress.

formulation of PMP, to the best of our knowledge). Figure 2.4 shows the ranges of m and p for which the model can be loaded into the MILP solver (in our case Xpress). For different sizes of the $m \times m$ input matrix we checked for which range of p the formulation can be loaded into the MILP solver (i.e., is small enough to fit into the memory). Clearly, this range is bounded from above by the line $p = m$. As Elloumi's formulation does not account for the number of medians, there exist some critical size of the cost matrix beyond which the formulation becomes prohibitively large, irrespectively of p. At the same time, our formulation based on the pseudo-Boolean representation of the instance data can be loaded by a general-purpose MILP solver for some values of p even if the input matrix is of huge dimension (see Figure 2.4).

Finally, we compared running times of two solution approaches (our MBpBM and Elloumi's NF) applied to selected OR instances and to our generated instances of the same size and with the same number of medians p. As was presumed, instances with permutations given by (2.88) are easy for the MILP solver and the running times are of the same magnitude as running times for OR instances (even though the number of coefficients in the formulation is much larger). Thus, we compared running times of OR instances and our complex instances with randomized permutation matrices and differences matrices containing all unit entries. The results of this comparison are presented in Table 2.6 and show that our benchmark instances are complex also in terms of running time. In particular, for small values of p computation times explode even for 100×100 input matrices. Also, for the unsolved instances we compared the best found integer solutions with solutions obtained by heuristics and it was observed that heuristics produced better solutions. This contradicts the common observation that MILP solvers based on branch-and-bound procedures spend only a very small portion of the total running time on finding the optimal solution (while most of the time is spent on proving its optimality). Thus, from this point of view, our instances with randomized permutation matrices are also complex.

2.6 Application of the pseudo-Boolean approach: Equivalent PMP instances

Generally speaking, there exist many different PMP instances that have the same (reduced) Hammer-Beresnev polynomial, mainly because similar monomials can be aggregated and disaggregated. Correspondingly, if two PMP instances have the same size (the same number of potential locations) and the same Hammer-Beresnev polynomial, then any solution $\mathbf{y}$ has the same objective value for both of them. This implies that a solution that is optimal for one of the instances is also optimal for the other one (provided the number of medians p are the same for both instances). This allows to define a notion of equivalence based on a pseudo-Boolean representation. Two instances of the p-Median problem defined by cost matrices C and D are called *equivalent* if they have the same size, the same number of medians p and $\mathcal{B}_C = \mathcal{B}_D$. The most important point here is that equivalence of two instances can be checked in time that is polynomially bounded in the size of the input costs matrix, even though the PMP is itself NP-hard. This becomes clear if one observes that a Hammer-Beresnev representation can be generated in polynomial time and contains a polynomially bounded number of monomials. However, it should be noted that such equivalence is only a sufficient condition for two PMP instances to have the same optimal solution. The following counter-example illustrates this.

Example. Consider two instances defined by costs matrices C and D:

$$C = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix} \quad \text{and} \quad D = \begin{bmatrix} 1 & 1 \\ 3 & 3 \end{bmatrix} \tag{2.95}$$

If $p = 1$ then the Hammer-Beresnev functions $\mathcal{B}_C(\mathbf{y}) = 2 + 2y_1$ and $\mathcal{B}_D(\mathbf{y}) = 2 + 4y_1$ are different but the instances have a unique optimal solution $\mathbf{y}^* = (0, 1)^{\mathrm{T}}$. $\triangleleft$

Let us consider the set of all PMP instances that are equivalent to a given instance defined by matrix C:

$$\mathcal{P}_{C,p} = \left\{ D \in \mathbb{R}_+^{mn} : \ \mathcal{B}_{D,p} = \mathcal{B}_{C,p} \right\}. \tag{2.96}$$

This set can be defined as

$$\mathcal{P}_{C,p} = \bigcup_{\Pi \in PERM(C)} P_{C,\Pi,p} \ , \tag{2.97}$$

where

$$P_{C,\Pi,p} = \left\{ D \in \mathbb{R}_+^{mn} : \mathcal{B}_{C,p} = \mathcal{B}_{D,\Pi,p} \right\}. \tag{2.98}$$

and $PERM(C)$ is a set of all $m \times n$ permutation matrices that can be induced by $\mathcal{B}_{C,p}$. Any set of embedded terms of Hammer-Beresnev polynomial defines at least one permutation of $\{1, \ldots, m\}$ that can be viewed as some column of permutation matrix. We say that a permutation matrix Π of equivalent instance is induced by $\mathcal{B}_{C,p}$ if it has the same size as C, each its column is defined by some set of embedded terms of $\mathcal{B}_{C,p}$ and each terms is used in generation of some column of Pi.

Let us now consider the set $PERM(C)$. It should be noted that $perm(C) \subseteq PERM(C)$ and having fixed some permutation $\Pi \in perm(C)$ all other elements of this set can be obtained by application of operations from a finite set to it. First such operation is a permutation of the columns of Π – it is clear that it does not affect the polynomial. Second operation is a permutation of such elements i and k from some column j for which holds $c_{\pi_{ij}j} = c_{\pi_{kj}j}$. It is easy to check that all terms containing either of variables $y_{\pi_{ij}}$ and $y_{\pi_{kj}}$ (but not both) have zero coefficients. In order to provide the following two operations, it is useful to introduce matrix representation of the Hammer-Beresnev polynomial: each such polynomial can be represented as an $m \times n$ matrix, every entry of which corresponds to some monomial of the Hammer-Beresnev polynomial (possibly with zero coefficient). Moreover, the following restrictions take place:

(a) every row i contains monomials of i-th degree;

(b) monomials within each column have embedded terms.

Having a polynomial and a fixed permutation it is possible to restore the matrix representation (the inverse is also true – having a matrix representation it is possible to restore the polynomial and some permutation(s)). For example,

$$\mathcal{B}_C(\mathbf{y}) = 7 + 5y_1 + 4y_2 + 4y_1y_2 \quad \text{and} \quad \Pi = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 3 \end{bmatrix} \tag{2.99}$$

lead to many optional matrix representations, for example

$$\begin{bmatrix} 7 & 0 \\ 5y_1 & 4y_2 \\ 3y_1y_2 & 1y_1y_2 \end{bmatrix}, \begin{bmatrix} 3 & 4 \\ 5y_1 & 4y_2 \\ 0y_1y_2 & 4y_1y_2 \end{bmatrix} \tag{2.100}$$

It can be seen from this example that by interchanging the entries in the second row the restrictions (a) and (b) on the matrix representation of the polynomial are not violated. At the same time, this leads to a different permutation matrix:

$$\begin{array}{cc} \text{polynomial} & \text{permutation} \\ \begin{bmatrix} 3 & 4 \\ 5y_1 & 4y_2 \\ 3y_1y_2 & 1y_1y_2 \\ 4y_1y_2y_3 & 5y_1y_2y_4 \end{bmatrix} & \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix} \\ \downarrow & \downarrow \\ \begin{bmatrix} 3 & 4 \\ 4y_2 & 5y_1 \\ 3y_1y_2 & 1y_1y_2 \\ 4y_1y_2y_3 & 5y_1y_2y_4 \end{bmatrix} & \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 3 & 4 \\ 4 & 3 \end{bmatrix} \end{array} \tag{2.101}$$

Thus, the third operation is permutation of the entries within one row such that the embedded structure of the matrix is preserved. The fourth operation is reduction of similar monomials; it leads to an increase in the number of zero coefficients in the matrix representation and thus provides more opportunities for application of the second operation:

$$\begin{array}{cc} \text{polynomial} & \text{permutations} \\ \begin{bmatrix} 3 & 4 \\ 5y_1 & 4y_2 \\ 3y_1y_2 & 1y_1y_2 \\ 4y_1y_2y_3 & 5y_1y_2y_4 \end{bmatrix} & \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix} \\ \downarrow & \downarrow \\ \begin{bmatrix} 3 & 4 \\ 5y_1 & 4y_2 \\ 4y_1y_2 & 0y_1y_2 \\ 4y_1y_2y_3 & 5y_1y_2y_4 \end{bmatrix} & \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 2 & 4 \\ 3 & 2 \\ 4 & 3 \end{bmatrix} \end{array} \tag{2.102}$$

Similarly to AlBdaiwi et al. (2009), we show that the set $P_{C,\Pi}$ can be described by a system of linear inequalities.

Let us assume that $\Pi, \Psi \in perm(C)$. The choice of the particular Π and Ψ is unimportant since the truncated Hammer-Beresnev polynomials for all permutations within $perm(\cdot)$ are identical (see AlBdaiwi et al., 2011). The truncated Hammer-Beresnev polynomial for C is

$$\begin{aligned} \mathcal{B}_{C,p}(\mathbf{y}) \quad = & \quad \sum_{j=1}^{n} \Delta c[1,j] + \sum_{j=1}^{n} \Delta c[2,j] y_{\pi_{1j}} + \\ & \quad \sum_{k=3}^{m-p} \sum_{j=1}^{n} \Delta c[k,j] \prod_{r=1}^{k} y_{\pi_{rj}}. \end{aligned} \tag{2.103}$$

while that for D is

$$\begin{aligned} \mathcal{B}_{D,p}(\mathbf{y}) \quad = & \quad \sum_{j=1}^{n} \Delta d[1,j] + \sum_{j=1}^{n} \Delta d[2,j] y_{\psi_{1j}} + \\ & \quad \sum_{k=3}^{m-p} \sum_{j=1}^{n} \Delta d[k,j] \prod_{r=1}^{k} y_{\psi_{rj}}. \end{aligned} \tag{2.104}$$

For the PMP defined on D to be equivalent to the PMP defined on C, $\mathcal{B}_{D,p}(\mathbf{y})$ has to be equal to $\mathcal{B}_{C,p}(\mathbf{y})$. By equating similar monomials in $\mathcal{B}_{C,p}(\mathbf{y})$ and $\mathcal{B}_{D,p}(\mathbf{y})$, we see that for equivalence entries in D have to satisfy the following equations.

Equating the coefficients of the constant and linear monomials in (2.103) and

(2.104) yields

$$\begin{aligned}\sum_{j=1}^{n}\Delta d[1,j] &= \sum_{j=1}^{n}\Delta c[1,j] && (2.105)\\ \sum_{j:\psi_{1j}=k}\Delta d[2,j] &= \sum_{j:\pi_{1j}=k}\Delta c[2,j] \quad k=1,\ldots,m-p. && (2.106)\end{aligned}$$

By equating the coefficients of non-linear monomials we get the equations

$$\sum_{\{\psi_{1j},\ldots,\psi_{kj}\}=\{\pi_{1j},\ldots,\pi_{kj}\}}\Delta d[k,j]-\Delta c[k,j]=0 \quad k=3,\ldots,m-p;\ j=1,\ldots,n. \quad (2.107)$$

Finally, since $\Pi \in perm(C)$ and $\Psi \in perm(D)$, and since all entries in the instances are assumed to be non-negative, we have that

$$\begin{aligned}\Delta d[k,j] &\geq 0 && k=1,\ldots m-p;\ j=1,\ldots,n && (2.108)\\ d_{ij} &\geq 0 && i=1,\ldots,m;\ j=1,\ldots,n && (2.109)\end{aligned}$$

Consider the instance in (2.18) with $p = 1$ and $\mathcal{B}_C(\mathbf{y}) = \mathcal{B}_{C,p=1} = 8+0y_1+1y_2+2y_4+0y_1y_2+1y_1y_3+1y_2y_3+1y_2y_4+1y_3y_4+7y_1y_2y_3+1y_1y_3y_4+5y_2y_3y_4$. Then P_{C,Π_1} (where Π_1 is given by (2.19)) is defined by the following system. Note that by adding a specific permutation we just specify the names of entries in an equivalent matrix.

Equations corresponding to constants (2.105):

$$d_{11}+d_{32}+d_{23}+d_{14}+d_{45}=33. \quad (2.110)$$

Equations corresponding to linear monomials (2.106):

$$\begin{aligned} y_1: &\quad (d_{31}-d_{11})+(d_{24}-d_{14})=0,\\ y_2: &\quad (d_{32}-d_{22})=1,\\ y_3: &\quad (d_{42}-d_{32})=0,\\ y_4: &\quad (d_{23}-d_{43})+(d_{35}-d_{45})=2.\end{aligned}$$

Equations corresponding to non-linear monomials (2.107):

$$\begin{aligned} y_1y_2: &\quad (d_{34}-d_{24})=0,\\ y_1y_3: &\quad (d_{21}-d_{31})=1,\\ y_2y_3: &\quad (d_{42}-d_{32})=1,\end{aligned}$$

$$
\begin{aligned}
&y_2y_4: \quad (d_{33}-d_{23})=1,\\
&y_3y_4: \quad (d_{15}-d_{35})=1,\\
&y_1y_2y_3: \quad (d_{41}-d_{21})+(d_{44}-d_{34})=7,\\
&y_1y_3y_4: \quad (d_{25}-d_{15})=1,\\
&y_2y_3y_4: \quad (d_{12}-d_{42})+(d_{13}-d_{33})=5.
\end{aligned}
$$

Inequalities corresponding to nonnegativity of differences(2.108):

$$
\begin{aligned}
&d_{31}-d_{11}, d_{24}-d_{14}, d_{32}-d_{22}, d_{23}-d_{43}, d_{35}-d_{45}\geq 0\\
&d_{34}-d_{24}, d_{21}-d_{31}, d_{42}-d_{32}, d_{33}-d_{23}, d_{15}-d_{35}\geq 0\\
&d_{41}-d_{21}, d_{44}-d_{34}, d_{25}-d_{15}, d_{12}-d_{42}, d_{13}-d_{33}\geq 0
\end{aligned}
$$

Inequalities corresponding to nonnegativity of costs(2.109):

$$d_{11}, d_{12}, \dots, d_{44}, d_{45}\geq 0. \tag{2.111}$$

For $p=2$ we have that $\mathcal{B}_{C,p=2}=8+0y_1+1y_2+2y_4+0y_1y_2+1y_1y_3+1y_2y_3+1y_2y_4+1y_3y_4+$, and all equations corresponding to cubic terms will be replaced by

$$
\begin{aligned}
&y_1y_2y_3: \quad (d_{41}-d_{21})+(d_{44}-d_{34})=0,\\
&y_1y_3y_4: \quad (d_{25}-d_{15})=0,\\
&y_2y_3y_4: \quad (d_{12}-d_{42})+(d_{13}-d_{33})=0.
\end{aligned}
$$

Hence, given a cost matrix C, any solution D to the set of inequalities (2.105)–(2.109) will be a matrix for an equivalent instance.

Inequalities (2.105)–(2.109) define a family of polyhedra in $\mathbb{R}_+^{mn}$ (each polyhedron in the family corresponds to some permutation $\Pi_i \in perm(C)$). Further, we show that any two such polyhedra (2.98) either have an empty intersection or coincide.

From the following example it can be seen that in the simplest case, when each coefficient in the Hammer-Beresnev polynomial is defined by the costs of serving only one client (by entries of only one column in the cost matrix), the claimed property can be verified by considering only the equations (2.105)–(2.107). The polynomial is presented in a matrix form, such that each column of the matrix contains monomials with embedded terms.

Let us consider two representations of the same polynomial,

$$\begin{bmatrix} 7 & 0 \\ 5y_1 & 4y_2 \\ 4y_1y_2 & 0y_1y_2 \end{bmatrix} \begin{bmatrix} 7 & 0 \\ 4y_2 & 5y_1 \\ 4y_1y_2 & 0y_1y_2 \end{bmatrix}, \tag{2.112}$$

their corresponding cost matrices

$$\begin{bmatrix} 7 & 4 \\ 12 & 0 \\ 16 & 4 \end{bmatrix} \qquad \begin{bmatrix} 11 & 0 \\ 7 & 5 \\ 15 & 5 \end{bmatrix} \tag{2.113}$$

and permutations

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 3 \end{bmatrix} \qquad \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 3 & 3 \end{bmatrix} \quad . \tag{2.114}$$

Then, equations describing polyhedra for the two cases are

$$\begin{aligned} const.: \quad & d_{11} + d_{22} = 7 & \quad const.: \quad & d_{21} + d_{12} = 7 \\ y_1: \quad & d_{21} - d_{11} = 5 & y_2: \quad & d_{11} - d_{21} = 4 \\ y_2: \quad & d_{12} - d_{22} = 4 & y_1: \quad & d_{22} - d_{12} = 5 \\ y_1y_2: \quad & d_{31} - d_{21} = 4 & y_1y_2: \quad & d_{31} - d_{11} = 4 \end{aligned} \tag{2.115}$$

It can be seen that equations in the second line (in the third, as well) contradict each other, so the two polyhedra have empty intersection. On the other hand, the only way to eliminate this contradiction is to put r.h.s. of these equations to 0. However, in this case we will have two equivalent systems of equations:

$$\begin{aligned} d_{11} + d_{22} &= 7 & d_{21} + d_{12} &= 7 \\ d_{21} - d_{11} &= 0 & d_{11} - d_{21} &= 0 \\ d_{12} - d_{22} &= 0 & d_{22} - d_{12} &= 0 \\ d_{31} - d_{21} &= 4 & d_{31} - d_{11} &= 4 \end{aligned} \tag{2.116}$$

In a more general case, however, by considering only the equations it is not possible to prove the claimed property of the polyhedra induced by the Hammer-Beresnev polynomial. The following counter-example illustrates this point.

Consider two following matrix representations of the same polynomial

$$\begin{bmatrix} 7 & 0 & 0 \\ 5y_1 & 4y_2 & 0y_1 \\ 4y_1y_2 & 0y_1y_2 & 2y_1y_3 \end{bmatrix} \begin{bmatrix} 7 & 0 & 0 \\ 4y_2 & 5y_1 & 0y_1 \\ 4y_1y_2 & 0y_1y_2 & 2y_1y_3 \end{bmatrix}, \tag{2.117}$$

their corresponding cost matrices

$$\begin{bmatrix} 7 & 4 & 0 \\ 12 & 0 & 2 \\ 16 & 4 & 0 \end{bmatrix} \qquad \begin{bmatrix} 11 & 0 & 0 \\ 7 & 5 & 2 \\ 15 & 5 & 0 \end{bmatrix} \tag{2.118}$$

and permutations

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 3 \\ 3 & 3 & 2 \end{bmatrix} \qquad \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 3 \\ 3 & 3 & 2 \end{bmatrix} \quad . \tag{2.119}$$

Then, equations describing polyhedra for the two cases are

$$\begin{aligned} &const.: & d_{11}+d_{22}+d_{13}&=7 & &const.: & d_{21}+d_{12}+d_{13}&=7 \\ &y_1: & d_{21}-d_{11}+d_{33}-d_{13}&=5 & &y_1: & d_{22}-d_{12}+d_{33}-d_{13}&=5 \\ &y_2: & d_{12}-d_{22}&=4 & &y_2: & d_{11}-d_{21}&=4 \\ &y_1y_2: & d_{31}-d_{21}+d_{32}-d_{12}&=4 & &y_1y_2: & d_{31}-d_{11}+d_{32}-d_{22}&=4 \\ &y_1y_3: & d_{23}-d_{33}&=2 & &y_1y_3: & d_{23}-d_{33}&=2 \end{aligned} \tag{2.120}$$

There are no contradicting equations in the two systems and it can be verified that they have common solutions. Thus, if only equalities are considered, the polyhedra can intersect in general case, but, if constraints on non-negativity (2.108)–(2.109) are added, it is possible to formulate the following lemma.

Lemma 2.7. *Polyhedra induced by different permutation matrices either do not intersect or coincide.*

Proof. Suppose, for some column j it is possible to change its permutation from $\pi_1 = (\cdots, i, k, \cdots)^{\mathrm{T}}$ to $\pi_2 = (\cdots, k, i, \cdots)^{\mathrm{T}}$, so that only two adjacent entries are interchanged. The first permutation will lead to the following inequality (among the others)

$$d_{i,j} - d_{k,j} \geq 0$$

Similarly, the second permutation leads to a system that includes

$$d_{k,j} - d_{i,j} \geq 0$$

It is straightforward that the polyhedra intersect only if $d_{k,j} = d_{i,j}$, but in this case the systems of equations that correspond to the two cases (permutations, matrices) are equivalent. ■

As mentioned above (2.97), the union $\mathcal{P}_C$ of polyhedra corresponding to possible permutations of the cost matrix C describes all equivalent PMP instances. Once the equivalence relation is established, it is natural to estimate the dimension of equivalent data (dimension of $\mathcal{P}_C$). In order to do that, we introduce additional notations. Let us denote the system of equations (2.105)–(2.107) by $\mathcal{E}$. In turn, $\mathcal{E}$ can be presented in a matrix form as $\mathbf{A} \cdot d = b$, where $\mathbf{A}$ – $mn \times N$ matrix, d - vector of entries of the cost matrix (variables), $d = (d_{11}, d_{21}, \ldots, d_{mn})^{\mathrm{T}}$, N – the number of equations in (2.105)–(2.107). Under these notations it is possible to formulate the following properties of $\mathcal{E}$ (we assume that **A** has at least 2 rows, i.e. $m \geq 2$).

Observation 1. *For any two equations eq_1 and eq_2 from $\mathcal{E}$ there exist two variables d_{ij} and d_{kl}, such that*

$$d_{ij} \in eq_1, d_{ij} \notin eq_2 \tag{2.121}$$

$$d_{kl} \in eq_2, d_{kl} \notin eq_1 \tag{2.122}$$

Observation 2. *All coefficients in the equations from $\mathcal{E}$ belong to $\{-1, 1\}$.*

Observation 3. *There exist variables $d_{mi}, i = 1, \dots, n$ that are included in exactly one equation with coefficient +1. All the other variables are included in exactly two equations with coefficients +1 and -1, correspondingly.*

The latter becomes clear if one considers equations corresponding to coefficients of a chain of embedded terms of the Hammer-Beresnev polynomial.

Observations 1 and 2 have an important consequence formalised in the following Lemma.

Lemma 2.8. *Constraints matrix* **A** *describing the polyhedron of equivalent instances is totally unimodular. The transposed minor of A excluding the first row (the one corresponding to constraints (2.105)) is also totally unimodular if all subpermutations in Π are different, i.e. if the pseudo-Boolean representation of the instance has no similar monomials.*

Proof. Th first statement can be easily verified by observing that every column of **A** has at most two nonzero entries and they the opposite sign. Together with the statement of Observation 2 this matches the well-known sufficient conditions for total unimodularity (see, e.g., Papadimitrou & Steiglitz, 1998, Theorem 13.3).

In order to prove the second statement, observe that absence of similar monomials implies that each coefficient in the Hammer-Beresnev polynomial (except the constant) is uniquely defined as a difference of two entries of the costs matrix. This means that each row of **A** (correspondingly, each column of $\mathbf{A}^{\mathrm{T}}$ except the first one has exactly two nonzero entries of the opposite sign. Thus, the sufficient conditions used in the first part of the proof are satisfied. ■

Lemma 2.8 implies that the polyhedron of equivalent instances has only integral vertices and that each costs matrix within an equivalence class can be represented as a conic combination of some integer-valued costs matrices from the same equivalence class.

The following observations reflect some additional properties of the constraints defining the equivalence polyhedron.

Observation 4. *If some two equations eq_1, eq_2 from $\mathcal{E}$ contain the same variable d_{ij}, then exactly one of the following holds (λ_1, λ_2– some constants):*

- ○ *$d_{ij} \in \lambda_1 eq_1 + \lambda_2 eq_2$ and there exist no other equation $eq_3 \in \mathcal{E}$ that contains d_{ij};*
- ○ *$d_{ij} \notin \lambda_1 eq_1 + \lambda_2 eq_2$ and there exist no other equation $eq_3 \in \mathcal{E}$ that contains d_{ij}.*

Observation 5. (ELIMINATION OF VARIABLES) *If in a linear combination $\lambda_1 eq_1 + \lambda_2 eq_2$ of two equations from $\mathcal{E}$ some variable d_{ij} (that was present in both of them) is eliminated then this combination contains $d_{(i+1)j}$ and $d_{(i-1)j}$, and no other equation from $\mathcal{E}$ contains both these variables.*

Lemma 2.9. *The system of equations $\mathcal{E}$ is linearly independent.*

Proof. By definition, the system is linearly dependent if there exist such constants $\lambda_1, \ldots, \lambda_N$ ($\sum_i |\lambda_i| > 0$) that $\mathcal{L} \equiv \lambda_1 eq_1 + \lambda_2 eq_2 + \ldots + \lambda_N eq_N = 0$, or, in other words, coefficients at all variables in $\mathcal{L}$ are zero. Let us show that such set of constants $\lambda_i, i = 1 \ldots N$ does not exist by induction on the number of equations in $\mathcal{L}$.

For $N = 1$ – trivially, in each equation there are variables with non-zero coefficients (see Fact 2).

Suppose, for some $N > 1$ holds $\mathcal{L} = \lambda_1 eq_1 + \lambda_2 eq_2 + \ldots + \lambda_N eq_N \neq 0$, this means that $\mathcal{L}$ contains at least one variable with a non-zero coefficient. Let us show that by adding one more equation to $\mathcal{L}$ one will not obtain 0. If $\mathcal{L}$ and eq_{N+1} do not have common variables then for any λ_{N+1} holds

$$\mathcal{L} + \lambda_{N+1} eq_{N+1} \neq 0 \tag{2.123}$$

If $\mathcal{L}$ and eq_{N+1} have some common variable d_{ij} then exactly one of the following cases takes place:

- ○ $d_{(i+1)j} \in \mathcal{L},\ d_{(i+1)j} \notin eq_{N+1}$
- ○ $d_{(i+1)j} \in eq_{N+1}, d_{(i+1)j} \notin \mathcal{L}$
- ○ $d_{(i+1)j} \notin eq_{N+1},\ d_{(i+1)j} \notin \mathcal{L}$

In the first two cases we have a variable that cannot be eliminated by adding $\lambda_{N+1} eq_{N+1}$ to $\mathcal{L}$. The third case implies that $d_{(i+1)j}$ was eliminated in $\mathcal{L}$ and according to Fact 5 there exist some d_{kj}, $k > i$ that is in $\mathcal{L}$ and not in eq_{N+1}. This finishes the proof. ■

As all equations in $\mathcal{E}$ are linearly independent, then $\mathbf{A}$ has the maximum possible rank that is equal to the number of rows in it, i.e. $rank(\mathbf{A}) = N$. If one now denotes by $|T|$ the number of monomials in the initial Hammer-Beresnev polynomial (equal to the number of non-zero entries in the difference matrix Δ) and by $|B|$ - the number of terms in the reduced polynomial (with combined similar monomials) then the following bounds on $rank(\mathbf{A})$ take place.

Lemma 2.10.

$$rank(\mathbf{A}) \geq |B| \tag{2.124}$$

$$rank(\mathbf{A}) \leq |B| + mn - |T| \tag{2.125}$$

Proof. As every term with a non-zero coefficient corresponds to an equation in $\mathcal{E}$, then the number of equations cannot be smaller than $|B|$. However, equations for some terms of the polynomial with zero coefficients are to be included into the system as their corresponding zero entries existed in the difference matrix. So, the upper bound is obtained by adding the quantity of zeros induced by equal entries in the columns of the costs matrix. ■

It should be noted that for a PMP instance defined on a cost matrix D to be equivalent to a PMP instance defined on a cost matrix C, $perm(D)$ has to be identical to $perm(C)$. Note that a choice among many equivalent matrices might be refined by adding either additional constraints reflecting some sufficient conditions for the polynomially solvable special case as well as some additional requirements to the entries included in, for the sake of simplicity, a linear objective function defined on $P_{C,p}$.

The problem of finding an equivalent matrix D with the minimum number of columns to the given matrix C can be solved using the following well-known Dilworth's decomposition theorem (see, e.g., Schrijver, 2003, Theorem 14.2):

"The set of terms T_a with positive coefficients in a pseudo-Boolean polynomial are subsets of partially ordered set T, and hence, the minimum number of chains covering T_a (nothing else as the minimum number of aggregated columns of C) is equal to the maximum size of an antichain (the maximum number of non-embedded terms)."

The maximum size of an antichain found for matrix (2.18) is equal to four, and if the permutation matrix Π_E is chosen to be

$$\Pi_E = \begin{bmatrix} 1 & 2 & 4 & 3 \\ 3 & 3 & 2 & 4 \\ 2 & 1 & 1 & 1 \\ 4 & 4 & 3 & 2 \end{bmatrix} \tag{2.126}$$

then the corresponding Hammer-Beresnev polynomial $\mathcal{B}_{C,p=2}(\mathbf{y}) = [1 + 0y_1 + 1y_1y_3 + 2y_1y_2y_3] + [1 + 1y_2 + 1y_2y_3 + 3y_2y_3y_4] + [1 + 1y_4 + 1y_2y_4 + 2y_2y_3y_4] + [3 + 0y_1 + 0y_1y_2 + 5y_1y_2y_3] + [2 + 1y_4 + 1y_3y_4 + 1y_1y_3y_4]$ yields the following (in)equalities. Equations corresponding to constants (2.105):

$$e_{11} + e_{22} + e_{43} + e_{34} = 8 \tag{2.127}$$

Equations corresponding to linear monomials (2.106):

$$\begin{aligned} y_1 :& \quad (e_{31} - e_{11}) = 0 \\ y_2 :& \quad (e_{32} - e_{22}) = 1 \\ y_3 :& \quad (e_{44} - e_{34}) = 0 \\ y_4 :& \quad (e_{23} - e_{43}) = 2 \end{aligned}$$

Equations corresponding to non-linear monomials (2.107):

$$\begin{aligned} y_1y_3 :& \quad (e_{21} - e_{31}) = 1 \\ y_2y_3 :& \quad (e_{12} - e_{32}) = 1 \\ y_2y_4 :& \quad (e_{13} - e_{23}) = 1 \\ y_3y_4 :& \quad (e_{14} - e_{44}) = 1 \\ y_1y_2y_3 :& \quad (e_{41} - e_{21}) + (e_{42} - e_{12}) = 0 \\ y_1y_2y_4 :& \quad (e_{33} - e_{13}) = 0 \\ y_1y_3y_4 :& \quad (e_{24} - e_{14}) = 0 \end{aligned}$$

Nonnegativity inequalities corresponding to (2.108):

$$\begin{aligned} & e_{31} - e_{11}, e_{32} - e_{22}, e_{44} - e_{34} \geq 0 \\ & e_{23} - e_{43}, e_{21} - e_{31}, e_{12} - e_{32} \geq 0 \\ & e_{13} - e_{23}, e_{14} - e_{44}, e_{41} - e_{21} \geq 0 \end{aligned}$$

$$e_{42} - e_{12}, e_{33} - e_{13}, e_{24} - e_{14} \geq 0$$

Nonnegativity inequalities corresponding to (2.109):

$$e_{11}, e_{12}, \ldots, e_{43}, e_{44} \geq 0 \tag{2.128}$$

Finally, the reconstructed costs matrix E that is equivalent to matrix C (2.18) is

$$E = \begin{bmatrix} 8 & 2 & 3 & 1 \\ 9 & 0 & 2 & 1 \\ 8 & 1 & 3 & 0 \\ 9 & 2 & 0 & 0 \end{bmatrix} \tag{2.129}$$

2.7 Summary and Future Research Directions

This chapter presents a new approach to formulation of models for the p-Median problem (PMP). We first formulate the PMP using a pseudo-Boolean polynomial (pBp) as the objective function, and with just one constraint related to the number of medians in a feasible solution keeping all decision variables Boolean. We then reduce the size of the objective function by truncation and reducing similar monomials. After that we linearise all non-linear terms in the objective function with additional variables and linear constraints. The resulting model that we call Mixed Boolean pseudo-Boolean Model (MBpBM) is within the well studied class of Mixed Boolean Linear Programming models. The number of non-zero coefficients in the objective function of MBpBM is minimal compared to all previously published models for the PMP. Since the number of Boolean decision variables is equal to m and cannot be further reduced (except by well known reduction tests finding some open and/or closed sites, see, e.g., Goldengorin et al. (2003), Avella et al. (2007), AlBdaiwi et al. (2009) and references within) and the number of linear constraints is equal to the number of non-negative variables one may conclude, that the MBpBM has the smallest number of constraints related to the non-negative variables. As we have shown, the matrix of constraints related to the non-negative decision variables is as sparse as possible if we would be able to solve a generalization of the classical set covering problem defined on the set of all terms involved in the pseudo-Boolean formulation of PMP. Unfortunately, this set covering problem is NP-hard (Garey & Johnson, 1979). As shown by our computational experiments for a PMP instance with $m = n = 1000$ the corresponding ground set of covering problem might be in magnitudes larger. From the other side, even if we might be able to find the most sparse matrix of constraints, then the created MBpBM is still in the class of mixed Boolean linear programming models which are computationally intractable (NP-hard). Anyway, if we evaluate the optimality of a model within the class of Mixed Boolean Linear Programming models by the smallest number of non-negative variables and corresponding constraints, our MBpBM is an optimal one and PMP instance specific!

The main distinction between MBpBM and all the well known Mixed Boolean PMP formulations is that the number of non-negative variables in MBpBM is automatically adjusted according to the number p of medians, i.e. the number of non-negative variables as well as the number of constraints decrease linearly with increasing values of p. This feature of MBpBM implies that PMP instances with relatively large numbers of medians are easier to solve using standard MILP software applied with our MBpBM. Thus, our models (MBpBM, MBpBMb and MBpBMb1) and pseudo-Boolean approach to their creation do not only extend the capabilities of general-purpose software in solving larger sized p-median problems, but also makes it possible to solve smaller problems more efficiently while using general-purpose MILP software.

The MBpBM allows either to solve much larger problems by general-purpose MILP software than what is possible using previous model formulations or to speed up essentially the best known models for the PMP. In sharp contrast, CPLEX is unable to solve the 15 largest OR test problems (see Table 1 in this paper) in classical formulation of PMP (see Avella et al., 2007; Beltran et al., 2006)).

The MBpBM approach may also lead to reduced model sizes for other location models like the simple plant location problem and the capacitated simple plant location problem (a generalization of PMP with fixed charges) as well as to improve data correcting approach to the SPLP (see Goldengorin et al., 2003). Together with the MBpBM we have introduced two variations of MBpBM, namely MBpBMb and MBpBMb1. The MBpBMb includes preprocessing of monomials and corresponding linear constraints based on a lower bound to a subproblem of MBpBM induced by a subspace determined by the term of the corresponding monomial in pBp. The MBpBMb1 is based on further reductions of monomials in pBp induced by a monomial with zero coefficient and embedded in all terms with higher degrees.

Computational results reported in Tables 2.3–2.4 show that MBpBMb1 outperforms the best available MILP Elloumi's model for all OR-library instances except pmed38. Our MBpBM allows solving PMPs with much less execution times than required by the best known models in the literature. It also allows solving much larger problems by general-purpose MILP software than is currently possible using previous model formulations. The MBpBM has been able to obtain an optimal solution to the fl1400 instance with $p = 400$, which remained unsolved in Avella et al. (2007) by their state-of-the-art-algorithm for PMP as well as by Beltran's et al. advanced semi-Lagrangean approach based on Proximal-Analytic Center Cutting Plane Method (Beltran et al., 2006). Our models MBpBM, MBpBMb, MBpBMb1 are computationally more efficient than all available in the literature MILP formulations of PMP and outperform corresponding state-of-the-art algorithms available in the literature, see Cornuejols et al. (1980), Church (2003), Church (2008), Senne et al. (2005), Beltran et al. (2006), Avella et al. (2007), Brusco & Köhn (2008), and Elloumi (2010).

Computational results reported in Table 2.5 show that our MBpBM will be useful for p-median approach applicable to large cell formation instances in group technology such that the optimal number p of cells can be found (see, e.g., Won & Lee, 2004). To summarize, in this paper we have shown that our model extends the ability to solve large scale p-median problem instances to optimality by means of general-purpose software, e.g., Xpress-MP.

With regard to the main subject of this thesis – the cell formation problem – one may conclude that application of PMP to solving the cell formation problem is beneficial as the former can be solved to optimality very efficiently using the introduced MBpBM formulation. Taking into account that the size of the problem is quite limited (e.g., 100 machines is already too much for a typical manufacturing system) one may expect tiny solution times and this expectation will be verified in the following chapters.

Table 2.4. Comparison of computing times for our and Elloumi's NF formulations (Resende and Werneck random instances)

instance	m	p	MBpBM	MBpBMb	MBpBMb1	Elloumi
rw100	100	10	678.91	671.28	452.52	845.30
	100	20	4.00	6.44	2.22	5.25
	100	30	0.09	0.43	0.03	0.13
	100	40	0.08	0.34	0.02	0.14
	100	50	0.06	0.28	0.02	0.13
rw250	250	10	–	–	–	–
	250	25	–	–	–	–
	250	50	340.86	1633.58	225.83	335.86
	250	75	1.09	8.50	0.48	2.08
	250	100	0.50	5.44	0.11	0.88
	250	125	0.66	5.02	0.27	1.38
rw500	500	10	–	–	–	–
	500	25	–	–	–	–
	500	50	–	–	–	–
	500	75	–	–	–	–
	500	100	–	–	–	–
	500	150	2.97	105.63	1.22	12.27
	500	200	2.25	54.78	0.28	4.11
	500	250	1.77	44.83	0.13	4.36
rw1000	1000	10	–	*	–	–
	1000	25	–	*	–	–
	1000	50	–	*	–	–
	1000	75	–	*	–	–
	1000	100	–	*	–	–
	1000	200	–	*	–	–
	1000	300	118.91	*	13.40	234.99
	1000	400	11.49	*	1.16	21.81
	1000	500	9.08	*	0.77	28.47

– Not solved within 1 hour.
$*$ Not enough memory for the MILP solver

Table 2.5. MBpBM for different numbers of medians in pmed39/pmed40

	pmed39				pmed40			
p	$f_C(S^*)$	M_{tr}	constr	MBpBM	$f_C(S^*)$	M_{tr}	constr	MBpBM
1	14720	29042	706612	7.3	17425	31641	744257	12.8
5	11069	28839	705976	79.7	12305	31268	743056	39.3
9	9690	27883	702341	429.2	10740	30155	738633	54.8
10	9423	27637	701168	121.2	10491	29905	737406	88.0
20	7894	26008	690135	80.5	8717	28143	725285	104.3
30	7051	24983	679640	567.4	7731	27109	714318	138.7
40	6436	24272	669999	182.9	7037	26372	703930	113.4
50	5941	23688	660138	93.9	6518	25813	694355	782.7
60	5545	23229	651280	38.6	6083	25304	684402	171.9
70	5215	22815	641971	5.7	5711	24883	674846	33.3
80	4929	22439	632520	4.6	5398	24503	665476	5.5
90	4684	22147	624079	4.5	5128	24164	656067	5.7
100	4462	21844	615198	4.9	4878	23851	646520	5.4
200	2918	19731	529883	2.7	3132	21623	557661	3.1
300	1968	18252	449160	2.0	2106	20066	473237	2.4
400	1303	17000	371790	1.5	1398	18725	391820	1.7
500	821	15812	298029	1.3	900	17431	314045	1.3
600	471	14495	223027	1.0	530	16040	237199	0.9
700	244	12962	151916	0.7	271	14337	161826	0.6
800	100	10684	81510	0.3	100	11781	86772	0.7

Table 2.6. Running times in seconds for our MBpBM and Elloumi's NF for OR instances and our complex instances of the corresponding size

			OR instances			our instances	
m	p	MBpBM	MBpBMb	Elloumi	MBpBM	MBpBMb	Elloumi
100	5	0.22	0.20	0.25	4434.66	3443.13	24684.42
	10	1.47	0.58	4.08	878.78	1141.05	3926.20
	20	0.11	0.06	0.14	92.95	26.25	62.94
	33	0.22	0.05	0.13	0.28	0.11	0.61
200	5	15.22	17.67	17.06	*	*	*
	10	0.73	0.55	0.77	*	*	*
	20	0.49	0.31	0.55	*	*	*
	40	0.41	0.28	0.45	1616.33	1218.45	1753.47
	67	0.27	0.14	0.41	1.08	0.63	1.34
300	5	4.00	4.61	4.50	*	*	*
	10	8.59	8.33	7.36	*	*	*
	30	0.80	0.56	1.25	*	*	*
	60	1.05	1.13	2.34	*	*	*
	100	0.48	0.30	0.86	1.16	0.27	1.81
400	5	42.47	30.78	23.38	*	*	*
	10	25.16	21.19	32.02	*	*	*
	40	1.73	1.31	2.97	*	*	*
	80	0.97	0.72	1.61	*	*	*
	133	0.73	0.80	1.25	3.83	1.86	6.28
500	5	4.52	3.92	6.22	*	*	*
	10	51.63	64.05	98.59	*	*	*
	100	1.42	0.97	2.33	*	*	*
	167	1.44	0.88	1.84	14.91	4.14	18.56
600	5	163.84	111.81	180.31	*	*	*
	10	27.59	21.31	43.73	*	*	*
	120	1.78	1.31	2.91	*	*	*
	200	1.50	0.78	4.81	49.81	15.41	201.39
700	5	153.22	57.05	90.95	*	*	*
	10	33.13	43.39	37.64	*	*	*
	70	3.09	2.69	4.73	*	*	*
	140	3.72	1.97	7.11	*	*	*
800	5	70.30	154.41	514.72	*	*	*
	10	2256.83	4252.13	6737.25	*	*	*
	80	3.91	3.08	7.00	*	*	*
900	5	1328.34	2041.28	1143.97	*	*	*
	10	572.81	444.08	473.95	*	*	*
	90	5.39	4.02	8.42	*	*	*

* not solved within 24 hours

Chapter 3

Application of the PMP to cell formation in group technology

3.1 Introduction

Cell formation, being a popular concept in industrial engineering, suggests grouping machines into manufacturing cells and parts into product families such that each family is processed mainly within one cell. The problem of optimal (usually, with respect to the amount of intercell movement) cell formation has been studied by many researchers. An overview can be found in (Selim et al., 1998; Yin & Yasuda, 2006) and recently in Bhatnagar & Saddikuti (2010). However, no tractable algorithms that guarantee optimality of the obtained solutions were reported because of computational complexity of the problem. Moreover, even worst-case performance estimates are not available for most approaches. In fact, it was only checked that they produce good solutions for artificially generated instances without any kind of worst or average case analysis. At the same time, today's highly competitive environment makes it extremely important to increase the efficiency of manufacturing systems as much as possible. In these conditions any noticeable improvement (e.g., achieved by properly designed manufacturing cells) can provide a secure position for a company in a highly competitive market.

This chapter is based on the papers (Goldengorin & Krushinsky, 2011b; Goldengorin et al., 2012).

3.1.1 Background

The problem of cell formation can be traced back to the works of Flanders (1925) and Sokolovski (1937) but is oftenly attributed to Mitrofanov's group technology (Mitrofanov, 1959, 1966) and Burbidge's product flow analysis (PFA, see Burbidge, 1961). Burbidge showed that it can be reduced to a functional grouping of machines based on binary machine-part incidence data. Thus, in its simplest and earliest form cell formation is aimed at the functional grouping of machines based on similarity of the sets of parts that they process. Input data for such a problem is usually given by a binary machine-part incidence matrix $\mathbf{A} = [a_{ij}]$, where $a_{ij} = 1$ if and only if part j machine i at some step of its production process. In mathematical terms, the problem of cell formation was first defined as one of finding independent permutations of rows and columns that lead to an (almost) block-diagonal structure of matrix **A** – *uncapacitated functional grouping*.

Early approaches to cell formation (McCormick et al., 1972; King, 1980; Chandrasekharan & Rajagopalan, 1986b; H. M. Chan & Milner, 1982; Kusiak & Chow, 1987) (see Figure 3.1) were restricted to the functional grouping and produced optimal results only for the data with a perfect cellular structure. The next step in the development of the cell formation problem was made by introducing the production volumes issue. A binary input matrix was replaced by a real-valued one with entries reflecting actual production volumes. Further, various types of data from real manufacturing systems (e.g. operational sequences, alternative routings, available workers) and/or additional constraints (e.g., on the number of machines or workers within a cell, workload) were taken into account leading to a bunch of new approaches to solving the corresponding problems. Some of these approaches use genetic (Mak et al., 2000; Filho & Tiberti, 2006) or simulated annealing heuristics (Adil & Rajamani, 2000; Xambre & Vilarinho, 2003). Others focused on artificial neural networks exploiting their ability of (self-)learning and a variety of available architectures and learning paradigms: backpropagation learning Kao & Moon (1991), competitive learning (Malave & Ramachandran, 1991), adaptive resonance theory (Suresh et al., 1999; Yang & Yang, 2008), self-organizing maps (Guerrero et al., 2002). Despite being robust and adaptive, neural networks usually need an adjustment of learning parameters that are hard to interpret and are selected on a trial-and-error basis. In contrast to neural network approaches, the ones based on mixed-integer linear programming, MILP, (Chen & Heragu, 1999; Slomp et al., 2005) and graph theory, in particular, on the p-Median problem (Wang & Roze, 1997; Deutsch et al., 1998; Ashayeri et al., 2005; Won & Currie, 2006) and its modifications (see, e.g., Bhatnagar & Saddikuti, 2010), are easy to understand and to interpret, and need only the dissimilarity measure to be defined for each pair of machines. However, as these approaches lead to computationally intractable (NP-hard) problems (except the ones based on the minimum spanning tree problem, see, e.g., Ng (1996)), researchers still focus on development of sophisticated heuristics. We also would like to mention the paper by Chen & Heragu (1999) separately, because it differs from the bulk of other works in several aspects. First of all, authors made an attempt to solve the problem exactly. However, they were able to solve optimally only instances of moderate size. Secondly, their formulation does not use a dissimilarity measure and deals directly with machine-part relations. This makes the applicability of the model questionable, as in the real systems the number of parts can be estimated in thousands leading to a huge formulation, even though the number of machines is small. For example, an instance with 4415 parts was considered by Park & Suresh (2003), and we experienced much larger ones.

Thus, until now there is no approach that guarantees optimality of obtained solutions. In fact, for most of the available approaches even worst-case quality analysis is not available. This means that for real problems it is not known how far from true optima the obtained solutions are. Finally, most models, being an approximation of the original cell formation problem, are solved by heuristic methods thus accumulating two errors: an intrinsic error of modelling and an error induced by a heuristic solution method.

3.1.2 Objectives and outline

This chapter is motivated by the observation that while most approaches induce a modelling error, the underlying problems are usually solved by heuristics leading to a computational error that further deteriorates the solution quality. In contrast, by means of the p-Median problem (PMP) we show that the computational error can be completely avoided. We also provide an experimental study showing that the modelling error is relatively low. Despite its NP-hardness, our PMP-based model can be solved to optimality in acceptable time for real world data just by intensively exploiting its properties and reducing its size. I.e. we can transform the original NP-hard problem into another NP-hard problem of a size small enough to allow its resolution by a general-purpose MILP solver within seconds. By this example we would like to draw the attention of the OR community to the importance of a careful model choice and opportunities provided by problem size reduction techniques.

This chapter is aimed at the development of a tractable MILP model for solving real-world cell formation problems. We present an efficient formulation that is based on the p-Median problem (PMP) and allows solving large-size cell formation instances (typical for the real manufacturing systems) in acceptable time by general-purpose MILP solvers. We show that our model outperforms contemporary approaches in terms of solutions quality and can be used as a starting point for further extensions. Even though PMP is NP-hard (Kariv & Hakimi, 1979), we present an efficient MILP formulation that intensively exploits the structure of the input data thus substantially reducing the problem size. Moreover, we show that additional linear constraints reflecting capacities of the cells, workload balancing, sequencing of operations, etc. can be incorporated into our model while preserving its practical computational tractability. We also claim that our model not only allows solving previously considered problems but also presents a new flexible framework for dealing with real world cell formation. Numerical experiments will show that our model outperforms several other contemporary approaches in quality of the obtained solutions, while keeping computing times below one second. In addition, it is worth mentioning that the applicability of our approach is not restricted to cell formation applications as models based on the p-Median problem were proposed in various fields, including cluster analysis, quantitative psychology, marketing, telecommunications industry, sales force territories design, political districting, optimal diversity management, vehicle routing, and topological design of computer communication networks (references can be found in AlBdaiwi et al. (2009)). Thus, we would like to draw attention of both – managers, industrial engineers and researchers – to the new possibilities provided by our flexible and optimally solvable p-Median model.

The chapter is organized as follows. The next section describes the p-Median approach to the cell formation problem including general formulation of the PMP, its interpretation in terms of cell formation and our efficient MILP formulation. Section 3.3 gives some examples of constraints that can be incorporated into the proposed model to illustrate its practical applicability. In Section 3.4 we provide results of our experiments with instances used in recent papers. Finally, Section 3.5 summarizes the chapter and outlines possible directions for future research.

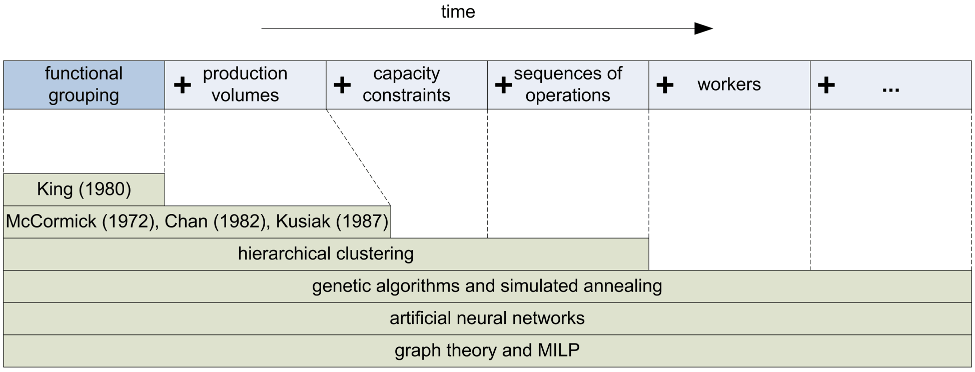

time
functional grouping
+ production volumes
+ capacity constraints
+ sequences of operations
+ workers
+ ...
King (1980)
McCormick (1972), Chan (1982), Kusiak (1987)
hierarchical clustering
genetic algorithms and simulated annealing
artificial neural networks
graph theory and MILP

Figure 3.1. Evolution of cell formation problem and applicability of approaches.

3.2 The p-Median Approach to Cell Formation

The p-Median Problem (PMP) was applied to cell formation in group technology by a number of researchers (see Won & Lee (2004), Deutsch et al. (1998) and references within). However, to the best of our knowledge, in all CF related papers PMP (as well as almost any other model based on graph partitioning or MILP) is solved by some heuristic method. At the same time, for the p-Median problem there exist efficient formulations (the most recent one derived in Elloumi (2010)) that allow solving medium and large size instances to optimality. In this paper we utilize our new model for the PMP that represents the instance data even in a more compact way thus leading to a smaller MILP formulation. This allows solving large scale CF problems to optimality within seconds.

PMP is one of well-known minisum location-allocation problems. A detailed introduction to this problem and solution methods appears in Reese (2006) and Mladenovic et al. (2007). For a directed weighted graph $G = (V, A, C)$ with $|V|$ vertices, set of arcs $(i, j) \in A \subseteq V \times V$ and weights (distances, dissimilarities, etc.) $C = \{c_{ij} : (i, j) \in A\}$, the PMP consists of determining p nodes (the *median nodes*, $1 \leq p \leq |V|$) such that the sum of weights of arcs joining any other node and one of these p nodes is minimized (see Figure 3.2).

In terms of cell formation, vertices represent machines and weights c_{ij} represent dissimilarities between machines i and j. These dissimilarities can be derived from the sets of parts that are being processed by either of the machines (e.g., if two machines process almost the same set of parts they have small dissimilarity and are likely to be in the same cell) or from any other desired characteristics (e.g. workers skill matrix, operational sequences, etc.). Moreover, usually there is no need to invent a dissimilarity measure as it can be derived from one of the available similarity measures using an expression $d(i,j) = c - s(i,j)$, where $d(.,.)/s(.,.)$ is a dis/similarity measure and c – some constant large enough to keep all dissimilarities non-negative. As can be seen from the literature, several similarity measures were proposed and the particular choice can influence results of cell formation. For our experiments we have chosen one of the most widely used – Wei and Kern's "commonality score" (Wei & Kern, 1989), and derived our dissimilarity measure as

$$d(i,j) = r \cdot (r-1) - \sum_{k=1}^{r} \Gamma(a_{ik}, a_{jk}) \tag{3.1}$$

where

$$\Gamma(a_{ik}, a_{jk}) = \begin{cases} (r-1), & if\ a_{ik} = a_{jk} = 1 \\ 1, & if\ a_{ik} = a_{jk} = 0 \\ 0, & if\ a_{ik} \neq a_{jk} \end{cases} \tag{3.2}$$

where a_{ij} - entries of the machine-part incidence matrix, r - number of parts.

Thus, if applied to cell formation, the p-Median problem means finding p machines that are best representatives (centres) of p manufacturing cells, i.e. the sum over all cells of dissimilarities between such a centre and all other machines within the cell is minimized. Once p central machines are found, the cells can be produced by assigning each other machine to the central one such that their dissimilarity is minimum. Note that the desired number of cells p is part of the input for the model and should be known beforehand. Otherwise, it is possible to solve the problem for several numbers of cells and pick the best solution.

Further, for the sake of clarity for those familiar with the PMP, we will follow the terminology inherited from location–allocation applications and represent the set of vertices V as a union of two (possibly intersecting) sets I and J, such that $|I| = m$, $|J| = n$. We will call the elements of I locations and those of J – clients. Moreover, we treat weights c_{ij} as costs of serving client j ($j \in J$) from location i ($i \in I$). In terms of cell formation the set of locations I contains potential centres of the cells and the set of clients J contains all machines. Clearly, in case of cell formation sets I and J coincide as any machine, potentially, can be a centre of a cell. This implies that PMP applied to cell formation has a symmetric costs matrix.

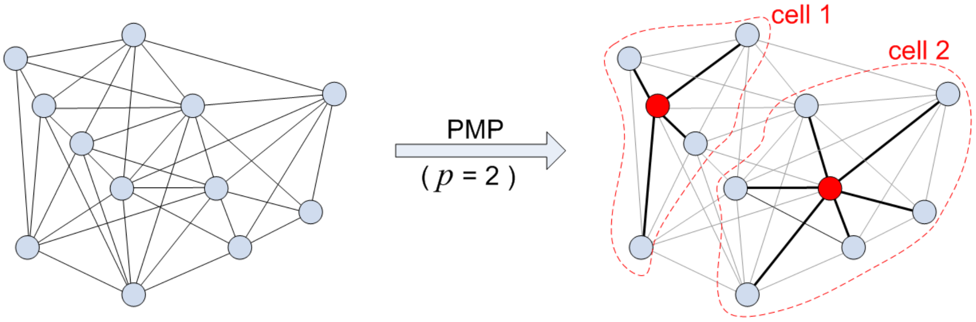


Figure 3.2. The p-Median problem: minimize the total weight of solid edges.

3.2.1 The MBpBM Formulation

Our approach is based on a compact MILP formulation for PMP – the Mixed-Boolean pseudo-Boolean formulation (MBpBM), discussed in detail in Chapter 2 and in Goldengorin & Krushinsky (2011a). Here we briefly describe the major idea behind the formulation, as needed for the further analysis.

The MBpBM formulation is derived from the so-called pseudo-Boolean formulation of PMP (see AlBdaiwi et al. (2009) or Goldengorin & Krushinsky (2011a)) that associates with a cost matrix C a permutation matrix Π, a differences matrix Δ and a vector of Boolean variables $\mathbf{y} = (y_1, \ldots, y_m)$, reflecting opened ($y_i = 0$) and closed ($y_i = 1$) locations. Each column of Π is a permutation that sorts the entries from the corresponding column of C in a nondecreasing order; each column of Δ contains differences between consecutive sorted entries of C (δ_{i1} is defined as the smallest elements in column i). It can be shown that the PMP can be expressed in terms of a polynomial on Boolean variables, abbreviated as $B_{C,p}(\mathbf{y})$, with only one constraint requiring exactly p locations to be opened: $\sum_{i=1}^{m} y_i = m - p$. The pseudo-Boolean formulation can then be linearised by introducing for each product of y-variables in $B_{C,p}(\mathbf{y})$ a nonnegative z-variable and a constraint reflecting the relation between z- and y-variables. The resulting MBpBM formulation can be expressed as follows:

$$f(\mathbf{y}) \quad = \quad \alpha_0 + \sum_{r=1}^{m} \alpha_r y_r + \sum_{r=m+1}^{|B|} \alpha_r z_r \to \min$$

$$\text{s.t.} \qquad \sum_{i=1}^{m} y_i = m - p \tag{3.3}$$

$$z_r \geq \sum_{i \in T_r} y_i - |T_r| + 1, \quad r = m+1, \ldots, |B| \tag{3.4}$$

$$z_r \geq 0, \quad r = m+1, \ldots, |B| \tag{3.5}$$

$$\mathbf{y} \in \{0,1\}^m, \tag{3.6}$$

where α_r are coefficients of $B_{C,p}(\mathbf{y})$, $|B|$ denotes the number of monomials in $B_{C,p}(\mathbf{y})$, and T_r is the set of variable indices in monomial r, i.e. $z_r = \prod_{i \in T_r} y_i$.

The following example demonstrates how our formulation works for a small CF instance.

Example. Let the instance of the cell formation problem be defined by the following machine-part incidence matrix (MPIM)

$$\begin{array}{cc|ccccc|} & & \multicolumn{5}{c}{\textit{parts}} \\ & & 1 & 2 & 3 & 4 & 5 \\ \hline & 1 & & 1 & & 1 & 1 \\ \textit{machines} & 2 & 1 & & 1 & & \\ & 3 & & 1 & & 1 & \\ & 4 & 1 & & 1 & & \\ \hline \end{array} \tag{3.7}$$

with 4 machines and 5 parts (zero entries are skipped for better visualisation). Now one can construct the machine-machine dissimilarity matrix C by applying the defined above dissimilarity measure (3.1):

$$C = \begin{bmatrix} 6 & 20 & 10 & 20 \\ 20 & 9 & 19 & 9 \\ 10 & 19 & 9 & 19 \\ 20 & 9 & 19 & 9 \end{bmatrix} \tag{3.8}$$

For example, the left top entry c_{11} is obtained in the following way:

$$\begin{aligned} c_{11} = \quad & r(r-1) - \sum_{k=1}^{r} \Gamma(a_{1k}, a_{1k}) = \\ & 5(5-1) - \Gamma(0,0) - \Gamma(1,1) - \Gamma(0,0) - \Gamma(1,1) - \Gamma(1,1) = \\ & 20 - 1 - 4 - 1 - 4 - 4 = 6 \end{aligned} \tag{3.9}$$

These dissimilarities can be thought of as some constant (needed to ensure nonnegative values) minus the weighted sum of the number of matching zeros and ones in two corresponding rows of the MPIM. The weights are chosen such that to ensure that any matching one cannot be compensated by any number of matching zeros within a row.

Possible permutation and differences matrices for the costs matrix (3.8) are:

$$\Pi = \begin{bmatrix} 1 & 2 & 3 & 2 \\ 3 & 4 & 1 & 4 \\ 2 & 3 & 2 & 3 \\ 4 & 1 & 4 & 1 \end{bmatrix} \quad \Delta = \begin{bmatrix} 6 & 9 & 9 & 9 \\ 4 & 0 & 1 & 0 \\ 10 & 10 & 10 & 10 \\ 0 & 1 & 0 & 1 \end{bmatrix} \tag{3.10}$$

These lead to the following pseudo-Boolean polynomial $B_C(\mathbf{y})$:

$$B_C(\mathbf{y}) = 33 + 4y_1 + 1y_3 + 20y_1y_3 + 20y_2y_4 + 2y_2y_3y_4 \tag{3.11}$$

If one is interested in having two manufacturing cells then the number of medians p in the formulation should be set to 2 and the pseudo-Boolean polynomial can be truncated to the degree of $(m - p) = 2$:

$$B_{C,p=2}(\mathbf{y}) = 33 + 4y_1 + 1y_3 + 20y_1y_3 + 20y_2y_4 \tag{3.12}$$

The obtained polynomial has two non-linear terms that we have to linearise by introducing additional z-variables: $z_5 = y_1y_3$ and $z_6 = y_2y_4$. Now, our MBpBM formulation allows expressing the given instance of cell formation as the following mixed-integer LP problem:

$$f(\mathbf{y},\mathbf{z}) = 33 + 4y_1 + 1y_3 + 20z_5 + 20z_6 \longrightarrow \min \tag{3.13}$$

$$y_1 + y_2 + y_3 + y_4 = 2 \tag{3.14}$$

$$z_5 \geq y_1 + y_3 - 2 + 1 \tag{3.15}$$

$$z_6 \geq y_2 + y_4 - 2 + 1 \tag{3.16}$$

$$z_i \geq 0,\ i = 5,6 \tag{3.17}$$

$$y_i \in \{0,1\},\ i = 1,\ldots,4 \tag{3.18}$$

Its solution $y = (0, 0, 1, 1)^T$, $z = (0, 0)^T$ leads to the following cells:

$$\begin{array}{cc|ccc|cc|}
 & & \multicolumn{5}{c}{\mathit{parts}} \\
 & & 2 & 4 & 5 & 1 & 3 \\ \hline
 & 1 & 1 & 1 & 1 & & \\
\mathit{machines} & 3 & 1 & 1 & & & \\ \hline
 & 2 & & & & 1 & 1 \\
 & 4 & & & & 1 & 1 \\ \hline
\end{array} \tag{3.19}$$

◁

3.2.2 Compactness of the MBpBM formulation

Taking into account that there is a one-to-one correspondence between nonlinear monomials of $B_{C,p}(\mathbf{y})$ and nonnegative variables and constraints in MBpBM, the properties of $B_{C,p}$ directly apply for the MBpBM formulation.

The fundamental property of the pseudo-Boolean formulation is that for real-world instances the number of monomials in $B_{C,p}(\mathbf{y})$ can be essentially reduced comparatively to the number of entries in the initial costs matrix. In particular, the following three reductions take place:

- only pairwise different elements in each column of the costs matrix play a role;
- all equal column subpermutations in Π contribute to a single monomial in $B_{C,p}(\mathbf{y})$;
- the degree of $B_{C,p}(\mathbf{y})$ is at most $m - p$, i.e. only $m - p + 1$ smallest different entries in each column of the costs matrix are meaningful ("p-truncation").

The cell formation application supports these reductions. Consider, for example, an instance with p perfect cells, i.e. its machine-part incidence matrix can be transformed into an ideal block-diagonal form with p blocks. In this case each column of the corresponding costs matrix for PMP has at most p different entries, which is normally much less then $m - p + 1$. Next, the number of different subpermutations of each length is also equal to p. Thus, in case of p perfect cells the objective has at most $p \times p$ monomials, irrespectively of the number of machines and parts. This results in a MBpBM formulation with at most $p \times (p - 1)$ nonnegative variables and corresponding constraints, irrespectively of the number of machines and parts. Of course, perfect cells are uncommon in practice and the problem becomes larger, however, these considerations demonstrate that the size (and, therefore, complexity) of the model is closely related to the complexity of the instance. It should be noticed that the classical formulation of the PMP (which is most widely used, see, e.g., Won & Lee (2004)) contains all $m \times m$ coefficients in the objective function.

To illustrate this point we performed a number of computational experiments. A 200 × 200 block-diagonal matrix with 5 ideal blocks was generated and then gradually perturbed by adding random *flips* (change 1 within a diagonal block into 0, or 0 outside a block into 1). For each obtained instance we estimated the number of coefficients in the objective of the MBpBM formulation and the solution time. The size of the instance (200 machines) was intentionally chosen larger than normally occur in practice: we could not find instances with more than 50 machines in literature, while the number of parts does not influence the formulation. This was done in order to show that the performance of our model does not deteriorate with an increase in the instance size. The experimental results are presented in Figure 3.3, where the numbers of coefficients in the objective and running times are plotted against the amount of flips, expressed as a percentage of the total number of elements in the input matrix. Only the cases with less than 15% of flips are considered because otherwise the potential intercell movement becomes too large and the CF itself does not make sense. As can be seen from the figure, even for the instances with 200 machines the computing times are normally below 1 second, except rare cases (85 out of 6000) when up to 10 minutes were needed. We believe that these outliers are caused only by the MILP solver due to "incorrect" branching.

Speaking more generally, MBpBM contains all known reductions for PMP not involving (pre)-solving the instance, unlike other MILP formulations for PMP. For example, the formulation from (Elloumi, 2010) does not use p-truncation. On the other hand, it is possible to reduce the size of the MBpBM formulation further by involving estimation of lower bounds on the subspaces of feasible solutions (a possible framework is described in Goldengorin & Krushinsky, 2011a).

3.2.3 A note on optimality of PMP based models

The PMP does not explicitly optimize the goal of cell formation, see (Goldengorin & Krushinsky, 2011b). Thus, it is important to analyse the quality of solutions produced by a PMP based model.

Let us first consider the case of a manufacturing system in which p perfect cells

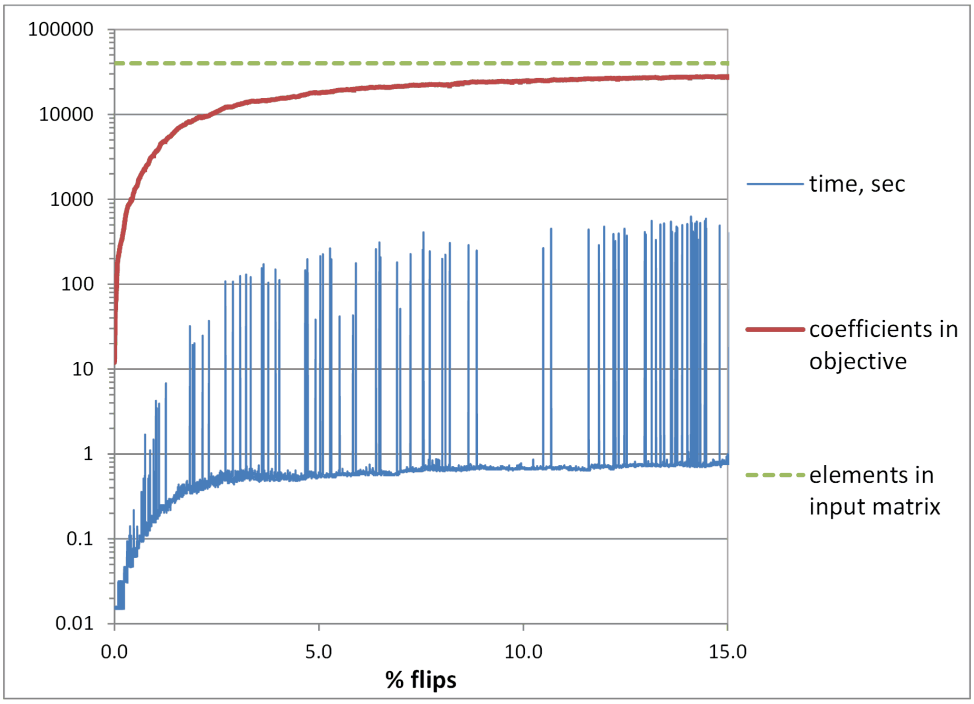


Figure 3.3. Performance of a PMP based model for CF ($m = 200$, $r = 200$, $p = 5$.)

are possible. It is not hard to understand that a PMP model equipped with a reasonable dissimilarity measure (like the one described above (3.1)) will discover those p cells, thus producing optimal results; see (Goldengorin & Krushinsky, 2011b). In practice, however, perfect cellular structure is distorted to some extent. If input data are given in a form of a machine-part incidence matrix then there are two types of distortions: voids – zeroes in diagonal blocks, and exceptions – ones outside the diagonal blocks.

The following propositions provide sufficient conditions for optimality of the obtained solution.

Proposition 3.1. *Suppose that a block structure without exceptions exists (only voids are allowed). In this case a PMP based model produces an optimal solution if in each cell there is at least one machine that is needed for all parts from the corresponding part family.*

Proof. First we prove that only machines needed for all parts in the cells can become medians. Observe that the dissimilarity measure (3.1) is designed in such a way that for any two machines (rows of the machine-part incidence matrix) each coinciding one weighs more than any number of zeros. This implies that only the machine that is needed for all parts assigned to a cell will "cover" the maximum number of ones and will be selected as a median. As soon as medians are defined, all other machines are uniquely assigned to the cells where they are needed – assumption of the proposition implies that a structure where each machine is needed in exactly one cell is possible. As a result, completely independent cells will be obtained. ■

Proposition 3.2. *Suppose that a solution with a block structure without exceptions is found (only voids are allowed). If in each cell the median machine has at least one part in common with any other machine in a cell, then the solution is optimal.*

Proof. Straightforward, as moving any machine to a different cell will create at least one exception. ■

Proposition 3.3. *Suppose, a solution with a block structure without voids is found (only exceptions are allowed). If the number of exceptions in each row is strictly less than the number of within-block ones in this row then the solution is optimal.*

Proof. Absence of voids in the blocks guarantees that the assignment of machines (rows) to cells (blocks) is not sensitive to a particular choice of medians. At the same time, a limited number of exceptions induced by any machine guarantees that its current position is optimal, irrespectively of the configuration of other blocks. This is due to the fact that moving a machine to the other cell will reduce the number of matching ones and this cannot be compensated with any increase in the number of matching zeros, due to the used dissimilarity measure (3.1). ■

Proposition 3.3 can be generalized to allow for both voids and exceptions.

Proposition 3.4. *If there exist an optimal solution to the CF problem satisfying the following requirement, then it will be found by a PMP-based model: for any two machines (rows) i and j belonging to the same cell (block) k the total number of voids in rows i and j is strictly less than the difference between the number of parts (columns) assigned to cell k and the number of exceptions in either of the rows i and j. The inverse is also true: if such a solution is found then it is optimal.*

Proof. The condition insures that any machine (row) has more matching ones with any other machine from the same cell (by the pigeonhole principle) than with a machine from another cell. This guarantees that the assignment of machines (rows) to cells (blocks) is not sensitive to a particular choice of medians. The rest of the reasoning is similar to the proof of Proposition 3.3. ■

As is noticeable in Propositions 3.1–3.3, presence of dense blocks is critical for an optimality unless exceptional elements can be avoided. These propositions assume some properties of the optimal solution, thus they can only be used for posterior assessment of optimality. Yet, as our numerical experiments show, the solution time for our model is very small and it is reasonable to solve the problem and then check the optimality of the obtained solution.

As the conditions of Propositions 3.1–3.3 are not always met, we performed an experimental study on the possible modelling error introduced by the PMP models for CF. We generated input matrices with an ideal block-diagonal structure and then gradually destroyed it by adding (unbiased) random flips (change 1 within a diagonal block into 0, or 0 outside a block into 1). For each instance we compared the performance of the original configuration of the cells and the one discovered by our PMP-based model in terms of the number of exceptions (expressed as a percentage of the total number of ones in the matrix). The latter quantity is exactly the amount of intercell movement. The typical behaviour of a PMP based model is presented in Fig.3.4, where the number of flips is expressed as a percentage of the total number of elements $m \times r$ in the input matrix and each data point is averaged for about 1000 trials. As the figure shows, the average error is quite low and does not exceed 1%. The maximum error in our experiments was also quite limited and did not exceed 10%. Clearly, as the number of flips gets larger, the initial configuration of cells is no more optimal and becomes dominated. It can be easily checked that as the amount of flips approaches 50%, the matrix approaches a completely random one (i.e. each element is 1 with probability 0.5). It is also important to understand that in this case the CF problem itself does not make sense because the underlying system does not possess a cellular structure and cannot be decomposed in a reasonable way. In fact, cellular decomposition of the manufacturing system makes sense in practice only if the resulting amount of intercell movement (exceptions in the block-diagonalized matrix) is below $10 - 15\%$; in these cases the maximum modeling error in our experiments does not exceed 4%, while the average error is of the order $10^{-3}\%$.

Thus our experimental study on the modelling error can be summarized as follows: if a PMP-based model discovers a reasonable solution (with less than 15% intercell movement) then it is very close to the optimal one, otherwise a “good” solution most probably does not exist.

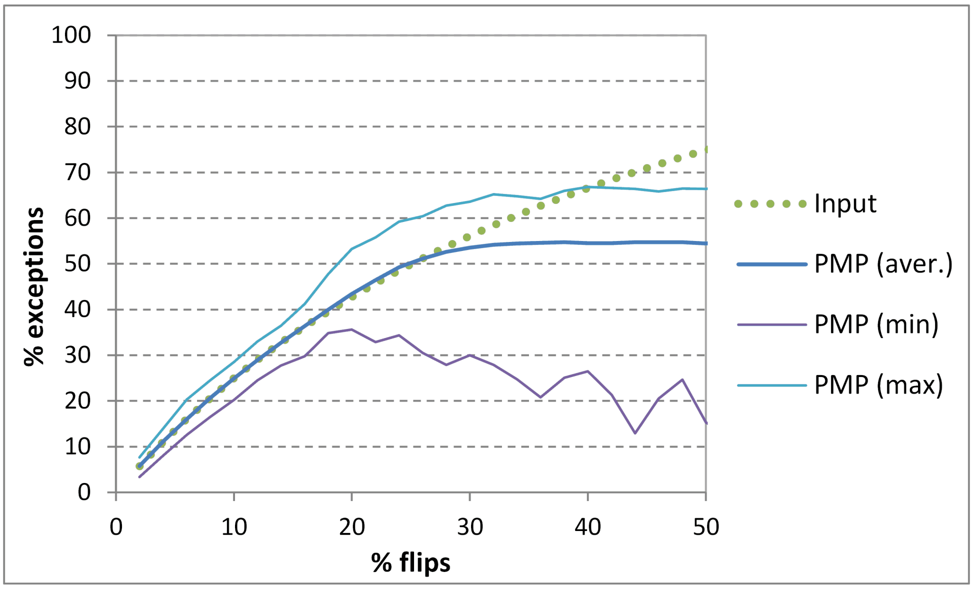


Figure 3.4. Solution quality of a PMP based model for CF ($m = 25$, $r = 50$, $p = 4$, cell sizes vary from 4×9 to 8×15

3.3 Possible Extensions of the Model

In this section we would like to discuss the possibilities of introducing additional real-life factors and constraints into the model. Thus, we are not interested here in describing all constraints that can be incorporated, but rather in demonstrating the possibility of extending the model appropriately.

Clearly, there are three places in our model where additional factors can be incorporated:

- dissimilarity coefficients
- objective function (structure)
- constraints

The use of dissimilarity coefficients can be illustrated, for example, as follows. The availability of skills in a manufacturing system can be represented by a machine-worker skills matrix, i.e. in a way very similar to the input for machine-part grouping. This means that any available machine-machine (dis)similarity measure can be applied to this skills matrix. Being then plugged into a similarity-based cell formation approach such measure minimizes a number of workers that can operate a machine outside of their cell, or, equivalently, maximizes a number of machines that each worker can operate within his cell. Similarities based on either of these data can be combined in a number of ways, e.g. linearly or multiplicatively. The case of a linear combination with equal weighting coefficients is equivalent to having a one aggregated incidence matrix where each column corresponds either to a part or to a worker. It should be mentioned that the same approach is used in Bhatnagar & Saddikuti (2010) for what they call a concurrent model. In that paper it is also demonstrated that such an approach gives better results than two-stage procedures that make cells and assign workers consecutively.

The objective function can be extended, e.g., by penalising assignment of some machines to the same cell. In this way an issue of equivalent machines (the ones with similar functionality) can be resolved. Another example is the use of manufacturing sequences: terms accounting for multiple transits of a part between the cell can be added to the objective.

Finally, a wide variety of linear constraints can be included. These range from simple variable fixing constraints, to capacity, workload balancing and other ones. For example, just by fixing some z-variables one can force or prohibit assignment of some machines to the same cells – this can be necessary because of safety, engineering or managerial considerations.

In the rest of this section we provide examples of extending the model with several particular factors.

3.3.1 Availability of Workforce

The availability of skills in a manufacturing system can be represented by a machine-worker skills matrix, i.e. in a way very similar to the input for machine-part grouping. This means that any available machine-machine (dis)similarity measure can be applied to this skills matrix. Being then plugged into a similarity-based cell formation approach such measure minimizes a number of workers that can operate a machine outside of their cell, or, equivalently, maximizes a number of machines that each worker can operate within his cell. Clearly, cells produced by skills-based clustering can differ from those produced by functional grouping. Thus, the machine-part incidence matrix cannot be simply substituted by skills matrix in the definition of (dis)similarity measure. On the other hand, (dis)similarities based on either of these data ($d_p(i,j)$ and $d_w(i,j)$, respectively) can be combined in a number of ways, e.g. linearly ($d(i,j) = \alpha_1 d_p(i,j) + \alpha_2 d_w(i,j)$) or multiplicatively ($d(i,j) = d_p(i,j) \times d_w(i,j)$). The case of a linear combination with equal weighting coefficients is equivalent to having a one aggregated incidence matrix obtained by simply joining the machine-part incidence matrix and a skills matrix together:

$$\begin{array}{cc|cccc|cccc|}
 & & \multicolumn{4}{c}{\textit{parts}} & \multicolumn{4}{c}{\textit{workers}} \\
 & & 1 & 2 & \dots & r & 1 & 2 & \dots & w \\
\hline
 & 1 & & & & & & & & \\
\textit{machines} & \vdots & & & & & & & & \\
 & m & & & & & & & & \\
\hline
\end{array} \tag{3.20}$$

It should be mentioned that the same approach – joint use of machine-part and skills matrices in definition of (dis)similarity coefficients is used in Bhatnagar & Saddikuti (2010) for what they call a concurrent model. In that paper it is also demonstrated that such an approach gives better results than two-stage procedures that make cells and assign workers consecutively.

3.3.2 Capacity Constraints

The considered above model does not account for the size of cells that it produces and thus can lead to highly unbalanced manufacturing systems (e.g. having cells containing only one machine). This implies that some additional constraints restricting the number of machines in a cell are needed. Suppose we want each cell to contain at least n_L and at most n_U machines. Keeping in mind that our MBpBM model can be augmented by any linear constraints there emerge two major questions:

- can such capacity constraints be expressed in a linear form?

- is the number of these new constraints small enough to ensure acceptable solution times?

Let us consider the first question. By constructing a linear capacity constraint we will show that it has a positive answer. Let us introduce auxiliary Boolean variables x_{ij} such that $x_{ij} = 1$ iff j-th machine is assigned to a cell clustered around i-th machine (or, in terms of PMP, j-th client is served from i-th facility). This can happen only if two conditions are satisfied simultaneously: i-th machine is a center of the cluster and all machines k such that $c_{kj} \leq c_{ij}$ are not centres of the clusters (i.e. for all k s.t. $\pi_{kj} \leq \pi_{ij}$ holds $y_k = 1$). These considerations lead to the following expression for x_{ij}:

$$x_{ij} = (1-y_i)\prod_{k\colon \pi_{kj}\le\pi_{ij}} y_k = \prod_{k\colon \pi_{kj}\le\pi_{ij}} y_k - y_i \prod_{k\colon \pi_{kj}\le\pi_{ij}} y_k \qquad (3.21)$$

If the corresponding entries $\Delta[i,j]$ and $\Delta[i-1,j]$ in the differences matrix are nonzero then our MBpBM formulation contains z-variables that are equal to the products in (3.21)

$$z' = \prod_{k\colon \pi_{kj}\le\pi_{ij}} y_k\,, \qquad z'' = y_i \prod_{k\colon \pi_{kj}\le\pi_{ij}} y_k$$

and x_{ij} can be expressed as $x_{ij} = z' - z''$. Having linear expressions for x_{ij} (in terms of z-variables), capacity constraints can be written as:

$$\sum_{j\in J} x_{ij} \ge n_L \quad \text{and} \quad \sum_{j\in J} x_{ij} \le n_U,\ i \in I$$

Let us now consider the question about the number of additional constraints. Naturally, $2m$ constraints are always needed, however, depending on products in (3.21) something else may be required. Even though some products can be eliminated from (3.21) by p-truncation as they always contain at least one zero variable, there may still be products for which additional z-variables must be introduced, as well as the corresponding constraints of type (3.4). To sum up, the number of additional constraints is equal to m plus number of zero entries in first $(m-p)$ rows of the differences matrix (this number is always less then $m(m-p)$ and is polynomial in the instance size).

The capacitated version of the considered above model for cell formation instance is as follows:

$$\begin{aligned} f(\mathbf{y},\mathbf{z}) \;=\; & 33 + 4y_1 + 1y_3 + 20z_5 + 20z_6 \longrightarrow \min \\ & y_1 + y_2 + y_3 + y_4 = 2 \\ & z_5 \ge y_1 + y_3 - 1 \\ & z_6 \ge y_2 + y_4 - 1 \\ & 1 + y_3 - y_1 - z_5 \ge n_L \\ & 1 + y_3 - y_1 - z_5 \le n_U \\ & 2 + 2z_5 - 2y_2 \ge n_L \\ & 2 + 2z_5 - 2y_2 \le n_U \\ & 1 + y_1 + 2z_6 - y_3 - z_5 \ge n_L \end{aligned}$$

$$1 + y_1 + 2z_6 - y_3 - z_5 \le n_U$$
$$2y_2 - 2y_6 \ge n_L$$
$$2y_2 - 2y_6 \le n_U$$
$$z_i \ge 0,\ i = 5,6$$
$$y_i \in \{0,1\},\ i = 1,\dots,4$$

3.3.3 Workload Balancing

Another type of constraints that can be incorporated into the MBpBM force the obtained cells to have almost equal workload in terms of machine-hours spent by each cell. The workload of a cell is a sum of workloads of all the machines within it. Within any model based on the PMP (including ours) a cell can be indicated by its central machine (a median point in PMP terminology) – a machine that can be considered the most typical representative of its cluster. Now, suppose some machine *i* is the centre of the cluster. Any other machine *j* is assigned to cell containing machine *i* if and only if among all the central machines it minimizes a dissimilarity measure, i.e.

$$i = arg \min_{k \in I, y_k = 0} d(k, j)$$

In other words, machine *j* is assigned to the cell containing machine *i* iff in *j*th column of the permutation matrix the first entry that corresponds to a zero *y*-variable is *i*, i.e. holds $\gamma(j,i) = 1$ where

$$\gamma(j,i) = \prod_{k=1}^{K:\ \pi_{jK}=i} y_{\pi_{jk}} \tag{3.22}$$

Equation (3.22) defines a value that can be used as an indicator for adding or not adding a workload of a particular machine *j* to the total workload of the cell clustered around machine *i*. To introduce workload balancing into the model one has to sum up workloads of all machines multiplied by such indicators and do that for cell clustered around every machine, thus leading to $O(m)$ constraints. I.e., if one wants to bound the workload of all cells from below by some value W_L or from above by W_U then the following constraints are to be added:

$$\sum_{j=1}^{m} [\gamma(j,i) \cdot w(j)] \ge W_L(1 - y_i), \quad i = 1, \dots, m \tag{3.23}$$

$$(1-y_i)\sum_{j=1}^{m}[\gamma(j,i)\cdot w(j)] \leq W_U, \quad i=1,\ldots,m \tag{3.24}$$

where $w(j)$ - a number of hours that are needed to process all parts by machine j. The multiplier $(1-y_i)$ is used in order to cancel the restrictions on the cells that are not actually established. It is straightforward that constraints (3.23)-(3.24) are nonlinear if used as they are given because of the products in $\gamma(j,i)$, however, for most of these products there were defined z-variables that can be substituted into the constraints thus making them linear. We will illustrate these constraints with the considered above numerical example (3.7), assuming for the sake of simplicity that each operation on any machine takes one time unit and $w(j) = \sum_{k=1}^{r} a_{kj}$. The load balancing constraints for machines 1, 2, 3, and 4 will be:

$$1: \quad 3+2y_2y_3y_4+2y_3+2y_2y_3y_4 \geq W_L(1-y_1) \tag{3.25}$$

$$(1-y_1)(3+2y_2y_3y_4+2y_3+2y_2y_3y_4) \leq W_U \tag{3.26}$$

$$2: \quad 2+3y_1y_3+2y_1y_3+2 \geq W_L(1-y_2) \tag{3.27}$$

$$(1-y_2)(2+3y_1y_3+2y_1y_3+2) \leq W_U \tag{3.28}$$

$$3: \quad 2+2y_1+2y_2y_4+2y_2y_4 \geq W_L(1-y_3) \tag{3.29}$$

$$(1-y_3)(2+2y_1+2y_2y_4+2y_2y_4) \leq W_U \tag{3.30}$$

$$4: \quad 2+2y_1y_2y_3+2y_2+2y_1y_2y_3 \geq W_L(1-y_4) \tag{3.31}$$

$$(1-y_4)(2+2y_1y_2y_3+2y_2+2y_1y_2y_3) \leq W_U \tag{3.32}$$

It should be mentioned that these constraints can be subjected to combining similar monomials and p-truncation by observing that each product of more than $(m-p)$ variables is zero in any feasible solution, like it was done for the objective function in (2.13), i.e. for a pseudo-Boolean polynomial. After doing that and replacing all the products by z-variables, constraints (3.25)-(3.32) will become:

$$1: \quad 3+2y_3+W_Ly_1 \geq W_L \tag{3.33}$$

$$3+2y_3-3y_1-2z_5 \leq W_U \tag{3.34}$$

$$2: \quad 4+5z_5+W_Ly_2 \geq W_L \tag{3.35}$$

$$4+5z_5-4y_2 \leq W_U \tag{3.36}$$

$$3: \quad 2+2y_1+4z_6+W_Ly_3 \geq W_L \tag{3.37}$$

$$2+2y_1+4z_6-2y_3-2z_5 \leq W_U \tag{3.38}$$

$$4: \quad 2+2y_2+W_Ly_4 \geq W_L \tag{3.39}$$

$$2 + 2y_2 - 2y_4 - 2z_6 \leq W_U \tag{3.40}$$

and the augmented MBpBM model (3.13) looks like:

$$f(\mathbf{y}, \mathbf{z}) = 33 + 4y_1 + 1y_3 + 20z_5 + 20z_6 \longrightarrow \min \tag{3.41}$$

$$y_1 + y_2 + y_3 + y_4 = 2 \tag{3.42}$$

$$z_5 \geq y_1 + y_3 - 1 \tag{3.43}$$

$$z_6 \geq y_2 + y_4 - 1 \tag{3.44}$$

$$3 + W_L y_1 + 2y_3 \geq W_L \tag{3.45}$$

$$3 - 3y_1 + 2y_3 - 2z_5 \leq W_U \tag{3.46}$$

$$4 + W_L y_2 + 5z_5 \geq W_L \tag{3.47}$$

$$4 - 4y_2 + 5z_5 \leq W_U \tag{3.48}$$

$$2 + 2y_1 + W_L y_3 + 4z_6 \geq W_L \tag{3.49}$$

$$2 + 2y_1 - 2y_3 - 2z_5 + 4z_6 \leq W_U \tag{3.50}$$

$$2 + 2y_2 + W_L y_4 \geq W_L \tag{3.51}$$

$$2 + 2y_2 - 2y_4 - 2z_6 \leq W_U \tag{3.52}$$

$$z_i \geq 0,\ i = 5, 6 \tag{3.53}$$

$$y_i \in \{0, 1\},\ i = 1, \ldots, 4 \tag{3.54}$$

3.3.4 Utilizing Sequences of Operations

Sequences of operations on parts are not taken into account in the classical models of cell formation, while in real world this factor influences optimality of the obtained decomposition into cells. This can be explained by the following considerations. Assume, a perfect cell decomposition is not possible, i.e. it is not possible to exclude all intercellular interactions and some parts have to travel from one cell to another. It should be mentioned that this assumption is very realistic as most of the real manufacturing systems does not possess a perfect cellular structure (see, e.g., Chu & Hayya, 1991). In such a setting there is a difference between parts that start their production process in one cell and end it in another and those parts that start at one cell then move to another and then move back to the cell from which they started. Clearly, in the latter case the intercellular flow is twice as much as that for the former case and the classical model for cell formation can substantially underestimate the real intercellular flows. Even though there exist an acceptable for our model approach that accounts for operational sequences by defining a machine-

machine similarity measure based on them (Suresh et al., 1999), here we propose a different method in order to demonstrate the flexibility of our MBpBM based model. Hence, in order to take into account the impact of operational sequences we propose to penalize the objective function when two machines that are adjacent in the operational sequence of some part are placed in different cells. Such penalty terms for a pair of machines i and j can have the following general form:

$$P(i,j) = (1 - \gamma(i,j)) \sum_{k=1}^{r} V(k)\gamma_k(i,j) \tag{3.55}$$

where $V(k)$ – production volume of part k, $\gamma_k(i,j) \in \{0,1\}$ is 1 iff part k should be processed by machine j immediately after machine i, $\gamma(i,j) \in \{0,1\}$ is 1 iff machines i and j are in the same cell, and summation is done for all parts. The sum in (3.55) is just a constant that can be calculated directly from the input data, while for indicators $\gamma(i,j)$ a linear representation is needed. If we represent each $\gamma(i,j)$ by a Boolean variable u_{ij} then the objective function will have the form

$$f(\mathbf{y},\mathbf{z},\mathbf{u}) = f(\mathbf{y},\mathbf{z}) + \sum_{i,j=1}^{m} (1 - u_{ij}) \sum_{k=1}^{r} V(k)\gamma_k(i,j)$$

and some constraints are needed to force new variables u_{ij} to take value 1 iff machines i and j are placed into the same cell. In a general form u-variables can be defined by the following Boolean expression:

$$\begin{array}{rlll} & \langle y_1 = 0 & AND & \textit{machines i and j are clustered around machine } 1\rangle \\ OR & \langle y_2 = 0 & AND & \textit{machines i and j are clustered around machine } 2\rangle \\ OR & \langle y_3 = 0 & AND & \textit{machines i and j are clustered around machine } 3\rangle \\ OR & \langle y_4 = 0 & AND & \textit{machines i and j are clustered around machine } 4\rangle \end{array}$$

Now, it is enough to find a set of linear constraints that represent the above Boolean expressions. As in general case such a representation leads to extensive notations, we will use a small example to derive it. Suppose, in the considered above manufacturing system (3.7) the operational sequences for the five parts are given as:

$$\begin{aligned} 1:&\quad 2 \rightarrow 4\\ 2:&\quad 1 \rightarrow 3\\ 3:&\quad 4 \rightarrow 2\\ 4:&\quad 1 \rightarrow 3\\ 5:&\quad 1 \end{aligned}$$

Assuming unit production volumes of all parts for simplicity, there are only three nonzero penalties $P(i,j)$:

$$\begin{aligned} P(1,3) &= (1-\gamma(1,3))\cdot 2\\ P(2,4) &= (1-\gamma(2,4))\cdot 1\\ P(4,2) &= (1-\gamma(4,2))\cdot 1 \end{aligned}$$

We can see that $P(4,2) = P(2,4)$ in this case by comparing their expressions and recalling that indicator $\gamma(i,j)$ is symmetric, i.e. $\gamma(2,4) = \gamma(4,2)$. Let us consider the corresponding u-variables:

$$\begin{aligned} u_{13} &= (\bar{y}_1 \wedge 1 \wedge y_3) \vee (\bar{y}_2 \wedge y_1y_3 \wedge y_1y_3) \vee (\bar{y}_3 \wedge y_1 \wedge 1) \vee (\bar{y}_4 \wedge y_1y_2y_3 \wedge y_1y_2y_3)\\ u_{24} &= (\bar{y}_1 \wedge y_2y_3y_4) \vee (\bar{y}_2 \wedge 1 \wedge 1) \vee (\bar{y}_3 \wedge y_2y_4 \wedge y_2y_4) \vee (\bar{y}_4 \wedge y_2 \wedge y_2)\\ u_{42} &= u_{24} \end{aligned}$$

The last equality holds because both u_{42} and u_{24} indicate that machines 2 and 4 are placed into the same cell. Observing that any product of more than $(m-p)$ variables is 0 for any feasible solution and applying elementary transformations one can get:

$$\begin{aligned} u_{13} &= (\bar{y}_1 \wedge y_3) \vee (\bar{y}_2 \wedge y_1y_3) \vee (\bar{y}_3 \wedge y_1)\\ u_{24} &= (\bar{y}_2) \vee (\bar{y}_3 \wedge y_2y_4) \vee (\bar{y}_4 \wedge y_2)\\ u_{42} &= u_{24} \end{aligned}$$

As we have a minimization problem with an objective function that contains u-variables with negative coefficients, these variables will be maximized and disjunction of n variables x_i can be represented by the constraints

$$\begin{array}{lcl} u = \bigvee\limits_{i=1}^{n} x_i & \Leftrightarrow & u \leq \sum\limits_{i=1}^{n} x_i \\ u \rightarrow \max & & u \leq 1 \end{array}$$

and nonnegativity of u-variables is sufficient, i.e. no new Boolean variables are introduced. Conjunctions can be replaced by products and, in turn, by z-variables. After all substitutions and transformations the augmented model (3.13) is (we defined $u_7 = u_{13}$, $u_8 = u_{24} = u_{42}$):

$$\begin{aligned} f(\mathbf{y}, \mathbf{z}, \mathbf{u}) &= 37 + 4y_1 + 1y_3 + 20z_5 + 20z_6 - 2u_7 - 2u_8 \rightarrow \min && (3.56) \\ & y_1 + y_2 + y_3 + y_4 = 2 && (3.57) \\ & z_5 \geq y_1 + y_3 - 1 && (3.58) \\ & z_6 \geq y_2 + y_4 - 1 && (3.59) \\ & u_7 \leq y_1 + y_3 - z_5 && (3.60) \\ & u_8 \leq 1 && (3.61) \\ & u_i \leq 1,\ i = 7, 8 && (3.62) \\ & u_i \geq 0,\ i = 7, 8 && (3.63) \\ & z_i \geq 0,\ i = 5, 6 && (3.64) \\ & y_i \in \{0, 1\},\ i = 1, \ldots, 4 && (3.65) \end{aligned}$$

We would like to conclude the section by saying that the reductions that make our model efficient are based exclusively on the properties of the underlying clustering model and assume nothing about its further extension. This implies that any additional constraints expressed in a linear form can be added to our compact formulation.

3.4 Experimental Results

The aim of our numerical experiments was twofold. First, we would like to show that the model based on PMP produces high-quality cells and in most cases outperforms other contemporary approaches, thus making their use questionable. Second, by showing that computation times are negligibly small, we argue the use of heuristics for solving PMP itself.

Up to this point one basic notion remained undefined in this paper – the quality measure of the obtained decomposition into cells. We used two most widely used measures so as to ensure consistent comparison of results. The first one, the group capability index (GCI) proposed by Hsu (1990) can be calculated as follows:

$$GCI = 1 - \frac{number\,of\,exceptional\,elements}{total\,number\,of\,ones} \quad \times 100\% \tag{3.66}$$

where *exceptional elements* are those nonzero entries of the block-diagonalized machine-parts coincidence matrix that lie outside of the blocks and the *total number of ones* is the total number of nonzero entries in the machine-parts incidence matrix. It should be mentioned that this measure does not account for zeroes inside the blocks, i.e. does not take into account density of intracell flows. The second quality measure, group efficiency (η), was proposed by Chandrasekharan & Rajagopalan (1986a) and is a weighted sum of two factors η_1 and η_2:

$$\eta = \omega\eta_1 + (1-\omega)\eta_2 \quad \times 100\%\,,\ 0 \le \omega \le 1\,. \tag{3.67}$$

In turn, η_1 and η_2 are expressed as:

$$\eta_1 = \frac{o-e}{o-e+v}$$
$$\eta_2 = \frac{mr-o-v}{mr-o-v+e}$$

where m – number of machines, r – number of parts, o – number of ones in the part-machine matrix, e – number of exceptional elements, v – number of zeroes in diagonal blocks. The weighting factor ω is usually set to 0.5 and we used this value.

For the considered above instance (3.19) these performance measures have the following values: $GCI = 100\%$, $\eta = 0.5 \cdot (\frac{9-0}{9-0+1} + \frac{20-9-1}{20-9-1+0}) \times 100\% = 95\%$. It should be mentioned that the sum of voids and exceptions $(v+e)$ sometimes is used as a performance measure (see, e.g., Bhatnagar & Saddikuti, 2010).

Taking into account the aim of our experiments, we compared our results with those reported in four recent papers and (Chen & Heragu, 1999). The main focus was made on the largest instances. The first paper is by Won & Lee (2004) and, like us, uses a p-Median approach but solves PMP by a heuristic procedure. They use Wei and Kern's (Wei & Kern, 1989) similarity measure and GCI (3.66) as a quality measure. We were not able to derive the value of η because solutions are not provided in their paper. The second paper by Yang & Yang (2008) applies the ART1 neural network to cell formation, thus using a completely different approach. The authors used η-measure (3.67) to estimate solution quality and included solutions (block-diagonalized matrices) in their paper, thus making it possible for us to compute GCI and to fill in the gaps in the following Table 3.1 that summarizes results of our comparative experiments. The third paper is by Ahi et al. (2009) and demonstrates an application of a decision-making technique (TOPSIS) to the cell formation problem. Authors report values of group efficiency η and we derived values of GCI from their solutions.

Table 3.1 contains data on computational experiments with instances used in the three mentioned above papers: Won & Lee (2004), Yang & Yang (2008) and Ahi et al. (2009). Column "source" indicates the source of the cell formation instance and of the performance data. Next two columns contain information on the size of input, such as the number of machines m, the number of parts r, and the number of cells to be made p. The last four columns indicate quality of solutions (in terms of GCI and η) reported in the discussed papers and obtained by us, correspondingly.

As can be seen from Table 3.1, in most of the considered cases our results outperform those reported in literature by up to $85.43\% - 68.02\% \approx 17\%$ (see second to the last row in Table 3.1). On the other hand, there exist scarce instances for which our model gives is dominated by other heuristics. This can be explained by the fact that even though we solve PMP to optimality, the p-Median problem itself is not explicitly an exact model to optimize any of the used above quality measures of cell decomposition (their appropriateness can also be debated). Consequently, any model based on the p-Median problem is of a heuristic nature. However, unlike most of the other heuristics it grasps the clustering nature of cell formation and presents a flexible framework by allowing additional constraints reflecting real world manufacturing systems to be introduced. Such flexibility is inherent, in particular, to mathematical programming approaches, but in contrast to them, for PMP we have found an efficient formulation (see Section 3.2.1).

The fourth and the most recent paper considered in our computational experiments is by Bhatnagar & Saddikuti (2010). It uses a model that is very similar to the p-Median problem but differs in the following detail: a restriction specifying the number of cells is replaced by a constraint ensuring that each cell has at least two machines. To our opinion, this model has a potential drawback as it tends to split "reasonable" cells as can be seen from its objective function (taking into account that for similarities holds $s(i,i) \geq s(i,j)$ for any two machines i, j). We implemented the models for machine cell formation and part assignment from Bhatnagar & Saddikuti (2010) in Xpress and performed a number of experiments with the largest available in literature instances. Like in the previous cases we used only machine-part incidence matrices as an input and Wei and Kern's (dis)similarity measure. Taking into account that the model from (Bhatnagar & Saddikuti, 2010) automatically defines the best number of cells, we had to solve our PMP based model for all possible values of p and pick the best results.

Finally, we compared performance of our model and the one from Chen & Heragu (1999), which we implemented in Xpress. As this model, like ours, does not define the optimal number of cells, we tried to solve it for all possible values of p. However, this was not always possible due to the complexity of the model. We also limited the running time of the model by 10 hours and provide the best results that we could obtain.

The results for our model and the ones from Bhatnagar & Saddikuti (2010) and Chen & Heragu (1999) are summarized in Table 3.2, where the first column enumerates the test instances, the second one refers to the original source of the instance and the next column shows the number of machines and parts. The following six columns report the quality of solutions obtained by the three models. The last column indicates the time (in seconds) spent by the model from Chen & Heragu (1999) (note, that for our model it took about 1 second to solve either of the instances). As can be seen from Table 3.2, our model considerably outperforms the model from Bhatnagar & Saddikuti (2010). Also, in most of the cases our model outperforms the one from Chen & Heragu (1999) in terms of the two used performance measures. Moreover, in terms of computing times the difference is clear. This can be explained by the adaptability of our model to the input data. Consider, for example, instance 7 from Table 3.2 that has a perfect cellular structure with 7 cells. For this instance our MBpBM formulation has 42 variables, 19 constraints and 19 coefficients in the objective, while for the model from Chen & Heragu (1999) these numbers are 7168, 6851 and 20224. At the same time, for instances with perfect cells our model provides provably optimal solutions!

Also, we would like to mention that our results strongly outperform those mentioned in Doulabi et al. (2009).

Concerning the solution times of our PMP based model, each of the considered instances was solved within one second on a PC with 2.3GHz Intel processor, 2GB RAM and Xpress as a MILP solver. In our opinion, even if some heuristic can be faster, then the difference in computing times is negligibly small.

3.5 Summary and Future Research Directions

There is a tendency in the literature for the cell formation models to become more and more complicated. Such complication has two negative side effects. First of all, the sophisticated structure of the model usually prohibits its extension to additional factors and/or constraints taking place in real manufacturing systems. Secondly, a complicated model that was designed in order to improve the quality of the obtained solutions usually raises a problem of computational intractability. This forces the use of heuristics for solving not the initial cell formation problem but the model of it. Suboptimality of these heuristics can overwhelm the advantages of the model, making them questionable.

In this chapter we showed that these negative side effects can be avoided by presenting an efficient reformulation of the p-Median problem. Our reformulation is flexible enough to accept additional real-life constraints, like capacities, operational sequences, etc. At the same time, the computational experiments show that our model is computationally efficient and can be solved to optimality within one second on a standard PC by means of general purpose software, like CPLEX or Xpress. For the computational experiments we picked instances from four recent papers in the field and showed that the PMP-based model outperforms contemporary heuristics. We did not perform a thorough comparison of computation times as for our model it took less than one second to solve each of the considered problems. It should be also mentioned that the main part of our model can be solved by a general-purpose MILP solver and transformations with a pseudo-Boolean polynomial use only basic algebraic operations. This means that an implementation of our model does not require extensive additional efforts. A comparison with an exact approach was also performed and showed that in most cases our model provides better solutions (in terms of the widely used quality measures) while having incomparably smaller running time. Finally, by means of computational experiments we showed that the modelling error of a PMP based model is quite limited with an average of 1% and solution times stay within 1 second in 99% cases even for instances with 200 machines, i.e. much larger than those occurring in practice.

In the numerical experiments we considered the simplest possible approach to cell formation aimed at functional grouping of the machines (equivalently, at block-diagonalising the machine-part incidence matrix) without taking into account additional factors taking place in real manufacturing systems. There are two reasons for this. First, we wanted to demonstrate that even a computationally intractable model of cell formation (at least in its simplest form) can be solved to optimality, and this possibility, to the best of our knowledge, was overlooked in literature. Second, this choice was partially governed by available recent papers in the field with which we wanted to compare our results. At the same time, we showed that a wide range of constraints can be incorporated into the PMP based cell formation model thus making it more realistic and allowing to use all the available information about the manufacturing system.

Taking into account that the current trend is towards introducing into CF models additional real-world factors, the possible future research direction is to incorporate additional constraints into our model, such as availability of several machines of same type, alternative operational sequences, setup and processing times, etc. As our MBpBM formulation is optimal in the number of coefficients in the objective function and the number of linear constraints, insertion of new (linear) constraints, in our opinion, will preserve its tractability and will make it possible to create a flexible and efficient model for cell formation based on the p-Median problem. The issue of efficiency (low computing times) is getting importance from the perspective of Virtual Cell Manufacturing (Slomp et al., 2005) with its Virtual Cell Formation (VCF), a paradigm that becomes more and more promising nowadays. At the same time, our computational results show that at least in case of uncapacitated functional grouping our fast model is a feasible candidate for VCF.

To summarize, all ideas and attempts of extending the decision making for cell formation in group technology based on the classical p-Median model might be revised and essentially improved by using our MBpBM reformulation and adding practically motivated additional constraints reflecting the specific manufacturing environment. Thus, we would like to stress the importance of the model choice and to conclude by saying that the above considerations about the problem complexity and appropriateness of heuristics can be valid also for other applied operations research problems (especially for those that can be modelled via the PMP).

Table 3.1. Experimental comparison with Won & Lee (2004), Yang & Yang (2008) Ahi et al. (2009) .

source	$m \times r$	p	GCI	GCI$_{our}$	η	η_{our}
Won & Lee (2004)	30×41	3	92.2	95.3		59.38
		4	93.0	93.0		64.39
		5	91.4	91.4		72.14
		6	89.8	90.6		75.25
		7	81.3	89.8		77.93
	30×50	3	77.2	77.3		59.53
		4	74.9	76.1		62.14
	30×90	3	79.9	77.5		61.00
	40×100	2	79.5	93.6		55.61
		3	93.1	91.5		59.59
		4	89.8	88.8		63.84
		5	89.3	87.4		69.33
		6	89.3	88.1		75.77
		7	87.6	88.6		81.38
		8	85.5	89.1		85.66
	50×150	2	96.5	96.5		57.49
		3	86.4	90.1		62.63
		4	88.4	92.7		69.05
		5	89.7	91.5		76.44
		6	87.3	93.1		81.89
Yang & Yang (2008)	28×35	6	73.7	73.7	90.68	90.74
	46×105	7	84.1	84.9	87.54	87.57
Ahi et al. (2009)	8×20	3	83.6	83.6	92.11	98.08
	12×19	4	66.2	66.2	80.10	77.09
	20×20	6	67.1	82.3	87.89	90.11
	18×35	4	77.2	77.2	74.10	81.26
	25×40	7	61.5	76.2	68.02	85.43
	20×51	6	67.8	77.2	82.62	82.07

Table 3.2. Experimental comparison of our model and that from Bhatnagar & Saddikuti (2010) [BS10] and Chen & Heragu (1999) [CH99].

			$(e+v)$			$\eta, \%$			time, s
#	source	$m \times r$	[BS10]	[CH99]	our	[BS10]	[CH99]	our	[CH99]
1	Sandbothe (1998)*	20×10	16	9	11	95.40	94.29	95.93	2
2	Ahi et al. (2009)	20×20	34	26	26	92.62	90.70	93.85	2385
3	Mosier, Taube (1985)*	20×20	79	74	77	85.63	79.51	88.71	36000
4	Boe, Cheng (1991)*	20×35	87	77	83	88.31	84.44	88.05	24724
5	Carrie (1973)*	20×35	46	40	41	90.76	88.93	95.64	110
6	Ahi et al. (2009)	20×51	111	96	83	87.86	83.18	94.11	36000
7	Chandrasekharan, Rajagopalan (1989)*	24×40	20	0	0	98.82	100.00	100.00	2
8	Chandrasekharan, Rajagopalan (1989)*	24×40	37	21	21	95.33	95.20	97.48	1363
9	Chandrasekharan, Rajagopalan (1989)*	24×40	55	40	39	93.78	91.16	96.36	29009
10	Chandrasekharan, Rajagopalan (1989)*	24×40	86	122	81	87.92	74.38	94.32	14890
11	Chandrasekharan, Rajagopalan (1989)*	24×40	96	112	89	84.95	77.68	94.21	10968
12	Chandrasekharan, Rajagopalan (1989)*	24×40	94	118	89	85.06	75.29	92.32	16906
13	Nair, Narendran (1996)*	24×40	40	194	25	96.44	69.90	97.39	36000
14	Nair, Narendran (1996)*	24×40	39	27	26	92.35	92.27	95.74	3575
15	Nair, Narendran (1996)*	24×40	60	50	50	93.25	90.56	95.70	36000
16	Nair, Narendran (1996)*	24×40	59	109	50	91.11	78.08	96.40	36000
17	Ahi et al. (2009)	25×40	59	63	56	91.09	86.00	95.52	36000
18	Yang & Yang (2008)	28×35	108	72	71	93.43	91.21	93.82	36000
19	Kumar, Vanelli (1987)*	30×41	63	61	54	90.66	86.78	97.22	16967
20	Stanfel (1985)*	30×50	99	115	93	88.17	81.58	96.48	36000
21	King, Nakornchai (1982)*	30×90	228	202	206	83.18	83.25	94.62	36000
22	Chandrasekharan, Rajagopalan (1987)*	40×100	136	72	72	94.75	95.91	95.91	36000
23	Yang & Yang (2008)	46×105	376	268	271	90.98	87.12	95.20	36000
24	Zolfaghari, Liang (1997)*	50×150	544	502	470	93.05	82.00	92.92	36000

* – a reference to the original source of the instance can be found in [BS10]

Chapter 4

The minimum multicut problem and an exact model for cell formation

4.1 Introduction

Cell formation (CF) is a key step in the implementation of group technology – a concept in industrial engineering developed by Mitrofanov (1966) and Burbidge (1961), suggesting that similar things should be processed in a similar way. In the most general setting, the (unconstrained) CF problem can be formulated as follows. Given finite sets of machines and parts that must be processed within a certain time period, the objective is to group machines into manufacturing cells (hence the name of the problem) so that each part is processed mainly within one cell. This objective can be reformulated as minimization of what is usually referred to as the amount of *intercell movement* – the flow of parts travelling between the cells. This amount can be expressed via the number of parts, their total volume or mass, depending on the particular motivation for CF. For example, if cells are spatially distributed it may become important to reduce transportation costs that depend on the mass or volume rather than on the number of parts.

Throughout the decades the problem has gained a lot of attention resulting in

This chapter is based on the paper (Krushinsky & Goldengorin, 2012).

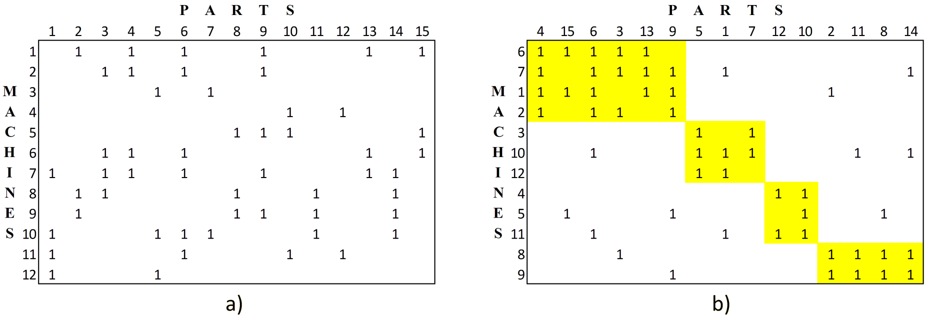
hundreds of papers and dozens of approaches that use all the variety of tools ranging from intuitive iterative methods (e.g., McCormick et al., 1972; King, 1980; Wei & Kern, 1989) to neural networks (e.g., Kaparthi & Suresh, 1992; Yang & Yang, 2008), evolutionary algorithms (e.g., Adil & Rajamani, 2000; Filho & Tiberti, 2006) and mixed-integer programming (e.g., Chen & Heragu, 1999; Bhatnagar & Saddikuti, 2010); an overview can be found in Selim et al. (1998). Despite all this variety, to the best of our knowledge, there is no tractable approach that explicitly minimises the intercell movement. In particular, all the available approaches have at least one of the following drawbacks:

- the model itself is an approximation to the original CF problem;
- the model is solved by a heuristic procedure.

To illustrate the first point we would like to mention that it is a common practice to reduce the size of the problem by considering only relations between machines instead of considering machine-part relations. Such a framework is quite beneficial due to the fact that the number of machines is quite limited (usually less then 100) while the number of parts can be magnitudes larger. This point will be clearly illustrated below by means of an industrial example. The reduction is usually implemented by introducing a machine-machine similarity measure that can be based on the similarity of sets of parts that are being processed by a pair of machines, on similarity of manufacturing sequences of these parts, etc. Literature reports several similarity measures, an overview can be found in Yin & Yasuda (2006). However, all of them are based on intuitive considerations and there is no strict reasoning why one of them is better than another. If such an inexact similarity measure is further plugged into some model, then the whole model is nothing more than an approximation to the CF problem. Finally, the resulting model often appears to be NP-hard and its authors are forced to use heuristic solution methods further deteriorating the solution quality.

The purpose of this chapter is to formulate an exact model for the CF problem, flexible enough to allow additional practically motivated constraints and solvable in acceptable time at least for moderately sized realistic instances.

The chapter is organized as follows. In the next section we discuss the exact model for cell formation, show that it is equivalent to the minimum multicut problem and discuss its computational complexity. In Sections 4.3 and 4.4 we motivate and present two MILP formulations for the problem. Section 4.5 is focused on additional constraints that may be introduced into the model, while Section 4.6


P A R T S
M A C H I N E S

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

a)

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

b)

Figure 4.1. An example of a machine-part incidence matrix (MPIM): (a) raw data; (b) block-diagonalized form (blocks are highlighted). Zero entries are not shown for clarity

provides results of experiments with real manufacturing data. Section 4.7 summarizes the chapter with a brief discussion of the obtained results and further research directions.

4.2 The essence of the cell formation problem

In this section we formalize the CF problem given two types of the input data and show how it can be modelled via the minimum multicut problem. For the rest of this chapter let sets $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, r\}$ enumerate machines and parts, respectively, and let p denote the number of cells.

Quite often, the input data for the CF problem is given by an $m \times r$ binary machine-part incidence matrix (MPIM) $\mathbf{A} = [a_{ij}]$ where $a_{ij} = 1$ only if part j needs among others machine i, see Fig.4.1a. Given such an input, the problem is equivalent (see, e.g., Burbidge (1991)) to finding independent permutations of rows and columns that turn $\mathbf{A}$ to an (almost) block-diagonal form or, equivalently, minimize the number of out-of-block ones, also known as *exceptional elements*. The diagonal blocks correspond to cells, and the number of exceptional elements reflects the amount of intercell movement, see Fig.4.1b.

Given such an interpretation, the problem is similar to the biclustering problem (Madeira & Oliveira, 2004, see, e.g.,). Though for the general biclustering problem there exist efficient exact methods (see, e.g., DiMaggio et al., 2008), they are hardly applicable to CF because most of them allow each row or column to belong to more than one cluster (see, e.g., Madeira & Oliveira, 2004, p. 41), while for CF the issue of non-overlapping blocks is critical. In addition, as we show further in this section, block-diagonalisation does not exactly minimise the intercell movement as it ignores operational sequences.

Though the well known block-diagonal interpretation is easy to perceive, we will consider the problem from a completely different, yet insightful, viewpoint. Without any loss of generality one can associate with matrix **A** an undirected bipartite graph $G(I \cup J, E)$ by simply treating **A** as an incidence matrix of G. (Note, that such an interpretation was also considered for the biclustering problem.) Taking into account that each nonzero element of **A** corresponds to an edge in G, it is not hard to understand that diagonal blocks of **A** correspond to disjoint nonempty subgraphs $G_1, \ldots, G_p$ of G. Consider now the set of edges E' corresponding to exceptional elements and observe that each edge from E' has its endpoints in different subgraphs G_i , $i \in \{1, \ldots, p\}$. Thus, E' can be thought of as a cut that splits G into p nonempty subgraphs. Further, we call a cut with this property a *p-cut*. Assuming that all edges of G have a unit weight and taking into account the relation between E' and exceptional elements, it is possible to reformulate the CF problem in terms of graphs as follows: given an undirected weighted graph find a p-cut of the minimum weight. Let us abbreviate this problem as MINpCUT, in literature it is also known as "min k-cut" (we prefer to denote the number of subgraphs by p as letter k is handy as an index).

One may notice that the MINpCUT based approach has a negative feature as compared to many other models. Instead of using machine-machine relations it works directly with machine-part data, i.e. a MINpCUT instance can be very large (G may have thousands of vertices) and there is no straightforward way to overcome this. However, we argue that this impossibility of reducing the problem size is induced by the "inadequate" format of input data rather than by the model itself. Indeed, irrespectively of the solution approach, the MPIM does not contain enough information to correctly handle the following aspects:

- distinguish between the following two cases:
  (a) a part is processed in one cell and then finished in the second cell;
  (b) a part is processed in one cell, then in the second cell and then again in the first one;
- a part visits some machines several times, i.e. its manufacturing sequence looks like . . .–M1–M2–M1–M2–. . . (this may correspond, for example, to cycles of thermal processing).

Thus, all approaches using the machine-part incidence matrix as an input (e.g., the one from Chen & Heragu (1999)) solve only approximation of the original problem, quite oftenly by a heuristic. In addition, the common practice of deriving machine-machine relations from a MPIM looks somewhat awkward from the methodological point of view. It seems more logical to derive these relations directly from the manufacturing data normally containing more information, e.g., the sequence in which machines are visited by each part.

The mentioned above considerations motivated us to reconsider the essence of the cell formation problem. As mentioned in the Introduction, the objective is to minimize the parts flow between cells. The latter quantity is nothing else than the parts flow between two machines summed up for all pairs of machines belonging to different cells. In terms of graphs this can be expressed as follows. Consider a weighted graph $G(I, E)$, where each vertex corresponds to a machine. An edge $(i, j) \in E$ is assigned a weight equal to the amount of parts going directly from machine i to j and in the opposite direction. Clearly, a p-cut in such a graph produces p machine cells and its weight is exactly equal to the amount of inter-cell movement that must be minimized. In particular, this means that an exact machine-machine similarity measure must be defined as the amount of parts travelling directly between a pair of machines. Once the machine cells are generated, part families can be compiled by assigning each part to a machine cell performing most operations on it. Thus, we again end up with the MINpCUT problem, but now it is defined on a graph that has only m vertices, as compared to $m + r$ vertices in case of input data given by a machine-part incidence matrix. We would like to mention that somewhat similar considerations about the graph-theoretic origins of the exact model for CF can be found in Boulif & Atif (2006). However, these authors do not mention its relation to the min multicut problem, nor provide evidence of tractability for their approach. Another graph-related approaches to CF include those based on the minimum spanning tree (MST; see, e.g., Ng, 1993) and the p-median (PMP; see, e.g., Won & Lee, 2004) problems. Their difference from our approach can be made clearer by observing that by minimising a p-cut one maximises the total weight of edges within p subgraphs. Instead of optimising all weights within subgraphs, MST and PMP-based approaches consider only those falling within a certain pattern: a spanning tree or a tree of depth 1, respectively.

Once we know that the cell formation problem is equivalent to MINpCUT, we may analyse its complexity based on the properties of the latter. First of all, consider the case $p = 2$. MIN2CUT is a straightforward generalization of the well-known min $s - t$ cut problem where optimization is to be done for all pairs (s, t). A closer view makes it possible to conclude that for a graph $G(V, E)$ it is enough to solve $|V| - 1$ min $s - t$ cut instances. As the minimum 2-cut (as well as any 2-cut) splits G into 2 subgraphs, one can fix s lying in one of them and iterate through all possible vertices t until the one lying in the other subgraph is found. Thus, in case of two cells the CF problem without additional constraints is polynomially solvable. On the other hand, as p gets close to $|V|$ the problem becomes easy as well. For example, if $p = |V| - 1$ there is exactly one pair of vertices that must be placed in one subgraph (other $p - 1$ subgraphs are just singletons). Further, if $p = |V| - 2$ there are either two pairs or one triple of vertices that must not be disconnected by a cut. This intuition can be extended further and it becomes clear that the combinatorial complexity of the problem quickly increases as p tends to $|V|/2$.

In a general case when p is a part of the input the problem is NP-hard, having a polynomial complexity $O(n^{p^2}T(n))$ for fixed p (Goldschmidt & Hochbaum, 1994); $T(n)$ denotes time for solving one min $s - t$ cut problem[1] for a graph with n vertices. For a particular case $p = 3$ there also exists an efficient $O(mn^3)$ algorithm by Burlet & Goldschmidt (1997), where n and m are numbers of vertices and edges, respectively. A number of approximate algorithms are known (see, e.g., Saran & Vazirani (1995); Ravi & Sinha (2008)) with the best approximation ratio being $(2 - 2/p)$ (Saran & Vazirani, 1995).

Thus, for $p = 2, 3$ and $|V| - 2, |V| - 1$ the MINpCUT problem (therefore, the unconstrained CF problem) can be efficiently solved even for large instances, while becoming computationally intractable as p gets closer to $|V|/2$. Moreover, most papers on MINpCUT propose specialized algorithms, not allowing additional constraints to be involved and thus inapplicable to CF. This lack of flexible approaches motivated us to develop MILP formulations that can be extended by any linear constraints and solved using a general purpose solver (at least, for moderately sized instances).

4.3 MINpCUT: a straightforward formulation (SF)

In this section we present and discuss a straightforward formulation (SF) of MINpCUT problem that will be further used in the numerical experiments. Let $G(V, E)$ be an undirected weighted graph with $|V| = n$ vertices, let c_{ij} denote the weight of edge $(i, j) \in E$ and define constant S as a sum of all edge weights. Through-

[1] This is equivalent to the maximum flow problem with source s and sink t.

out this paper, let indices i and j enumerate vertices, $i, j \in \{1, \dots, n\}$, and index k enumerate subgraphs, $k \in \{1, \dots, p\}$. SF uses two sets of variables: v_{ik} variables reflecting assignment of vertices to subgraphs and z_{ijk} variables reflecting assignment of pairs of vertices i and j to subgraphs k. Under the introduced notations the SF formulation can be written as follows:

$$S - \sum_{i} \sum_{j>i} \sum_{k} c_{ij} z_{ijk} \longrightarrow \min \tag{4.1}$$

$$\sum_{i} v_{ik} \geq 1 \quad \forall k \tag{4.2}$$

$$\sum_{k} v_{ik} = 1 \quad \forall i \tag{4.3}$$

$$z_{ijk} \leq v_{ik} \quad \forall i \neq j, k \tag{4.4}$$

$$z_{ijk} \leq v_{jk} \quad \forall i \neq j, k \tag{4.5}$$

$$z_{ijk} \geq v_{ik} + v_{jk} - 1 \quad \forall i \neq j, k \tag{4.6}$$

$$v_{ik} \in \{0, 1\} \quad \forall i, k \tag{4.7}$$

$$z_{ijk} \in [0, \infty) \quad \forall i \neq j, k \,. \tag{4.8}$$

The objective (4.1) minimizes the difference between the sum of all edge weights and the sum of weights of the edges within subgraphs, i.e. the weight of the p-cut. Constraints (4.2) ensure that each subgraph has at least one vertex, i.e. there are exactly p nonempty subgraphs. Constraints (4.3) ensure that each vertex is included into exactly one subgraph. Finally, constraints (4.4)–(4.6) are needed to guarantee that a pair of vertices i and j are assigned to the subgraph k if and only if each of them is assigned to subgraph k. The formulation uses $n \times p$ Boolean v-variables, while for z-variables nonnegativity is sufficient as constraints (4.4)–(4.6) force them to take Boolean values.

It is easy to see that the formulation SF has the following property: the number of variables and, therefore, complexity increases with increasing p. Though for small p the formulation is rather efficient (as will be shown in Section 4.6) for larger values of p it becomes intractable. It should be noted that SF does not reflect the fundamental property of the problem: tractability for both small and large (close to n) values of p. This observation motivated us to develop an alternative formulation that will reflect its complexity more adequately.

4.4 MINpCUT: an alternative formulation (AF)

Without any loss of generality one can think of G as of a complete graph with some edges (those not actually present) having zero weight. Under this assumption of completeness, a p-cut decomposes G into p subcliques, leading to the following properties of the feasible solutions. First of all, for any three vertices presence of any two edges between them induces presence of the whole triangle on these vertices. If one calls two edges having a vertex in common *adjacent edges*, then the property can be expressed as follows: each pair of adjacent edges induces the third edge adjacent to both of them. The next property is that any particular vertex in a subclique is connected to any other vertex in a subclique. These two simple properties play an important role in our formulation AF. It uses the following Boolean variables: x_{ij} is nonzero only if edge (i,j) is not removed by a p-cut, and y_i is nonzero only if vertex i is selected as a special vertex. Each vertex in a subclique can be selected as a special vertex, and exactly one vertex in a subclique is special. This setting is needed to count subcliques. The rest of notations are preserved from the previous sections, and AF can be expressed as:

$$S - \sum_i \sum_{j>i} c_{ij} x_{ij} \longrightarrow \min \tag{4.9}$$

$$\sum_i y_i = p \tag{4.10}$$

$$x_{ij} \leq 2 - y_i - y_j \quad \forall i \neq j \tag{4.11}$$

$$x_{ij} \geq x_{il} + x_{jl} - 1 \quad \forall i \neq j \neq l \tag{4.12}$$

$$x_{ij} = x_{ji} \quad \forall i \neq j \tag{4.13}$$

$$y_i \in \{0,1\} \quad \forall i \tag{4.14}$$

$$x_{ij} \in \{0,1\} \quad \forall i \neq j\ . \tag{4.15}$$

Similarly to SF, the objective (4.9) minimizes the difference between the sum of all edge weights and the sum of weights of the edges within subcliques, i.e. the weight of the p-cut. Constraint (4.10) ensures that exactly p special vertices must be selected, while constraints (4.11) force each pair of special vertices to be disconnected, such that each subclique contains a single special vertex. Constraints (4.12) ensure the mentioned above property: any two adjacent edges force the third adjacent edge to be preserved. Finally, constraints (4.13) preserve undirected structure of the problem. It is not hard to understand that these constraints can be used to eliminate half of the x-variables, i.e. to use only those x_{ij} for which $i < j$ holds. In our experiments we used such a reduced formulation.

4.5 Additional constraints

Though the MINpCUT based model exactly minimises the intercell movement, additional constraints ensuring practical feasibility of obtained solutions are usually needed. Moreover, it is desirable to be able to take into account additional factors and managerial preferences. As our model has quite a general structure, in principle, any constraints that can be expressed in a linear form can be included. In this section we give some examples of extending our formulations SF and AF with additional constraints.

First of all, some flexibility in the model is provided by weights c_{ij}. It is not hard to understand that these values can be defined either as the number of parts travelling between machines i and j, or their total mass, volume, etc. However, the range of possible factors is not limited to properties of parts. For example, it may be desirable to account for the available workforce and reduce the so-called cross-training costs (see, e.g., Bhatnagar & Saddikuti (2010)). In this case, the objective is to ensure that each worker is able to deal with as much machines in his cell as possible. This issue can be modelled by making weights c_{ij} dependent on the number of workers able to operate both machines i and j.

The next issue that can be easily dealt with is based on the fact that some machines cannot be placed in the same cell (e.g., because of safety reasons) while others must be placed close to each other because of managerial considerations or constructional peculiarities. In both formulations SF and AF it is easy to force a pair of machines i and j to be grouped in one cell or in different cells. In case of SF, the constraints

$$v_{ik} = v_{jk} \quad \forall k \tag{4.16}$$

and

$$v_{ik} + v_{jk} \leq 1 \quad \forall k \tag{4.17}$$

force or prohibit assignment of machines i and j to the same cell, respectively. For AF the corresponding constraints are

$$x_{ij} = 1 \tag{4.18}$$

and

$$x_{ij} = 0\,, \tag{4.19}$$

leading to a problem with fewer Boolean variables (as some x-variables become fixed).

Capacity constraints are, probably, the most popular ones in cell formation; these set a limit on the minimum or maximum number of machines in a cell. Indeed, there is little sense in cells containing a single machine, while such solutions are common for the manufacturing data that we experienced. In order to limit the number of machines per cell from below by n_L, SF must be equipped with the following constraints:

$$\sum_i v_{ik} \geq n_L \quad \forall k\,. \tag{4.20}$$

For AF the constraints look like

$$\sum_j x_{ij} + 1 \geq n_L \quad \forall i\,. \tag{4.21}$$

Validity of these constraints can be expressed by the fact that each vertex is connected to all other vertices within its subclique. Thus, the number of incident edges not removed by a p-cut plus the vertex i itself is equal to the total number of vertices in a subclique. It should be mentioned that the system of constraints (4.21) is redundant in a sense that p constraints written for vertices i all lying in different subcliques are sufficient. However, it is not known beforehand which p vertices will belong to different subcliques in an optimal solution (these are determined by y-variables). Upper bounds on the number of machines per cell can be set in a similar way.

Workload balancing constraints are used to ensure that cells have a balanced load in terms of working hours, so that the tasks are evenly divided between the cells. Such balancing helps to avoid the situation when one cell (and the corresponding team of workers) is overloaded, while another is underutilised. If one denotes by w_i the workload of machine i and by w_L the lower bound on the workload per cell, then constraints for SF and AF become

$$\sum_i w_i v_{ik} \geq w_L \quad \forall k \tag{4.22}$$

and

$$\sum_j w_j x_{ij} + w_i \geq w_L \quad \forall i \, , \tag{4.23}$$

respectively. It can be seen that the workload balancing and capacity constraints have very similar structure. Instead of limiting the workload per cell one may be interested in limiting the difference between workloads of cells. The corresponding constraints can be easily derived from (4.22) and (4.23) by observing that the left hand side of these constraints calculates the workload for each cell and by limiting the differences between all possible pairs of these workloads.

In the rest of this section we discuss a much less trivial issue – the presence of identical machines. It is not uncommon, especially in large manufacturing systems, that some most extensively used machines are present in several copies. This means that each part can be processed on either of these machines equally well and if one applies any clustering algorithm directly, the identical machines will always be grouped together. On the other hand, placing them in different cells reduces intercell movement (if they are needed in more than one cell). This issue is usually hard to model as it leads to the so-called disjunctive constraints – a part can be processed on either of the identical machines. However, here we show that our formulations can be adjusted to account for identical machines without significant complication.

Note that in the above discussion we could have used the term "machine types" instead of "machines", implicitly assuming that identical machines are placed together. Let us call placement of identical machines in different cells *separation of identical machines*. Now we are going to modify the formulations SF and AF such that they allow separation of identical machines; these will be denoted as SFs and AFs, respectively.

Let us denote by n_i the number of identical machines of type i. Recall that indices i and j enumerate machine types, $i, j \in \{1, \ldots, n\}$, and index k enumerates cells or subgraphs, $k \in \{1, \ldots, p\}$. The modification SFs of formulation SF can be written as follows:

$$S - \sum_i \sum_{j>i} c_{ij} x_{ij} \longrightarrow \min \tag{4.24}$$

$$\sum_i v_{ik} \geq 1 \quad \forall k \tag{4.25}$$

$$\sum_k v_{ik} \geq 1 \quad \forall i \tag{4.26}$$

$$\sum_k v_{ik} \leq n_i \quad \forall i \tag{4.27}$$

$$z_{ijk} \leq v_{ik} \quad \forall i \neq j, k \tag{4.28}$$

$$z_{ijk} \leq v_{jk} \quad \forall i \neq j, k \tag{4.29}$$

$$z_{ijk} \geq v_{ik} + v_{jk} - 1 \quad \forall i \neq j, k \tag{4.30}$$

$$x_{ij} \leq \sum_k z_{ijk} \quad \forall i \neq j \tag{4.31}$$

$$x_{ij} \leq 1 \quad \forall i \neq j \tag{4.32}$$

$$v_{ij} \in \{0, 1\} \quad \forall i \neq j \tag{4.33}$$

$$z_{ijk} \in [0, \infty) \quad \forall i \neq j, k \tag{4.34}$$

$$x_{ij} \in [0, \infty) \quad \forall i \neq j\ , \tag{4.35}$$

where v-, x- and z-variables have the same meaning as in Sections 4.3 and 4.4. The objective (4.24) minimizes the total weight of the edges removed by a p-cut, and constraints (4.25), (4.28)-(4.30) and (4.33)-(4.34) are inherited from SF. Constraints (4.26)-(4.27) are a generalization of constraint (4.3). These require each vertex (machine type) i to be included into at least one subgraph (cell) and at most n_i subgraphs (at most n_i machines of type i are used). Finally, constraints (4.31)-(4.32) cut the edge between i and j if this pair of vertices is not contained in any of p subgraphs and ensure that each edge can be cut only once. It can be seen that these constraints also force x-variables to take 0-1 values and the numbers of Boolean variables in formulations SF and SFs are equal.

The modification of the formulation AF to allow separation of the machines is even simpler than in case of SF. This task can be accomplished by considering a graph where each vertex corresponds to a single machine (not to a machine type, as in case of SFs) and penalising the objective such that vertices corresponding to identical machines are forced to be assigned to different subcliques. The penalising term for any pair of identical machines i and i' can be written as

$$+ \left[\sum_j c_{ij} \right] x_{ii'}\ , \tag{4.36}$$

where the constant in brackets is large enough to ensure that the negative impact of placing vertices i and i' into one subclique cannot be compensated by any arrangement of other vertices. Instead of penalising the objective, one may also add the following constraint:

$$x_{ii'} = 0\,. \tag{4.37}$$

However, such constraints may conflict with capacity or other constraints leading to an infeasible problem. Thus, AFs inherits the structure of AF and has few additional constraints or terms in the objective function.

In conclusion, we would like to mention that if for some machine type i holds $n_i \geq p$, then it can be excluded from consideration as a machine of this type can be added to each cell. In particular, this implies that in case of two cells identical machines make the problem smaller, therefore easier. On the negative side, separation of equivalent machines may lead to load-related problems: one of the machines may become overloaded while the other is rarely used. Thus, additional load balancing constraints may be necessary.

4.6 Computational Experiments

In order to motivate the exact model for cell formation we considered several used in literature instances that were tackled by heuristic approaches. The scope of this study was restricted to the instances containing the operations sequence data and to the papers reporting complete solutions (assignment of machines to cells) so that the amount of intercell movement can be estimated. In order to ensure the most consistent comparison, we restricted the number of machines per cell both from below and from above by the values inherent to the corresponding solutions from the literature. The computational results are summarised in Table 4.1, where the first two columns indicate the number of machines, parts, and the number of cells to be made. The next column indicates the amount of intercell movement achieved by our MINpCUT based model. The following two columns contain the best result we could find in the literature and a corresponding reference; the last column reports running times for SF. As can be seen from Table 3.1, in most cases contemporary heuristics were unable to find optimal solutions. At the same time, running times for our model are quite limited, except the last considered instance which we could not solve to optimality. In these and the following experiments we used a moderate PC (Intel Core2 Duo, 2.33 GHz, 2 GB RAM) and Xpress-MP as a MILP solver. The solver was restricted to use one processor core.

The aim of our further experiments was to check computational properties of

Table 4.1. Performance comparison with heuristic approaches from literature in terms of intercell movement.

size	p	our result	best known	source	time, sec.
8×20	3	17	17	Nair & Narendran (1998)	<1
12×19	2	9	16	Ahi et al. (2009)	<1
12×19	3	20	27	Ahi et al. (2009)	<1
18×35	4	47	54	Ahi et al. (2009)	7
20×20	5	17	18	Ahi et al. (2009)	10
20×51	5	36	36	Ahi et al. (2009)	17
20×20	5	18	19	Nair & Narendran (1998)	13
25×40	4	17	22	Ahi et al. (2009)	8
25×40	6	27	45	Ahi et al. (2009)	140
25×40	8	56	72	Nair & Narendran (1998)	∼ 24 h*

* – interrupted due to memory limitations, best integer solution is reported (best lower bound is 50.963). In fact, the reported solution was found within 1 hour, the rest of the time was spent on tightening the lower bound.

the introduced model and to show its practical applicability by means of an industrial case. As a testbed for the experiments we considered data from a small company producing high precision tools. The quantitative characteristics of the dataset are as follows:

- time period: 11 months;
- 30 machine types;
- 7563 part types;
- 25080 operations (4149 part moves between machines).

First, we tried to solve the unconstrained CF problem for all possible values of p using both our formulations. We limited the running time by 10 hours and the results are summarised in Fig.4.2. As predicted in Section 4.3, the running times for SF grow with increasing p, while AF is efficient for p close to 1 and to n. Note that the instance with 30 machines for p up to 15 can be solved within an hour, which is more than reasonable taking into account that cells are not reconfigured every day. Note also that the size of the instance under consideration is quite substantial: after reviewing hundreds of papers on cell formation we were able to find only 3 realistic instances with more than 30 machines (the largest one having 50 machines). At the same time, the number of parts does not affect the performance of our model.

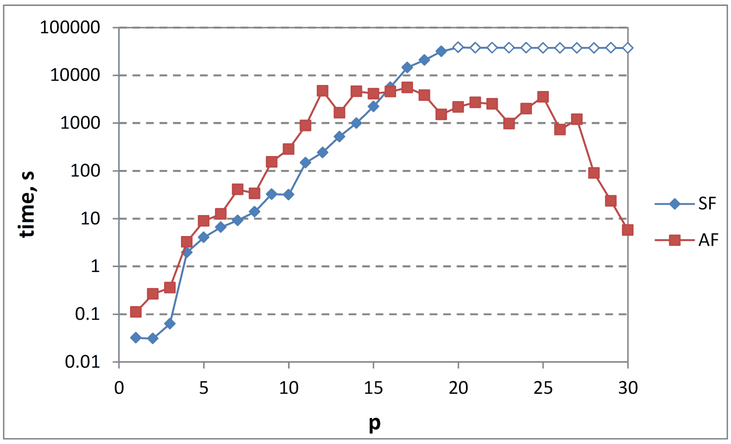


Figure 4.2. Solution times for an instance with 30 machines (limited by 10 hours)

Next, we conducted several experiments in order to demonstrate the issue of identical machines and its possible impact. Figure 4.3 shows the intercell movement, expressed as a percentage of the total parts movement, for $p = 2$ cells and different lower bounds on the number of machines per cell. It can be seen that if possibility of separating identical machines is ignored, balanced cells are impossible due to a large intercell movement of 18.29%. In the opposite case, two reasonable cells can be obtained with only 0.19% intercell movement. In case of three cells, the corresponding figures are 24.13% and 1.16%. Figures 4.4 and 4.5 illustrate the obtained cellular decompositions, matrices display the numbers of parts travelling directly between each pair of machines. Note that the fact that the rightmost cell in Figure 4.5 has much fewer intracell movement than the other two cells does not imply that this cell has very low load in terms of working hours. Small intracell movement means only that machines within a cell share very few parts, while each one can be sunstantially loaded in order to produce its unique set of parts.

4.7 Summary

In this chapter an exact model for the cell formation problem in group technology is developed. We have demonstrated that a machine-parts incidence matrix does not contain enough information to obtain truly optimal solutions. As becomes apparent from the presented experimental comparison, recent heuristics taking operational sequences into account usually lead to suboptimal solutions. We have

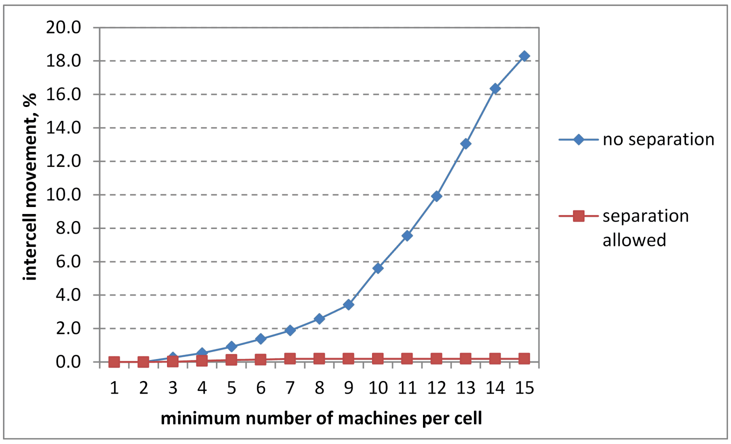
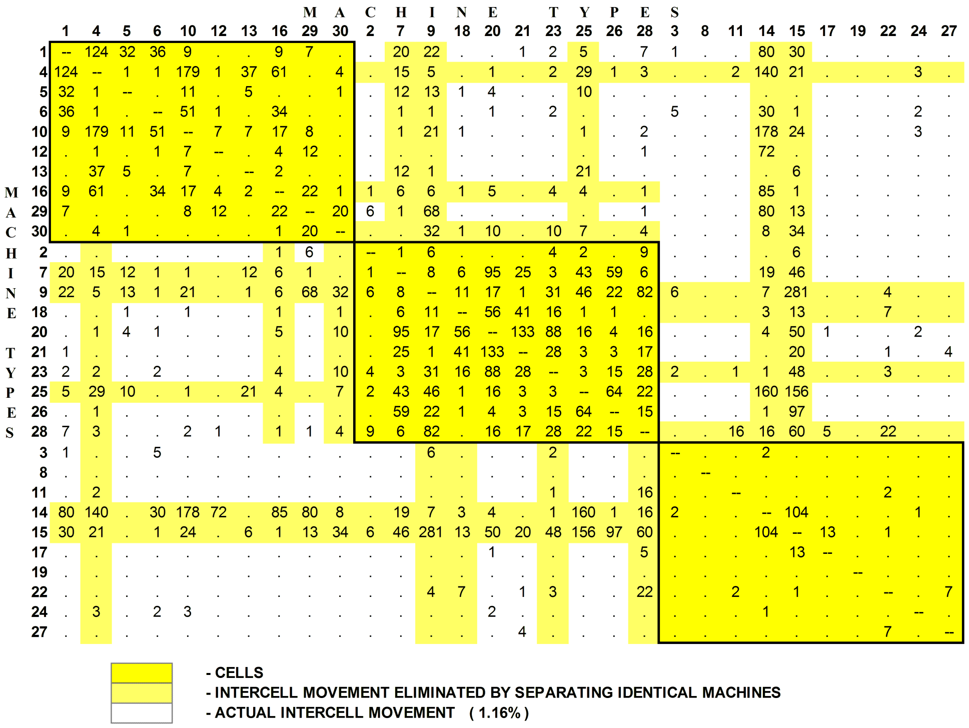


Figure 4.3. Intercell movement for different restrictions on the number of machines per cell for the case of two cells

									M	A	C	H	I	N	E		T	Y	P	E	S									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**9**	**10**	**12**	**13**	**14**	**15**	**24**	**29**	**8**	**11**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**25**	**26**	**27**	**28**	**30**
	1	--	.	1	124	32	36	20	22	9	.	.	80	30	.	7	.	.	9	.	.	.	.	1	.	2	5	.	.	7	.
	2	.	--	.	.	.	.	1	6	.	.	.	.	6	.	6	.	.	1	.	.	.	.	.	.	4	2	.	.	9	.
	3	1	.	--	.	.	5	.	6	.	.	.	2	.	.	.	.	.	.	.	.	.	.	.	.	2	.	.	.	.	.
	4	124	.	.	--	1	1	15	5	179	1	37	140	21	3	.	.	2	61	.	.	.	1	.	.	2	29	1	.	3	4
	5	32	.	.	1	--	.	12	13	11	.	5	.	.	.	.	.	.	.	.	1	.	4	.	.	.	10	.	.	.	1
	6	36	.	5	1	.	--	1	1	51	1	.	30	1	2	.	.	.	34	.	.	.	1	.	.	2	.	.	.	.	.
	7	20	1	.	15	12	1	--	8	1	.	12	19	46	.	1	.	.	6	.	6	.	95	25	.	3	43	59	.	6	.
M	**9**	22	6	6	5	13	1	8	--	21	.	1	7	281	.	68	.	.	6	.	11	.	17	1	4	31	46	22	.	82	32
A	**10**	9	.	.	179	11	51	1	21	--	7	7	178	24	3	8	.	.	17	.	1	.	.	.	.	.	1	.	.	2	.
C	**12**	.	.	.	1	.	1	.	.	7	--	.	72	.	.	12	.	.	4	.	.	.	.	.	.	.	.	.	.	1	.
H	**13**	.	.	.	37	5	.	12	1	7	.	--	.	6	.	.	.	.	2	.	.	.	.	.	.	.	21	.	.	.	.
I	**14**	80	.	2	140	.	30	19	7	178	72	.	--	104	1	80	.	.	85	.	3	.	4	.	.	1	160	1	.	16	8
N	**15**	30	6	.	21	.	1	46	281	24	.	6	104	--	.	13	.	.	1	13	13	.	50	20	1	48	156	97	.	60	34
E	**24**	.	.	.	3	.	2	.	.	3	.	.	1	.	--	.	.	.	.	.	.	.	2	.	.	.	.	.	.	.	.
	29	7	6	.	.	.	.	1	68	8	12	.	80	13	.	--	.	.	22	.	.	.	.	.	.	.	.	.	.	1	20
T	**8**	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	--	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Y	**11**	.	.	.	2	.	.	.	.	.	.	.	.	.	.	.	.	--	.	.	.	.	.	.	2	1	.	.	.	16	.
P	**16**	9	1	.	61	.	34	6	6	17	4	2	85	1	.	22	.	.	--	.	1	.	5	.	.	4	4	.	.	1	1
E	**17**	.	.	.	.	.	.	.	.	.	.	.	.	13	.	.	.	.	.	--	.	.	1	.	.	.	.	.	.	5	.
S	**18**	.	.	.	.	1	.	6	11	1	.	.	3	13	.	.	.	.	1	.	--	.	56	41	7	16	1	1	.	.	1
	19	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	--	.	.	.	.	.	.	.	.	.
	20	.	.	.	1	4	1	95	17	.	.	.	4	50	2	.	.	.	5	1	56	.	--	133	.	88	16	4	.	16	10
	21	1	.	.	.	.	.	25	1	.	.	.	.	20	.	.	.	.	.	.	41	.	133	--	1	28	3	3	4	17	.
	22	.	.	.	.	.	.	.	4	.	.	.	.	1	.	.	.	2	.	.	7	.	.	1	--	3	.	.	7	22	.
	23	2	4	2	2	.	2	3	31	.	.	.	1	48	.	.	.	1	4	.	16	.	88	28	3	--	3	15	.	28	10
	25	5	2	.	29	10	.	43	46	1	.	21	160	156	.	.	.	.	4	.	1	.	16	3	.	3	--	64	.	22	7
	26	.	.	.	1	.	.	59	22	.	.	.	1	97	.	.	.	.	.	.	1	.	4	3	.	15	64	--	.	15	.
	27	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	4	7	.	.	.	--	.	.
	28	7	9	.	3	.	.	6	82	2	1	.	16	60	.	1	.	16	1	5	.	.	16	17	22	28	22	15	.	--	4
	30	.	.	.	4	1	.	.	32	.	.	.	8	34	.	20	.	.	1	.	1	.	10	.	.	10	7	.	.	4	--

🟨 - CELLS
▨ - INTERCELL MOVEMENT ELIMINATED BY SEPARATING IDENTICAL MACHINES
□ - ACTUAL INTERCELL MOVEMENT (0.19%)

Figure 4.4. An optimal decomposition into two cells (zero entries are denoted by dots)



									M	A	C	H	I	N	E		T	Y	P	E	S									
		1	**4**	**5**	**6**	**10**	**12**	**13**	**16**	**29**	**30**	**2**	**7**	**9**	**18**	**20**	**21**	**23**	**25**	**26**	**28**	**3**	**8**	**11**	**14**	**15**	**17**	**19**	**22**	**24**	**27**
	1	--	124	32	36	9	.	.	9	7	.	.	20	22	.	.	1	2	5	.	7	1	.	.	80	30	.	.	.	.	.
	4	124	--	1	1	179	1	37	61	.	4	.	15	5	.	1	.	2	29	1	3	.	.	2	140	21	.	.	.	3	.
	5	32	1	--	.	11	.	5	.	.	1	.	12	13	1	4	.	.	10	.	.	.	.	.	.	.	.	.	.	.	.
	6	36	1	.	--	51	1	.	34	.	.	.	1	1	.	1	.	2	.	.	.	5	.	.	30	1	.	.	.	2	.
	10	9	179	11	51	--	7	7	17	8	.	.	1	21	1	.	.	.	1	.	2	.	.	.	178	24	.	.	.	3	.
	12	.	1	.	1	7	--	.	4	12	.	.	.	.	.	.	.	.	.	.	1	.	.	.	72	.	.	.	.	.	.
	13	.	37	5	.	7	.	--	2	.	.	.	12	1	.	.	.	.	21	.	.	.	.	.	.	6	.	.	.	.	.
M	**16**	9	61	.	34	17	4	2	--	22	1	1	6	6	1	5	.	4	4	.	1	.	.	.	85	1	.	.	.	.	.
A	**29**	7	.	.	.	8	12	.	22	--	20	6	1	68	.	.	.	.	.	.	1	.	.	.	80	13	.	.	.	.	.
C	**30**	.	4	1	.	.	.	.	1	20	--	.	.	32	1	10	.	10	7	.	4	.	.	.	8	34	.	.	.	.	.
H	**2**	.	.	.	.	.	.	.	1	6	.	--	1	6	.	.	.	4	2	.	9	.	.	.	.	6	.	.	.	.	.
I	**7**	20	15	12	1	1	.	12	6	1	.	1	--	8	6	95	25	3	43	59	6	.	.	.	19	46	.	.	.	.	.
N	**9**	22	5	13	1	21	.	1	6	68	32	6	8	--	11	17	1	31	46	22	82	6	.	.	7	281	.	.	4	.	.
E	**18**	.	.	1	.	1	.	.	1	.	1	.	6	11	--	56	41	16	1	1	.	.	.	.	3	13	.	.	7	.	.
	20	.	1	4	1	.	.	.	5	.	10	.	95	17	56	--	133	88	16	4	16	.	.	.	4	50	1	.	.	2	.
T	**21**	1	.	.	.	.	.	.	.	.	.	.	25	1	41	133	--	28	3	3	17	.	.	.	.	20	.	.	1	.	4
Y	**23**	2	2	.	2	.	.	.	4	.	10	4	3	31	16	88	28	--	3	15	28	2	.	1	1	48	.	.	3	.	.
P	**25**	5	29	10	.	1	.	21	4	.	7	2	43	46	1	16	3	3	--	64	22	.	.	.	160	156	.	.	.	.	.
E	**26**	.	1	.	.	.	.	.	.	.	.	.	59	22	1	4	3	15	64	--	15	.	.	.	1	97	.	.	.	.	.
S	**28**	7	3	.	.	2	1	.	1	1	4	9	6	82	.	16	17	28	22	15	--	.	.	16	16	60	5	.	22	.	.
	3	1	.	.	5	.	.	.	.	.	.	.	.	6	.	.	.	2	.	.	.	--	.	.	2	.	.	.	.	.	.
	8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	--	.	.	.	.	.	.	.	.
	11	.	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	16	.	.	--	.	.	.	.	2	.	.
	14	80	140	.	30	178	72	.	85	80	8	.	19	7	3	4	.	1	160	1	16	2	.	.	--	104	.	.	.	1	.
	15	30	21	.	1	24	.	6	1	13	34	6	46	281	13	50	20	48	156	97	60	.	.	.	104	--	13	.	1	.	.
	17	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	5	.	.	.	.	13	--	.	.	.	.
	19	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	--	.	.	.
	22	.	.	.	.	.	.	.	.	.	.	.	.	4	7	.	1	3	.	.	22	.	.	2	.	1	.	.	--	.	7
	24	.	3	.	2	3	.	.	.	.	.	.	.	.	.	2	.	.	.	.	.	.	.	.	1	.	.	.	.	--	.
	27	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	4	.	.	.	.	.	.	.	.	.	.	.	7	.	--

- CELLS
- INTERCELL MOVEMENT ELIMINATED BY SEPARATING IDENTICAL MACHINES
- ACTUAL INTERCELL MOVEMENT (1.16%)

Figure 4.5. An optimal decomposition into three cells (zero entries are denoted by dots)

also showed that an exact model can be independent on the number of parts and demonstrated importance of this property by an industrial example.

It was shown that the cell formation problem with a fixed number of cells is equivalent to the minimum multicut problem (also, if the input data is given by a machine-part incidence matrix). This fact immediately implies polynomial solvability of the former in case $p = 2$. For an arbitrary number of cells, however, the problem remains NP-hard. Yet, it can still be solved to optimality in many practical cases due to the limited number of machines in real manufacturing systems. If the instance is too large to be solved optimally, the following iterative heuristic procedure can be used. For the initial problem solve a MIN2CUT problem, then iteratively pick the largest cell and solve for it the minimum 2-cut until p cells are obtained. It is easy to show that whenever (almost) independent cells are possible the obtained solution will be globally optimal. Optimality conditions for this or another heuristics may become possible directions for future research.

We presented two MILP formulations that we call SF and AF, and demonstrated their tractability for moderately sized instances by means of an industrial example. It was found that SF is more efficient than AF for small values of p, becoming intractable for larger ones. At the same time, AF performs well for values of p close to 1 and to the number of vertices in a graph. The latter formulation better reflects structure of the problem and, potentially, may be more suitable for size reduction based on graph-theoretic considerations.

We considered several additional constraints ranging from very popular capacity constraints to the particular case of disjunctive constraints induced by identical machines. To the best of our knowledge, there are no models adequately handling identical machines, even though some attempts are reported in literature. In contrast, we showed that identical machines can be modelled in both our formulations and demonstrated how it works by means of an industrial example. Overall, our numerical experiments confirmed practical applicability of the proposed model for real-life problems with a moderate number of machine types.

Possible directions for the further research may include development of advanced formulations for the minimum multicut problem, and/or problem size reduction approaches. Such directions may be interesting taking into account that the problem has quite a general clustering nature and its possible applications are not limited to cell formation in group technology.

Chapter 5

Multiobjective nature of cell formation

5.1 Introduction

Throughout the long history of the cell formation problem, not only the solution methods evolved but also the major goal of making independent cells was interpreted differently by different authors. The earliest approaches to CF, dealing with a binary machine-part incidence matrix, were aimed only at minimising the number of intercell moves (exceptional elements in the block-diagonalised matrix, n_e). Quite soon this goal was extended to simultaneous minimisation of the number of intercell moves and maximisation of the number of intracell ones (i.e. minimisation of the number of voids in the diagonal blocks, n_v). Up to now, this objective has proved extremely popular, as becomes clear while looking at objective functions and the widely used similarity and solution quality measures (see Section 1.3).

The reasoning for introducing n_v into the objective is twofold. First of all, this value plays a crucial role for the algorithms determining the optimal number of cells. As discussed in Chapter 1, the minimum value of n_e is a nondecreasing function on the number of cells. This implies that any such algorithm tends to group all machines into one cell resulting in zero intercell movement. Taking into account that the minimum value of n_v has the opposite behaviour (depending on the number of cells), combining the two values in the objective forces a "reasonable" number of cells to be generated. Secondly, some authors (see, e.g., F. T. S. Chan et al., 2008) argue on an importance of intracell movements and claim that n_v must be present in the objective, irrespectively of the type of an algorithm. However, very little is said about the reasonable ratio between importances of the two factors, usually these are supposed to be equally important. For example, in the expression for grouping efficiency (3.67) the weighting coefficients for the two factors are equal.

While a majority of the papers in the field deals with intercell movement, a number of authors consider completely different objectives, for example, minimisation of cross-training costs (see, e.g., Bhatnagar & Saddikuti, 2010), minimisation of load imbalance (Suresh & Slomp, 2001) or set-up time reduction (see, e.g., Suresh, 1992; Shafer & Rogers, 1993).

The goal of this chapter is to consider several objectives relevant to cell formation and to present the ways of including them into the models from Chapters 3 and 4.

In the rest of this chapter we discuss appropriateness of combining the amounts of inter- and intracell movement in the objective and demonstrate possible pitfalls of minimizing only the intercell movement. We also discuss some other possible goals considered in the literature, like those related to workforce or set-up time savings. For each of the goals we describe how it can be integrated into the models presented in Chapters 3 and 4.

5.2 Problems with a minimisation of the intercell movement

Before discussing the appropriateness of minimising the amount of inter- and intracell movement, let us mention that by minimising the number of exceptions in a machine-part incidence matrix (MPIM) one does not necessary minimise the actual amount of intercell movement in the system (as shown in Chapter 4). This can be explained by the fact that each exception indicates that a certain part has to travel between two cells, without indicating how many travels are needed. Moreover, it is possible to prove that if an approach uses a (dis)similarity measure derived from the MPIM, it does not, in general case, minimise the number of exceptions.

Theorem 6. *None of the approaches using similarity coefficients derived from the machine-part incidence matrix minimises the number of exceptions.*

Proof. The counter-example shown in Figure 5.1 illustrates the issue. It can be seen that machine i shares several parts with only one machine from cell 1 and one part with several machines from cell 2. Thus, similarity between i and any other machine from cell 2 is some small value. On the other hand, machine i shares a more noticeable number of parts with machine j from cell 1 and, clearly, should be assigned to this cell in order to minimise the intercell movement. Consider the following three extreme cases:

1. for each cell the objective function contains a sum of similarities only from one central machine to all the other ones within the cell (like in case of the PMP);
2. for each cell the objective function contains a sum of similarities for all pairs within the cell (like in case of the MINpCUT);
3. for each cell the objective function contains a sum of similarities for some pairs falling into a spanning tree pattern (like in case of MST or hierarchical clustering).

In the first case it is always possible to force machine j to be not the central machine, leading to a zero similarity between i and centre of cell 1. Thus, machine i will be assigned to cell 2. In the second case, if cell 2 contains sufficiently large number of machines, then the sum of small similarities with all other machines results in a substantial number exceeding similarity between i and j. The third case reflects hierarchical clustering methods. Though being capable to deal with the example from Figure 5.1, they are heuristic by their nature and for each of them counter-examples exist.

Speaking more generally, if the objective function can be represented as follows (x_{ik} define if machines i and k are in the same cell, I_2 – set of machines from cell 2):

$$\ldots + s_{ij}x_{ij} + \sum_{k \in I_2} s_{ik}x_{ik} + \ldots\ , \tag{5.1}$$

then the sum of small similarities of machine i with machines from cell 2 may overwhelm the substantial similarity with machine j from cell 1. In addition, if in a feasible solution not all similarities within a cell are counted, then the similarity between machines i and j may be excluded from consideration. For a particular algorithm, one may arrange non-fixed elements in the matrix from Figure 5.1 such that at least one of these cases takes place. The only exception for which this may not work includes sequential algorithms (like MST or hierarchical clustering) that group machines by picking the "heaviest" links first (these all have a heuristic nature).

Clearly, the above considerations are valid not only for sums but also for any nondecreasing function. ■

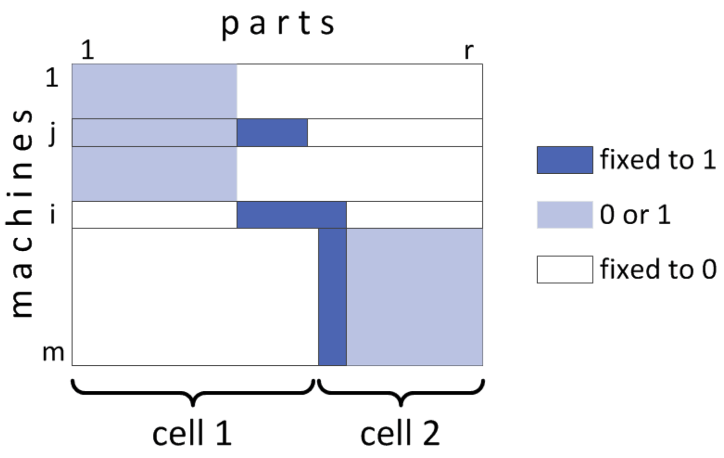
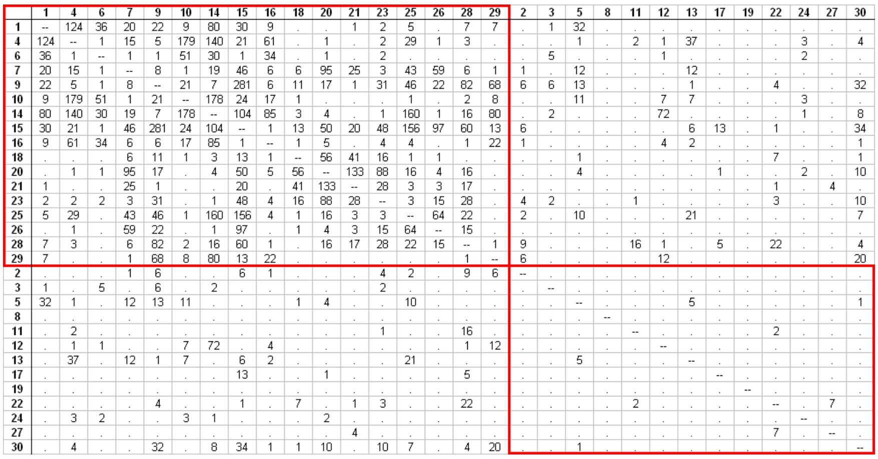


Figure 5.1. Counter-example illustrating Theorem 6 .

To make the further analysis accurate, we will refer to a MINpCUT based model, that was proven to be exact in Chapter 4. Clearly, the amount of intercell movement is an important characteristics that reflects the degree of cell independence and must be included into the objective. Though some authors (see, e.g, F. T. S. Chan et al., 2008) claim that the amount of intracell movement must also be optimised, we argue that it may be excluded from consideration. In order to support this argument, we would like to give an illustrative example based on real manufacturing data. Consider the cells presented in Figure 5.2 (we restricted the number of machines per cell such that the difference is at most 4). It can be seen that one of the cells has almost no intracell movement and contains machines that have very little in common. Clearly, this situation does not fit into the classical "theory" of cell formation. Yet, there is a natural reasoning behind the obtained solution and it can be checked that it is the best possibility available. Clearly, classical cells (with substantial intracell movement and small intercell movement) are not possible in the considered case because the pattern of connections is very dense. The model "understands" that and finds the only possible solution: separate machines that have fewer connections with the "core" of the manufacturing system. Such a solution makes sense from different perspectives. For example, if cells were made in order to comply with spatial constraints and/or are spatially distant, then the goal is reached: obtained cellular decomposition ensures the minimum traffic of parts between cells. If cells were made in order to make the management easier, then the goal is again reached: there are two smaller systems with a limited interaction



	1	4	6	7	9	10	14	15	16	18	20	21	23	25	26	28	29	2	3	5	8	11	12	13	17	19	22	24	27	30
1	--	124	36	20	22	9	80	30	9	.	.	1	2	5	.	7	7	.	1	32	.	.	.	.	.	.	.	.	.	.
4	124	--	1	15	5	179	140	21	61	.	1	.	2	29	1	3	.	.	.	1	.	2	1	37	.	.	.	3	.	4
6	36	1	--	1	1	51	30	1	34	.	1	.	2	.	.	.	.	.	5	.	.	.	1	.	.	.	.	2	.	.
7	20	15	1	--	8	1	19	46	6	6	95	25	3	43	59	6	1	1	.	12	.	.	.	12	.	.	.	.	.	.
9	22	5	1	8	--	21	7	281	6	11	17	1	31	46	22	82	68	6	6	13	.	.	.	1	.	.	4	.	.	32
10	9	179	51	1	21	--	178	24	17	1	.	.	.	1	.	2	8	.	.	11	.	.	7	7	.	.	.	3	.	.
14	80	140	30	19	7	178	--	104	85	3	4	.	1	160	1	16	80	.	2	.	.	.	72	.	.	.	.	1	.	8
15	30	21	1	46	281	24	104	--	1	13	50	20	48	156	97	60	13	6	.	.	.	.	.	6	13	.	1	.	.	34
16	9	61	34	6	6	17	85	1	--	1	5	.	4	4	.	1	22	1	.	.	.	.	4	2	.	.	.	.	.	1
18	.	.	.	6	11	1	3	13	1	--	56	41	16	1	1	.	.	.	.	1	.	.	.	.	.	.	7	.	.	1
20	.	1	1	95	17	.	4	50	5	56	--	133	88	16	4	16	.	.	.	4	.	.	.	.	1	.	.	2	.	10
21	1	.	.	25	1	.	.	20	.	41	133	--	28	3	3	17	.	.	.	.	.	.	.	.	.	.	1	.	4	.
23	2	2	2	3	31	.	1	48	4	16	88	28	--	3	15	28	.	4	2	.	.	1	.	.	.	.	3	.	.	10
25	5	29	.	43	46	1	160	156	4	1	16	3	3	--	64	22	.	2	.	10	.	.	.	21	.	.	.	.	.	7
26	.	1	.	59	22	.	1	97	.	1	4	3	15	64	--	15	.	.	.	.	.	.	.	.	.	.	.	.	.	.
28	7	3	.	6	82	2	16	60	1	.	16	17	28	22	15	--	1	9	.	.	.	16	1	.	5	.	22	.	.	4
29	7	.	.	1	68	8	80	13	22	.	.	.	.	.	.	1	--	6	.	.	.	.	12	.	.	.	.	.	.	20
2	.	.	.	1	6	.	.	6	1	.	.	.	4	2	.	9	6	--	.	.	.	.	.	.	.	.	.	.	.	.
3	1	.	5	.	6	.	2	.	.	.	.	.	2	.	.	.	.	.	--	.	.	.	.	.	.	.	.	.	.	.
5	32	1	.	12	13	11	.	.	.	1	4	.	.	10	.	.	.	.	.	--	.	.	.	5	.	.	.	.	.	1
8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	--	.	.	.	.	.	.	.	.	.
11	.	2	.	.	.	.	.	.	.	.	.	.	1	.	.	16	.	.	.	.	.	--	.	.	.	.	2	.	.	.
12	.	1	1	.	.	7	72	.	4	.	.	.	.	.	.	1	12	.	.	.	.	.	--	.	.	.	.	.	.	.
13	.	37	.	12	1	7	.	6	2	.	.	.	.	21	.	.	.	.	.	5	.	.	.	--	.	.	.	.	.	.
17	.	.	.	.	.	.	.	13	.	.	1	.	.	.	.	5	.	.	.	.	.	.	.	.	--	.	.	.	.	.
19	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	--	.	.	.	.
22	.	.	.	.	4	.	.	1	.	7	.	1	3	.	.	22	.	.	.	.	.	2	.	.	.	.	--	.	7	.
24	.	3	2	.	.	3	1	.	.	.	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	--	.	.
27	.	.	.	.	.	.	.	.	.	.	.	4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	7	.	--	.
30	.	4	.	.	32	.	8	34	1	1	10	.	10	7	.	4	20	.	.	1	.	.	.	.	.	.	.	.	.	--

Figure 5.2. Industrial example demonstrating “non-classical” cells.

implying a lower amount of uncertainty. In addition, one of these systems represents a set of non-interacting machines and is, therefore, very easy to manage (non-interacting machines can be managed independently). Thus, intracell movement does not play an important role, as sometimes cells with almost independent machines may be preferable. Note, that this example also illustrates another shortcoming of similarity based approaches: all of them assume intensive intracellular traffic and are very unlikely to produce the given in Figure 5.2 solution, even though it minimises the intercellular traffic. While providing a reasonable decomposition of the manufacturing system, this solution does not support implementation of group technology. Thus, Figure 5.2 illustrates that from the GT perspective the minimisation of intercell movement cannot be a sole objective.

5.2.1 Inter- versus intra-cell movement

The idea of minimising the intercell movement can be generalised, as done, for example, by Fallah-Alipour & Shamsi (2008): instead of minimising the cells dependence, minimise the total cost of cells. When aiming at independent cells, one minimises the intercell movement and implicitly assumes that any amount of intracell movement is acceptable. From the cost point of view, this can be interpreted as follows: intercell moves of parts are given some nonnegative cost, while intracell ones cost zero. At the same time, one may assume intracell moves to also have nonzero costs.

It is not hard to understand that such a generalised problem can be easily modelled within the MINpCUT framework just by extending the objective function by

$$+\sum_{i}\sum_{j>i} c'_{ij}\left(1-\sum_{k} z_{ijk}\right) \tag{5.2}$$

or

$$+\sum_{i}\sum_{j>i} c'_{ij}(1-x_{ij}) \tag{5.3}$$

for the formulations given in Sections 4.3 and 4.4, respectively; c'_{ij} – the cost of an intracell movement between machines i and j, indices i and j enumerate machines, k enumerates cells. It can be seen that these modifications change only the coefficients in the objective function without affecting its structure. This means that the complexity of the problem does not increase with these extensions.

It should be noted that the presence of two types of costs implies two objectives having an opposite behaviour depending on the number of cells p: as p increases, the amount of intercell movement also increases, while the amount of intracell movement decreases (see Section 1.3.2). The two objectives balance each other and the constraint on the number of cells can be dropped. In formulation AF (Section 4.3) this can be done in a straightforward way by eliminating constraint (4.10). As the formulation SF (Section 4.3) encodes the number of cells in its structure (the number of variables depends on p), an upper bound on the number of cells must be set and cells may be allowed to be empty by dropping constraints (4.2).

Thus, if both inter- and intracell movement have nonnegative costs, the MINpCUT based model does not need the number of cells to be predefined and can find the optimal one automatically. The PMP based model from Section 3.2 can also be adjusted appropriately, but the necessary modifications are more involved than in case of MINpCUT.

5.2.2 Preserving flows

Based on the example from Fig. 5.2, it is possible to conclude that in case no clear cells are present, minimisation of intercell movement tends to place the least connected machines together. One may argue that this tendency may lead to inappropriate results. For example, consider the two cellular configurations depicted in Figure 5.3 (a) and (b), respectively. In this figure, M1,. . .,M8 denote machines, num-

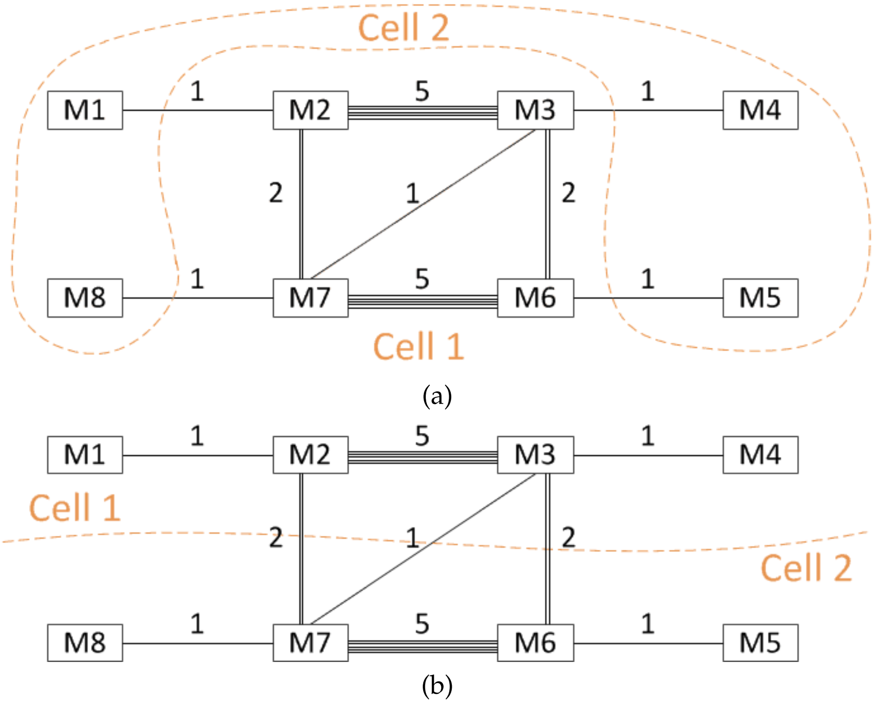


Figure 5.3. Two cellular configurations: (a) minimising intercell movement; (b) preserving line flows

bers at the edges denote the numbers of parts travelling between the corresponding machines. Configuration (a) is generated by minimising the intercell movement, while (b) is made manually. In both cases cells are forced to have an equal number of machines. One may argue that configuration (b) is preferred as it allows for two flows, going through machines M1–M2–M3–M4 and M5–M6–M7–M8. Yet, configuration (a) preserves one even more prominent closed flow going through M2–M3–M6–M7, while the rest of machines are put aside.

If one is interested in preserving flows, it is always possible to force either of the considered in the previous chapters models to keep certain machines in the same cell. However, identification of flows is an interesting problem on itself. Here we propose a simple and computationally efficient MILP model for the flows identification problem. The model uses Boolean decision variables x_{ij}, where indices i and j enumerate machines ($i, j \in \{1, \ldots, m\}$), and can be written as:

$$\sum_{i} \sum_{j, j \neq i} c_{ij} x_{ij} \longrightarrow \max \tag{5.4}$$

$$s.t. \tag{5.5}$$

$$\sum_{j,j} \neq i x_{ij} \leq 1 \quad \forall i \tag{5.6}$$

$$\sum_{i,i \neq j} x_{ij} \leq 1 \quad \forall j \tag{5.7}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j\ , \tag{5.8}$$

where c_{ij} may be chosen either to denote the amount of parts travelling between machines i and j (i.e. have the same meaning as in Chapter 4) or as an amount of parts travelling in one direction (in this case $c_{ij} \neq c_{ji}$ and these correspond to elements of the from-to matrix). The second alternative is better as it takes into account real directed flows instead of aggregated undirected ones. Objective (5.4) maximizes the total amount of flows, while constraints (5.6)-(5.7) ensure that each machine has at most one incoming flow and at most one outgoing flow, respectively. Such setting ensures that a solution will contain three types of flows:

- line flows;
- closed (loop) flows;
- isolated machines.

By playing with the right-hand sides of constraints (5.6)-(5.7) it is possible to extend this set of allowed flow types. For example, if one changes the r.h.s. in (5.7) to 10, tree-like flows with each machine having at most 10 outgoing flows become allowed.

It can be shown that formulation (5.4)-(5.8) essentially represents the well-known assignment problem (see, e.g., Papadimitrou & Steiglitz, 1998, p. 248) and is polynomially solvable. This also implies that combinatorial algorithms for the assignment problem can be applied to solve the flows identification problem. In practice, instances with hundreds of machines can be solved within seconds.

Finally, we would like to mention that the presented flows identification model can be directly used for cell formation, however, it does not guarantee highly independent cells. On the positive side, the model does not require the number of cells to be defined beforehand. Thus, we propose the flows identification model only for preliminary analysis such that its output (the number of cells, dominant flows, etc.) can be used to refine the input for cell formation (e.g. by assigning certain machines to the same cell so that flows are not interrupted).

5.3 Workforce related objectives

Unless a manufacturing system is completely automated, workforce plays an important role in its performance. For example, workers' skills directly influence the setup and processing times of parts, as well as quality of the latter. Moreover, workers may become ill, retire, etc., and it is desirable that someone can substitute them. In other words, it is desirable that a worker is able to operate more than one machine – this is usually referred to as *cross-training*. Importance of cross-training is discussed in a number of papers on CF (see, e.g., Bokhorst et al., 2004; Bhatnagar & Saddikuti, 2010) and the main related goals can be formulated as:

- it is desirable that a worker can operate as much machines in his cell as possible (this improves flexibility and robustness);
- it may be desirable that a worker can operate only machines within his cell (ability to operate machines from other cells means qualifications that are not used but might be paid for);
- costs of additional training must be as small as possible.

The simplest way of reaching the above goals is to define the machine-worker incidence matrix (MWIM, similarly to the machine-part incidence matrix – MPIM) and use it as an input. This idea was used, for example, in Bhatnagar & Saddikuti (2010) by augmenting the MPIM with columns corresponding to workers and reflecting their abilities to operate machines. Such a setting suggests that cell independence from parts and workers points of view are equally preferable. At the same time, it is possible to give these two objectives different priorities by deriving machine-machine similarities from MPIM and MWIM separately and then combining them with weighting coefficients. Clearly, this setting can be directly implemented within the models proposed in Chapters 3 and 4.

It is also worth mentioning that Bhatnagar & Saddikuti (2010) show that the described setting outperforms two-stage procedures that first make machine cells and then assign workers. This is quite natural, as in two stage procedures cells are actually made without taking workforce into an account.

5.4 Set-up time savings

One may notice that all the considered above objectives only implicitly improve the performance of the manufacturing system. In particular, based on the objective values of either of them (nor on the obtained solutions) it is not straightforward how to estimate, for example, the gains in the throughput time of the parts, savings in terms of labour hours, etc. The only exception is as follows: if cells are spatially distant then the amount of intercell movement influences the time needed to deliver parts from one cell to another and then translated into the throughput time of parts.

It is quite natural to pose the following question: is it possible to make cells that explicitly optimise some quantitative manufacturing factors? In this section we consider set-up times of parts as an example of such a factor and propose objectives that make cells supporting a set-up time reduction. Such an objective was considered, for example, in Suresh (1992) and Shafer & Rogers (1993).

First of all, let us briefly consider what is the set-up time and why it can be reduced. Clearly, before a part can be processed by a machine the latter must be adjusted in a proper way and have all necessary tools installed. The time needed for these operations is called the *set-up time* of a particular part on a particular machine. The set-up time is thus a fraction of the total throughput time of a part, and this fraction can be quite substantial. Naturally, several different parts may need similar setups at certain machines and if these parts are processed sequentially then the machines must be adjusted only once before processing all these parts. This means that for all but the first such part the set-up time will be zero. Thus, by proper scheduling of parts at each machines it is possible save on setups. In reality, time needed to change one setup to another may differ for each pair of parts. In this case the problem of finding an optimal schedule (a sequence of parts minimising the total set-up time) can be modelled by the well-known Asymmetric Travelling Salesman Problem (ATSP, see, e.g., Garey & Johnson, 1979) defined on a graph where vertices correspond to parts and weights of the edges correspond to the times needed to switch between the corresponding setups. This implies that a minimisation of set-up times is, generally speaking, an NP-hard problem, even for one machine and without precedence constraints (some parts arrive later than others).

The reality can be somewhat simplified by introducing a notion of a *part family* – a set of parts that have the same setup at a particular machine. Throughout the rest of this thesis we deal with machine-dependent part families, i.e. any two parts belonging to the same family at one machine may belong to different families at another machine. This setting is quite realistic. For example, two parts may need holes of equal diameter and belong to the same family at a drilling machine but if they have different thickness, they belong to different families at a cutting machine.

If these two parts must be painted into the same colour, then at a dyeing machine they are again in the same family. Now, scheduling can be done at a level of part families, rather then at a level of parts. In this way it is possible to reduce the problem size and if the number of families is small (say, 5) then an optimal schedule for a machine can be found reasonably fast even by a complete enumeration[1]. Assuming further that the set-up time of a part family at a machine is independent of which family was processed at this machine before, one may completely eliminate the scheduling issue as the order of families becomes irrelevant, the only requirement is that the parts from one family are processed consecutively.

An optimal scheduling is beyond the scope of this thesis, and here we concentrate on making cells that support set-up time savings, i.e. provide most opportunities for that.

Let us denote by F the maximum number of families per machine; one may assume that each machine has F families, some being empty. Let us also introduce the following notation:

$$F_{j,f,i} = \begin{cases} 1, \text{ if part } j \text{ belongs to family } f \text{ on machine } i \\ 0, \text{ otherwise} \end{cases} \tag{5.9}$$

Under the introduced notations one may define for each pair of machines i and j the following similarity measure $s(i,j)$:

$$s(i,j) = \sum_{k=1}^{r} \sum_{f=1}^{F} (F_{k,f,i} \cdot F_{k,f,j}), \tag{5.10}$$

where r is the number of parts. The similarity measure (5.10) reflects the number of parts belonging to the same family on both machines. It can be directly plugged into the MINpCUT based model presented in Chapter 4 or used to derive a dissimilarity measure for the PMP based model presented in Chapter 3. In either case, the cells made using this similarity measure will lead to the cells where the (machine-dependent) part families are as similar as possible. This implies that the orders in which parts (families) are processed at each machine in a cell can be similar (in the best case – identical). This becomes especially beneficial if cells have prominent line flows (identification of such flows in considered in Section 5.2), as similar schedules on machines in a line ensure that set-up time savings can be made at each machine

[1] Scheduling several interacting machines remains a complex problem even if each machine has a single part family due to precedence constraints

in this line.

In case set-up times of part families at a machine vary a lot, it becomes necessary to adjust the similarity measure such that the families with larger set-up times can be better scheduled. This can be achieved by modifying the similarity measure (5.10) in the following way:

$$s(i,j) = \sum_{k=1}^{r} \sum_{f=1}^{F} F_{k,f,i} \cdot F_{k,f,j} \cdot (S_{i,f} + S_{j,f}), \tag{5.11}$$

where $S_{i,f}$ and $S_{j,f}$ are set-up times of family f on machines i and j, correspondingly.

5.5 Concluding remarks

This chapter presented several possible objectives that can be optimised while creating cells. This list is, of course, not exhaustive, as a particular manufacturing system may need an objective not relevant to other systems. We tried to cover the most common objectives and to show how these can be incorporated into the models proposed in the previous chapters.

An important question not covered in the chapter is how to combine several possible objectives in one formulation. In fact, the range of possibilities is quite broad and here we would like to mention only few most widely used. First of all, objectives can be combined in a linear way with weighting coefficients reflecting a relative importance of each one. Though this way is the easiest one, it may not cover all optimal solutions. In particular, this happens if the Pareto-optimal front (the set of non-dominated solutions) of a particular multiobjective problem is non-convex. Furthermore, the choice of weights is not straightforward, especially if the objective values are measured in different units (e.g., a number of parts vs. a number of additional workers).

If objectives can be arranged in a linear order of importance, one may perform a sequential optimisation by improving one objective at a time and adding a constraint requiring the value of this objective to stay within certain limits. A similar approach is to simultaneously move some objectives to constraints. For example, one may be interested in maximising the possibilities for set-up time reduction while keeping intercell movement and cross-training costs below certain levels.

Finally, decision making techniques (e.g., TOPSIS; see Ahi et al., 2009) may be applied in order to select the best solution from several ones corresponding to varying weights or sequences in which objectives were optimised.

We would like to conclude by saying that the choice of objectives and the ways of combining them in a mathematical formulation essentially depends on particular goals and motivation for switching to a cellular layout at a particular company. Thus, we may leave these questions to management. At the same time, a methodology for estimating appropriateness and importance of different objectives based on given manufacturing data can be a topic for future research.

Chapter 6

Summary and conclusions

6.1 Summary

The thesis is focused on relevant and effective mathematical models for solving the cell formation (CF) problem, i.e. grouping machines into manufacturing cells such that the principles of group technology are implemented. Despite its long history and hundreds of published papers, very few attempts of solving the problem to optimality are known. At the same time, in today's highly competitive environment any noticeable improvement in performance is critical to the company's survival.

As can be seen from the literature review presented in Chapter 1, most of the available approaches to cell formation are based on intuitive considerations and incorporate at least one of the two error types: the modelling error (the objective function of the model does not exactly reflect the objective of CF) and the computation error (emerges if a resulting problem is solved heuristically). Even if only the modelling error takes place, its quantitative analysis is very complicated (e.g., in case of neural network approaches). Another problem of the existing approaches is flexibility: a substantial portion of them is based on ad hoc algorithms, and addition of new constraints or objectives requires a substantial modification of the approach. Finally, different performance and similarity measures are used, and the effect on the outcomes is not clear (they are motivated by intuitive considerations rather than by strict reasoning). The rest of the thesis presents two models for CF, based on the p-Median and the minimum multicut problems, respectively. The first one has zero computing error while keeping the modelling error and running times very limited. The second one is an exact model and can be solved to optimality only for reasonably sized (yet realistic) instances, as shown by a case study.

Chapter 2 is completely focused on the p-Median problem (PMP) and its properties. It is shown that by using a pseudo-Boolean representation of PMP it is possible to construct an efficient MILP formulation that includes all known problem size reductions for PMP (not relying on pre-solving the problem). The proposed formulation allows some large size instances to be solved to optimality. In particular, by efficiently reducing the problem size the proposed formulation allows a MILP solver to handle instances that could not be handled in earlier formulations because of the memory limitations. A pseudo-Boolean representation also provides insights into a complexity of instance data and properties of the PMP feasible polytope. A methodology for constructing PMP instances that are expected to be complex for any solution algorithm (existing or forthcoming) is described.

In Chapter 3 a model based on the proposed compact formulation for the PMP is presented and analysed in detail. It is shown that PMP based models have quite limited modelling error in case reasonable from the manufacturing perspective cells are possible, i.e. the amount of intercell movement is within 10-15%. A comparison with approaches from several recent papers was done. The results of the comparison show that our PMP based model outperforms other contemporary approaches in solution quality (in terms of widely used performance measures) and has very short running times (about 1 sec.). It is also shown that a number of additional realistic factors and constraints can be introduced into the compact model making it practically useful.

In Chapter 4 an exact model for CF that minimises the amount of intercell movement is derived. It is shown that the exact model is equivalent to the minimum multicut problem (that we abbreviate as MINpCUT), implying polynomial solvability of the former in case two cells are needed. Though there exist efficient algorithms for MINpCUT, these are hardly applicable to CF because they do not allow for additional constraints that are almost always needed to make practically feasible cells. Therefore, we propose two MILP formulations for the MINpCUT problem and show how different constraints can be introduced into either of them. Finally, a practical applicability of the proposed model is demonstrated by means of a case study with real manufacturing data. It is shown that in case of a reasonable number of machines (about 30) the proposed MILP formulations can be solved reasonably fast, and the solution time is not affected by the number of parts that can be quite substantial (the order of thousands). It is also shown that to guarantee optimal solutions (w.r.t. minimum intercell movement), the similarity between a pair of machines must be defined as an amount of parts travelling directly between these two machines.

Finally, in Chapter 5 several alternative objectives for CF are discussed. These include balancing inter- and intracell movements, workforce related objectives, and set-up time reduction. Ways of introducing these objectives into the proposed models are given, together with a brief discussion about how to combine several objectives. It is also proven that none of the similarity measures derived from the machine-part incidence matrix can guarantee optimal solutions, irrespectively of the approach used.

6.2 Conclusions

The goal of this thesis was to provide a flexible and efficient tool for solving the CF problem. In order to achieve the goal two models based on two different but related graph-partitioning problems, PMP and MINpCUT, were developed. Either of the proposed models proved to be efficient in solving the CF problem, but has its own weaknesses. For example, the PMP based model is extremely fast but introduces a (limited) modelling error. The MINpCUT based model is an exact one and is somewhat more flexible, but has a somewhat limited applicability in terms of the problem size. The effectiveness of the two proposed models can be explained by the fact that all the three problems (PMP, MINpCUT and the generic CF problem) have a common clustering nature. Yet, there are certain differences in constraints imposed on the clusters. For example, in case of the PMP each cluster is supposed to have a prominent centre (median) that is tightly connected with any other element in the cluster. Presence of this structure, in particular, makes the problem much easier – even for random 100×100 input matrices it can be solved within a minute. The MINpCUT problem, on the contrary, does not have any special constraints on clusters, and solving it for a 100×100 matrix normally takes many hours. The original CF problem normally does not restrict the structure of the cells[1] but imposes certain qualitative constraints. This makes the problem even more complex. However, the complexity of the problem can be made dependent only on the number of machines, unlike, for example, the model from (Chen & Heragu, 1999) where the number of parts also plays a role. In this case, it appears that real life instances have quite limited size (we could not find instances with more than 50 machines in the literature). This means that quite soon the ongoing progress in a develop-

[1] Sometimes it does: some machines may be required or prohibited to be in one cell. In this case some variables can be fixed and the problem becomes smaller.

ment of computers and MILP solution methods will make it possible to solve any CF instance to optimality by the proposed MINpCUT model. On the other hand, the size of manufacturing systems is not increasing with a trend towards smaller and more specialised ones. Thus, in this perspective, both proposed models are computationally tractable.

Possible directions for future research include development of more efficient methods for solving the constrained MINpCUT problem and extending either of the proposed models with real life constraints not mentioned in the thesis. Another direction is to introduce dynamics into the models: in reality, the product mix changes over time, and cells must be adjusted. At the same time, it is desirable that the cells do not change substantially from period to period. Thus, either a robust (with regard to a changing product mix) or a dynamic (with smooth changes) solution can be sought.

A final avenue of research is the development of a methodology for estimating appropriateness and importance of different objectives for CF based on manufacturing data.

References

Adil, G. K. & Rajamani, D. (2000). The tradeoff between intracell and intercell moves in group technology cell formation. *J. Manuf. Syst.*, *19*(5), 305-317.

Ahi, A., Aryanezhad, M., Ashtiani, B. & Makui, A. (2009). A novel approach to determine cell formation, intracellular machine layout and cell layout in the CMS problem based on TOPSIS method. *Comput. Oper. Res.*, *36*(5), 1478 - 1496.

Albadawi, Z., Bashir, H. A. & Chen, M. (2005). A mathematical approach for the formation of manufacturing cells. *Comput. Ind. Eng.*, *48*, 3 – 21.

AlBdaiwi, B., Ghosh, D. & Goldengorin, B. (2011). Data aggregation for p-median problems. *J. Comb. Optim.*, *21*, 348363.

AlBdaiwi, B., Goldengorin, B. & Sierksma, G. (2009). Equivalent instances of the simple plant location problem. *Comput. Math. Appl.*, *57*, 812-820.

Ashayeri, J., Heuts, R. & Tammel, B. (2005). A modified simple heuristic for the p-median problem, with facilities design applications. *Robot. Com-Int. Manuf.*, *21*(4-5), 451-464.

Askin, R. & Standridge, C. (1993). *Modeling and analysis of manufacturing systems*. New York: Wiley and Sons.

Avella, P. & Sassano, A. (2001). On the *p*-median polytope. *Math. Program.*, *89*, 395-411.

Avella, P., Sassano, A. & Vasil'ev, I. (2007). Computational study of large-scale *p*-median problems. *Math. Program.*, *109*, 89-114.

Avella, P. & Sforza, A. (1999). Logical reduction tests for the *p*-median problem. *Ann. Oper. Res.*, *86*, 105-115.

Balakrishnan, J. & Cheng, C. H. (2007). Multi-period planning and uncertainty issues in cellular manufacturing: A review and future directions. *Eur. J. Oper. Res.*, *177*, 281309.

Beasley, J. (1985). A note on solving large *p*-median problems. *Eur. J. Oper. Res.*, *21*, 270-273.

Belenky, A. (2008). Mathematical modeling of voting systems and elections: Theory and applications. *Math. Comput. Model.*, *48*(9-10), 1295-1676.

Beltran, C., Tadonki, C. & Vial, J.-P. (2006). Solving the p-median problem with a semi-lagrangian relaxation. *Comput. Optim. Appl.*, *35*, 239260.

Benjaafar, S. & Sheikhzadeh, M. (2000). Design of flexible plant layouts. *IIE Trans.*, *32*, 309-322.

Beresnev, V. L. (1973). On a problem of mathematical standardization theory. *Upravliajemyje Sistemy*, *11*, 43-54. (in Russian)

Bhatnagar, R. & Saddikuti, V. (2010). Models for cellular manufacturing systems design: matching processing requirements and operator capabilities. *J. Opl. Res. Soc.*, *61*, 827839.

Bokhorst, J. A. C., Slomp, J. & Molleman, E. (2004). Development and evaluation of cross-training policies for manufacturing teams. *IIE Trans.*, *36*(10), 969-984.

Boros, E. & Hammer, P. L. (2002). Pseudo-boolean optimization. *Discrete Appl. Math.*, *123*, 155–225.

Boulif, M. & Atif, K. (2006, Oct.). An exact multiobjective epsilon-constraint approach for the manufacturing cell formation problem. In *Service systems and service management, 2006 International Conference on* (Vol. 2, p. 883-888).

Briant, O. & Naddef, D. (2004). The optimal diversity management problem. *Oper. Res.*, *52*, 515-526.

Brusco, M. J. & Köhn, H.-F. (2008). Optimal partitioning of a data set based on the p-median problem. *Psychometrika*, *73*(1), 89105.

Burbidge, J. L. (1961). The new approach to production. *Prod. Eng.*, *December*, 3 - 19.

Burbidge, J. L. (1991). Production flow analysis for planning group technology. *J. Oper. Manag.*, *10*(1), 5-27.

Burlet, M. & Goldschmidt, O. (1997). A new and improved algorithm for the 3-cut problem. *Oper. Res. Lett.*, *21*, 225-227.

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. *Int. J. Prod. Res.*, *46*(10), 25892620.

Chan, H. M. & Milner, D. A. (1982). Direct clustering algorithm for group formation in cellular manufacture. *J. Manuf. Syst.*, *1*(1), 65-75.

Chandra, C., Irani, S. A. & Arora, S. R. (1993). Clustering effectiveness of permutation generation heuristics for machine-part matrix clustering. *J. Manuf. Syst.*, *12*(5), 338-408.

Chandrasekharan, M. P. & Rajagopalan, R. (1986a). An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. *Int. J. Prod. Res.*, *24*(2), 451 - 463.

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

Chen, J.-S. & Heragu, S. S. (1999). Stepwise decomposition approaches for large scale cell formation problems. *Eur. J. Oper. Res.*, *113*, 64-79.

Christofides, N. (1975). *Graph theory: An algorithmic approach*. London: Academic Press Inc. Ltd.

Chu, C.-H. & Hayya, J. C. (1991). Fuzzy clustering approach to manufacturing cell formation. *Int. J. Prod. Res.*, *29*(7), 1475-1487.

Church, R. L. (2003). COBRA: a new formulation of the classic p-median location problem. *Ann. Oper. Res.*, *122*, 103-120.

Church, R. L. (2008). BEAMR: An exact and approximate model for the p-median problem. *Comput. Oper. Res.*, *35*, 417-426.

Cornuejols, G., Nemhauser, G. & Wolsey, L. A. (1980). A canonical representation of simple plant location problems and its applications. *SIAM Journal on Matrix Analysis and Applications (SIMAX)*, *1*(3), 261-272.

Cornuejols, G., Nemhauser, G. & Wolsey, L. A. (1990). The uncapacitated facility location problem. In P. Mirchandani & R. L. Francis (Eds.), *Discrete location theory.* New York: Wiley-Interscience.

Dearing, P., Hammer, P. L. & Simeone, B. (1992). Boolean and graph theoretic formulations of the simple plant location problem. *Transport. Sci.*, *26*(2), 138–148.

Deutsch, S. J., Freeman, S. F. & Helander, M. (1998). Manufacturing cell formation using an improved p-median model. *Comput. Ind. Eng.*, *34*(1), 135 - 146.

DiMaggio, P. A., McAllister, S. R., Floudas, C. A., Feng, X. J., Rabinowitz, J. D. & Rabitz, H. A. (2008). Biclustering via optimal re-ordering of data matrices in systems biology: rigorous methods and comparative studies. *BMC Bioinformatics*, *9*(1), 458.

Doulabi, S. H. H., Hojabri, H., Seyed-Alagheband, S.-A., Jaafari, A. A. & Davoudpour, H. (2009, Oct.). Two-phase approach for solving cell-formation problem in cell manufacturing. In *Engineering and computer science WCECS, 2009 World Congress on* (p. 1226 - 1231). San Francisco, USA.

Elloumi, S. (2010). A tighter formulation of the *p*-median problem. *J. Comb. Optim.*, *19*, 69-83.

Fallah-Alipour, K. & Shamsi, R. (2008). A mathematical model for cell formation in CMS using sequence data. *J. Ind. Syst. Eng.*, *2*(2), 144-153.

Filho, E. V. G. & Tiberti, A. J. (2006). A group genetic algorithm for the machine cell formation problem. *Int. J. Prod. Econ.*, *102*, 1-21.

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

Garey, M. R. & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of np-completeness*. USA: W. H. Freeman.

Gindy, N. N. Z., Ratchev, T. M. & Case, K. (1995). Component grouping for gt applicationsa fuzzy clustering approach with validity measure. *Int. J. Prod. Res.*, *33*(9), 2493-2509.

Goldengorin, B. & Krushinsky, D. (2011a). Complexity evaluation of benchmark instances for the p-median problem. *Math. Comput. Model.*, *53*, 1719-1736.

Goldengorin, B. & Krushinsky, D. (2011b). A computational study of the pseudo-Boolean approach to the p-median problem applied to cell formation. *Lect. Notes Comput. Sci.*, *6701*, 503-516.

Goldengorin, B., Krushinsky, D. & Slomp, J. (2012). Flexible PMP approach for large size cell formation. *Oper. Res.*. (second round review)

Goldengorin, B., Tijssen, G. A., Ghosh, D. & Sierksma, G. (2003). Solving the simple plant location problems using a data correcting approach. *J. Global Optim.*, *25*, 377-406.

Goldschmidt, O. & Hochbaum, D. S. (1994). A polynomial algorithm for the k-cut problem for fixed k. *Math. Oper. Res.*, *19*(1), 24-37.

Gower, J. C. & Ross, G. J. S. (1969). Minimum spanning trees and single linkage cluster analysis. *Applied Statistics*, *18*(1), 54-64.

Guerrero, F., Lozano, S., Smith, K. A., Canca, D. & Kwok, T. (2002). Manufacturing cell formation using a new self-organizing neural network. *Comput. Ind. Eng.*, *42*(2-4), 377-382.

Gutin, G., Razgon, I. & Kim, E. J. (2008). Minimum leaf out-branching and related problems. *Lect. Notes Comput. Sci.*, *5034*, 235–246.

Hakimi, S. L. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Oper. Res.*, *12*, 450459.

Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Oper. Res.*, *13*, 462475.

Hammer, P. L. (1968). Plant location – a pseudo-boolean approach. *Israel Journal of Technology*, *6*, 330-332.

Hsu, C. P. (1990). *Similarity coefficient approaches to machine-component cell formation in cellular manufacturing. a comparative study*. Unpublished doctoral dissertation, Department of Industrial and Systems Engineering. University of Wisconsin, Milwaukee, USA.

Kao, Y. & Moon, Y. B. (1991). A unified group technology implementation using the backpropagation learning rule of neural networks. *Comput. Ind. Eng.*, *20*(4), 425-437.

Kaparthi, S. & Suresh, N. C. (1992). Machine-component cell formation in group technology: a neural network approach. *Int. J. Prod. Res.*, *30*(6), 1353-1367.

Kariv, O. & Hakimi, S. L. (1979). An algorithmic approach to network location problems. II: The p-medians. *SIAM J. Appl. Math.*, *37*, 539-560.

Keeling, K. B., Brown, E. C. & James, T. L. (2007). Grouping efficiency measures and their impact on factory measures for the machine-part cell formation problem: A simulation study. *Eng. Appl. Artif. Intel.*, *20*, 63-78.

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

Koskosidis, Y. & Powell, W. (1992). Clustering algorithms for consolidation of customer orders into vehicle shipments. *Transport. Res.*, *26B*, 365379.

Krushinsky, D. & Goldengorin, B. (2012). An exact model for cell formation in group technology. *Comput. Manag. Sci.*. (second round review)

Kumar, C. S. & Chandrasekharan, M. P. (1990). 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.

Kusiak, A. (2000). *Computational intelligence in design and manufacturing*. New York, USA: Wiley-Interscience.

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

Kusiak, A. & Chow, W. S. (1988). Decomposition of manufacturing systems. *IEEE Journal of Robotics and Automation*, *4*(5), 457-471.

Liang, M. & Zolfaghari, S. (1999). Machine cell formation considering processing times and machine capacities: an ortho-synapse Hopfield neural network approach. *J. Intell. Manuf.*, *10*, 437-447.

Library OR. (1990). (Available at the web address http://people.brunel.ac.uk/~mastjjb/jeb/orlib/pmedinfo.html)

Library TSP. (1995). (Available at the web address http:// www.iwr.uni-heidelberg.de/ groups/ comopt/ software/ TSPLIB95/)

Madeira, S. C. & Oliveira, A. L. (2004). Biclustering algorithms for biological data analysis: A survey. *IEEE-ACM T Comput BI*, *1*(1), 24-45.

Mak, K. L., Wong, Y. S. & Wang, X. X. (2000). An adaptive genetic algorithm for manufacturing cell formation. *Int. J. Adv. Manuf. Tech.*, *16*, 491-497.

Malave, C. O. & Ramachandran, S. (1991). Neural network-based design of cellular manufacturing systems. *J. Intell. Manuf.*, *2*(5), 305-314.

McAuley, J. (1972). Machine grouping for efficient production. *Prod. Eng.*, *51*(2), 53-57.

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

Miltenburg, J. & Zhang, W. (1991). A comparative evaluation of nine well-known algorithms for solving the cell formation problem in group technology. *J. Oper. Manag.*, *10*(1), 44-72.

Mitrofanov, S. P. (1946). *Scientific principles of group technology*. Leningrad: Leningrad University. (in Russian)

Mitrofanov, S. P. (1959). *Nauchnie osnovi gruppovoy tehnologii*. USSR: Lenizdat. (in Russian)

Mitrofanov, S. P. (1966). *Scientific principles of group technology, Part I*. Boston, MA: National Lending Library of Science and Technology.

Mladenovic, N., Brimberg, J., Hansen, P. & Moreno-Perez, J. A. (2007). The p-median problem: A survey of metaheuristic approaches. *Eur. J. Oper. Res.*, *179*(3), 927-939.

Montreuil, B., Venkatadri, U. & Rardin, R. L. (1999). Fractal layout organization for job shop environments. *Int. J. Prod. Res.*, *37*(3), 501-521.

Mulvey, J. & Beck, M. P. (1984). Solving capacitated clustering problems. *Eur. J. Oper. Res.*, *18*, 339-348.

Nair, G. J. & Narendran, T. T. (1998). CASE: A clustering algorithm for cell formation with sequence data. *Int. J. Prod. Res.*, *36*(1), 157-180.

Narayanaswamy, P., Bector, C. R. & Rajamani, D. (1996). Fuzzy logic concepts applied to machine-component matrix formation in cellular manufacturing. *Eur. J. Oper. Res.*, *93*(1), 88-97.

Ng, S. M. (1991). Bond energy, rectilinear distance and a worst-case bound for the group technology problem. *J. Opl. Res. Soc.*, *42*(7), 571-578.

Ng, S. M. (1993). Worst-case analysis of an algorithm for cellular manufacturing. *Eur. J. Oper. Res.*, *69*, 384-398.

Ng, S. M. (1996). On the characterization and measure of machine cells in group technology. *Oper. Res.*, *44*(5), 735-744.

Owsinski, J. (2009). Machine-part grouping and cluster analysis: similarities, distances and grouping criteria. *Bull. Pol. Ac.: Tech.*, *57*(3), 217-228.

Papadimitrou, C. & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*. Mineola, New York, USA: Dover.

Park, S. & Suresh, N. C. (2003). Performance of fuzzy ART neural network and hierarchical clustering for partmachine grouping based on operation sequences. *Int. J. Prod. Res.*, *41*(14), 3185 3216.

Pentico, D. W. (2008). The assortment problem: A survey. *Eur. J. Oper. Res.*, *190*, 295-309.

Pirkul, H. (1987). Efficient algorithms for the capacitated concentrator location problem. *Comput. Oper. Res.*, *14*(3), 197-208.

Rajagopalan, R. & Batra, L. (1975). Design of cellular production systems: a graph theoretic approach. *Int. J. Prod. Res.*, *13*(6), 567-579.

Ravi, R. & Sinha, A. (2008). Approximating k-cuts using network strength as a lagrangean relaxation. *Eur. J. Oper. Res.*, *186*, 77-90.

Reese, J. (2006). Solution methods for the p-median problem: An annotated bibliography. *Networks*, *48*(3), 125-142.

Resende, M. & Werneck, R. (2003, Jan.). On the implementation of a swap-based local search procedure for the p-median problem. In R. Ladner (Ed.), *Algorithm engineering and experiments (ALENEX'03), 2003 Fifth Workshop on* (p. 119-127). Baltimore, USA: SIAM.

ReVelle, C. S., Eiselt, H. A. & Daskin, M. S. (2008). A bibliography for some fundamental problem categories in discrete location science. *Eur. J. Oper. Res.*, *184*, 817-848.

ReVelle, C. S. & Swain, R. (1970). Central facilities location. *Geogr. Anal.*, *2*, 30-42.

Robertson, N. & Seymour, P. (1995). Graph minors. XIII. The disjoint path problem. *J. Comb. Theory*, *B 63*, 65-110.

Rosing, K. E., ReVelle, C. S. & Rosing-Vogelaar, H. (1979). The p-median and its linear programming relaxation: An approach to large problems. *J. Opl. Res. Soc.*, *30*, 815822.

Saran, H. & Vazirani, V. (1995). Finding k-cuts within twice the optimal. *SIAM J. Comput.*, *24*(1), 101-108.

Sarker, B. R. (2001). Measures of grouping efficiency in cellular manufacturing systems. *Eur. J. Oper. Res.*, *130*, 588-611.

Schrijver, A. (2003). *Combinatorial optimization. Polyhedra and efficiency*. Berlin, Germany: Springer.

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

Senne, E. L. F., Lorena, L. A. N. & Pereira, M. A. (2005). A branch-and-price approach to p-median location problems. *Comput. Oper. Res.*, *32*, 1655-1664.

Shafer, S. M. & Rogers, D. F. (1993). Similarity and distance measures for cellular manufacturing. part I. A survey. *Int. J. Prod. Res.*, *31*(5), 1133-1142.

Slomp, J., Chowdary, B. V. & Suresh, N. C. (2005). Design of virtual manufacturing cells: a mathematical programming approach. *Robot. Com-Int. Manuf.*, *21*, 273-288.

Suresh, N. C. (1992). Partitioning work centers for group technology: analytical extension and shop-level simulation investigation. *Decision Sci.*, *23*, 267-290.

Suresh, N. C. & Slomp, J. (2001). A multi-objective procedure for labour assignments and grouping in capacitated cell formation problems. *Int. J. Prod. Res.*, *39*(18), 41034131.

Suresh, N. C., Slomp, J. & Kaparthi, S. (1999). Sequence-dependent clustering of parts and machines: a fuzzy ART neural network approach. *Int. J. Prod. Res.*, *37*(12), 2793-2816.

Tharumarajah, A., Wells, A. J. & Nemes, L. (1996). Comparison of the bionic, fractal and holonic manufacturing system concepts. *Int. J. Comput. Integ. M.*, *9*(3), 217-226.

Venugopal, V. & Narendran, T. T. (1994). Machine-cell formation through neural network models. *Int. J. Prod. Res.*, *32*(9), 2105-2116.

Vin, E. (2010). *Genetic algorithm applied to generalized cell formation problems*. Unpublished doctoral dissertation, Université Libre de Bruxelles, Faculté des Sciences Appliquées.

Wang, J. & Roze, C. (1997). Formation of machine cells and part families: a modified p-median model and a comparative study. *Int. J. Prod. Res.*, *35*(5), 1259-1286.

Wei, J. C. & Kern, G. M. (1989). Commonality analysis. A linear cell clustering algorithm for group technology. *Int. J. Prod. Res.*, *27*(12), 2053-2062.

Wemmerlov, U. & Hyer, N. L. (1986). Procedures for the part family/machine group identification problem in cellular manufacturing. *J. Oper. Manag.*, *6*(2), 125-147.

Wolsey, L. (2008). Mixed integer programming. In B. Wah (Ed.), *Wiley encyclopedia of computer science and engineering*. UK: John Wiley & Sons, Inc.

Won, Y. & Currie, K. R. (2006). An effective p-median model considering production factors in machine cell/part family formation. *J. Manuf. Syst.*, *25*(1), 58-64.

Won, Y. & Lee, K. C. (2004). Modified p-median approach for efficient GT cell formation. *Comput. Ind. Eng.*, *46*(3), 495-510.

Xambre, A. R. & Vilarinho, P. M. (2003). A simulated annealing approach for manufacturing cell formation with multiple identical machines. *Eur. J. Oper. Res.*, *151*(2), 434-446.

Xu, H. & Wang, H.-P. B. (1989). Part family formation for GT applications based on fuzzy mathematics. *Int. J. Prod. Res.*, *27*(9), 1637-1651.

Yang, M.-S. & Yang, J.-H. (2008). Machine-part cell formation in group technology using a modified ART1 method. *Eur. J. Oper. Res.*, *188*(1), 140 - 152.

Yin, Y. & Yasuda, K. (2006). Similarity coefficient methods applied to the cell formation problem: A taxonomy and review. *Int. J. Prod. Econ.*, *101*(2), 329 - 352.

Samenvatting

De moderne industrie heeft te maken met een aantal uitdagingen, zoals groeiende diversificatie van producten en toenemende concurrentie. Deze ontwikkelingen dwingen bedrijven hun processen voor zover mogelijk effectiever te maken. De indeling van machines op de werkvloer is heel belangrijk want deze bepaalt onder andere de doorlooptijd van producten. Dit komt vooral door de koppeling tussen de indeling van de machines en de afstanden die door de producten worden afgelegd. Daarnaast spelen wachttijden en voorraden ook een rol. Een juiste indeling van machines kan voordelen bieden bij het maken van roosters (schedules). Als machines bijvoorbeeld in (bijna) onafhankelijke groepen kunnen worden ingedeeld, is het mogelijk voor elke groep afzonderlijk een rooster te construeren. Dit is voordelig omdat de complexiteit van de meeste roostering problemen zeer snel toeneemt naarmate de invoer groter wordt.

Zoals uit de literatuur blijkt, heeft de groepsgewijze indeling (in cellen) een aantal voordelen en is deze geschikt voor de meeste bedrijven, behalve in extreme situaties waarin sprake is van een zeer lage of juist heel hoge mate van productdiversificatie. In deze gevallen zijn de “production line” of de “functional layout”meer geschikt. Om bijbehorende voordelen te realiseren moet de indeling in groepen op de juiste manier plaatsvinden. Dat betekent vooral dat de cellen voor zover mogelijk onafhankelijk zijn (elke cel produceert zijn eigen verzameling van producten, een *productfamilie*, die bijna geen machines uit de andere cellen nodig hebben), maar er zijn ook andere factoren en beperkingen. Zo moeten het aantal machines in een cel, het aantal werknemers per cel en het aantal productie-uren per cel gebalanceerd zijn. Het indelen van machines in groepen (cellen) wordt *celformatie* (CF) genoemd. Het CF probleem deelt machines in groepen in zodat het aantal productbewegingen tussen verschillende cellen (*intercell movements*) minimaal is. Het CF probleem is dus gericht op optimale celformatie. Hoewel gedurende meer dan 50 jaar veel onderzoek is gedaan naar dit probleem, is er in de literatuur geen methode of model bekend die tegelijkertijd de volgende kenmerken heeft:

- gegarandeerde kwaliteit van oplossingen (geen of aantoonbaar gelimiteerd aantal fouten);

- redelijke rekentijden (rekentijden van bijvoorbeeld 1 seconde en 10 minuten zijn even goed aangezien cellen niet elke dag worden veranderd);

- flexibiliteit: bij elke bedrijf komen andere beperkingen voor, waarmee de methode (het model) rekening moet kunnen houden.

Dit proefschrift concentreert zich op de ontwikkeling van relevante mathematische modellen voor celformatie, die rekening houden met bovenstaande doelstellingen en in de praktijk kunnen worden gebruikt om bedrijfsprestaties te verbeteren.

Hoofdstuk 1 introduceert het CF probleem, geeft een beschouwing van de literatuur en beschrijft hiaten die worden ingevuld in dit proefschrift. Zoals het uit de literatuur blijkt, baseren de meeste CF-methoden zich op intuïtieve consideraties en kampen met ten minste één van de volgende fouten:

(a) modelleringsfout (*modelling error*); hiervan is sprake als de doelstellingsfunctie van een model het doel van CF niet juist weergeeft;

(b) rekenfout (*computation error*); deze komt voor als het probleem met behulp van een heuristiek wordt opgelost.

Zelfs als er alleen sprake is van een modelleringsfout, is de kwantitatieve analyse normaal gesproken al zeer complex (denk bijvoorbeeld aan neurale netwerkmodellen). Een ander nadeel van de meeste bestaande modellen is hun gebrek aan flexibiliteit: als een model op een ad hoc procedure is gebaseerd, veroorzaakt elke nieuwe beperking een belangrijke wijziging van het model. Tenslotte wordt gebruik gemaakt van verschillende performance en similarity measures, zonder dat de keuze voor een bepaalde measure duidelijk wordt onderbouwd. Daardoor is het gebruik van een bepaalde measure dubbelzinnig en is de invloed op het resultaat vaak onduidelijk. In het resterende deel van dit proefschrift worden twee modellen ontwikkeld die gebaseerd zijn op het p-Median en het minimum multicut problemen. In het eerste model is slechts sprake van een beperkte modelleringsfout terwijl er geen rekenfouten optreden en de rekentijden kort zijn. Het tweede model is exact, maar met dit model kunnen alleen kleine en middelgrote (maar desalniettemin realistische) CF problemen worden opgelost. De praktische toepasbaarheid van dit model wordt door een case study aangetoond.

Hoofdstuk 2 concentreert zich op het p-Median probleem (PMP) en zijn eigenschappen. Een pseudo-Booleaanse representatie is gebruikt om een doeltreffende Mixed-Integer Linear Programming (MILP) formulering op te bouwen die alle bekende reducties (behalve pre-solving) bevat. Met dit model kan de optimale oplossing van een aantal grote problemen worden bepaald. Doordat de probleemgrootte door de nieuwe formulering gereduceerd wordt is een aantal problemen, dat tot op heden vanwege beperkingen van het beschikbare geheugen van een computer niet kon worden opgelost, nu behapbaar geworden voor een MILP solver, bijvoorbeeld CPLEX of Xpress. Een pseudo-Booleaanse representatie geeft bovendien meer inzicht in de complexiteit van input data van verschillende instanties en van het PMP toelaatbare polytoop. In dit hoofdstuk wordt voorts een methodiek voorgesteld om PMP instanties te genereren die moeilijk zijn voor alle mogelijke algoritmen (zowel bestaande als nog te ontwikkelen algoritmen).

In hoofdstuk 3 wordt een op het PMP gebaseerd model gepresenteerd. Het model maakt gebruik van de doeltreffende formulering uit het voorgaande hoofdstuk en heeft korte rekentijden (plusminus 1 seconde). Het wordt aangetoond dat de modelleringsfout voor op PMP gebaseerde modellen beperkt is (hoogstens enkele procenten) wanneer de input data een praktische waarde hebben: de omvang van het aantal intercell movements ligt rond de 10 a 15%. Het hoofdstuk bevat ook een studie waarin methoden die recent in de literatuur beschreven zijn worden vergeleken. Hieruit blijkt dat ons model betere oplossingen levert in termen van de meest gebruikte performance measures. We bewijzen ook dat andere realistische factoren en beperkingen in het model kunnen worden verwerkt, zodat de praktische waarde ervan verhoogd wordt.

In hoofdstuk 4 wordt een exact model voor CF afgeleid, met als doel de omvang van het aantal intercell movements te minimaliseren. Het wordt aangetoond dat het exacte model equivalent is aan het minimum multicut probleem (dat we afkorten als MINpCUT). Dit feit impliceert dat het CF probleem makkelijk kan worden opgelost (polynomially solvable) als er maar twee cellen nodig zijn. Hoewel er effectieve methoden voor MINpCUT bestaan, hebben die weinig waarde voor de CF praktijk aangezien ze geen mogelijkheden bieden om extra beperkingen toe te voegen. Daarom worden twee MILP formuleringen voor het MINpCUT probleem beschreven. Tevens wordt geïllustreerd hoe een aantal praktische beperkingen kan worden opgenomen in elk van deze twee modellen. De toepasbaarheid van de MINpCUT-gebaseerde methode is met een case study geïllustreerd. Het belangrijkste kenmerk van deze methode is dat haar complexiteit alleen van het aantal machines (tientallen) afhankelijk is en niet van het aantal producten (duizenden). Het wordt ook aangetoond dat om optimale oplossingen te garanderen (met betrekking tot het aantal intercell movements) het aantal bewegingen van onderdelen tussen een tweetal machines gebruikt moet worden als similarity measure van de desbetreffende machines.

Hoofdstuk 5 richt zich op alternatieve doelen van CF, zoals het balanceren van inter- en intracel bewegingen, arbeidsgerelateerde doelen, en reductie van de installatietijden (set-up times). Een uiteenzetting wordt gegeven van mogelijke manieren om deze doelen aan alle bovengenoemde modellen toe te voegen en het wordt besproken hoe meervoudige doelen in een MILP formulering kunnen worden opgenomen. Voorts wordt aangetoond dat similarity measures die gebaseerd zijn op de "machine-part incidence matrix" geen optimale oplossingen kunnen garanderen, onafhankelijk van de methode die gebruikt wordt.

Hoofdstuk 6 resumeert tot slot de onderzoeksresultaten van dit proefschrift en formuleert enkele suggesties voor toekomstig onderzoek.