View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by LSE Theses Online

The London School of Economics and Political Science

On the completability of mutually orthogonal Latin rectangles

by

Anastasia Kouvela

A thesis submitted to the Management Science Group of
the London School of Economics and Political Science
for the degree of
Doctor of Philosophy

London, August 2013

Declaration

I certify that the thesis I have presented for examination for the MPhil/PhD degree of the London School of Economics and Political Science is solely my own work other than where I have clearly indicated that it is the work of others (in which case the extent of any work carried out jointly by me and any other person is clearly identified in it).

The copyright of this thesis rests with the author. Quotation from it is permitted, provided that full acknowledgement is made. This thesis may not be reproduced without my prior written consent.

I warrant that this authorisation does not, to the best of my belief, infringe the rights of any third party.

I declare that my thesis consists of 138 pages (including bibliography).

Declaration

I confirm that Chapter 7 was jointly co-authored with Professor Frits C.R. Spieksma and Dr Trivikram Dokka and I contributed 50% of this work.

Abstract

This thesis examines the completability of an incomplete set of m-row orthogonal Latin rectangles (MOLRm) from a set theoretical viewpoint. We focus on the case of two rows, i.e. MOLR2, and define its independence system (IS) and the associated clutter of bases, which is the collection of all MOLR2. Any such clutter gives rise to a unique clutter of circuits which is the collection of all minimal dependent sets. To decide whether an incomplete set of MOLR2 is completable, it suffices to show that it does not contain a circuit therefore full knowledge of the clutter of circuits is needed. For the IS associated with 2-row orthogonal Latin rectangles (OLR2) we establish a methodology based on the notion of an availability matrix to fully characterise the corresponding clutter of circuits. We prove that the clutter consists of twenty-one non-equivalent circuits and illustrate each one. This work adds to the few IS in the literature for which the clutter of circuits is fully characterised as it is known that unless $\mathcal{P}$=$\mathcal{NP}$ there is no polynomial time algorithm generating the clutter of either bases or circuits of an arbitrary IS. We also present seven non-equivalent circuits for the set of 3 MOLR2 thus providing a partial description of the associated clutter.

After establishing a particular relationship between sets of MOLR2 and Latin rectangles, we prove that completing an incomplete set of $n-1$ MOLR2 is $\mathcal{NP}$-complete. We also show that every set of t MOLR2 can be completed to a set of t' MOLR2 where $t < t' \leq n-1$, and conclude that the clutter of circuits for any set of t MOLR2 is a subset of the clutter of circuits for t' MOLR2.

To address the OLR2 completability problem from an integer programming (IP) perspective, we present a new IP 3-index formulation. After analysing its structure we formulate all twenty-one circuits of OLR2 as constraints and follow a sequential lifting procedure to derive maximally lifted inequalities, valid not only for the OLR2 polytope but also for the much larger mutually orthogonal Latin squares (MOLS) polytope. Next we present two IP formulations for the general MOLRm case and compare their structure. Three constraint programming (CP) formulations are also provided for which a series of redundant models are linked together with use of channelling or inverse constraints, aiming to improve constraint propagation and reduce computational time.

The most recent applications of MOLRm are also discussed, namely in telecommunications and wireless networks for the transmission of data. Latin rectangles are also known to have applications in scheduling and for this we present a variation known as the multi-index bottleneck assignment problem (MBA). For this problem we settle the complexity status and present IP and CP formulations. For fast computational results, we present three heuristic algorithms for particular instances and prove their approximation guarantee.

Acknowledgements

I would like to express my appreciation and gratitude to Professor Gautam Appa for his guidance and support, both as an academic supervisor and as a mentor in life.

I feel privileged to have worked with Dr Dimitris Magos and Dr Yiannis Mourtos during my studies. They have greatly influenced the formation of my scientific thought and perception of academic ethos. I also feel fortunate to have worked closely with Professor Reinhardt Euler and I would like to thank him for being an inspirational and supportive teacher.

Dr Trivikram Dokka and Professor Frits C.R. Spieksma are valuable colleagues and I would like to express my deep gratitude for the stimulating discussions we've had over the past years.

I thank Dr Katerina Papadaki, Dr Giacomo Zambelli and Professor Gregory Sorkin for their constructive feedback and guidance.

The Management Science Department of the London School of Economics and Political Science has been a place full of intelligent and welcoming people. I would like to give special thanks to Brenda Mowlam for her comforting words of reason throughout all these years. I feel fortunate to have spent the endless hours of study next to my fellow PhD students and friends Dr Nayat Horozoglu, Özlem Karsu, Ioannis Fragos and Dr Dimitrios Karamanis and I thank them for their positivity and sense of humour.

None of this would have been possible without Pavlina, Litsa and Konstantinos Kouvelas; my family. Each in their unique way, has been there for every step of the way.

Preface

This thesis considers the problem of completing m-row mutually orthogonal Latin rectangles (MOLRm) of order n and some variations of this problem.

The main inspiration for this topic has been the stimulating combinatorial problem of mutually orthogonal Latin squares (MOLS). Due to their particular symmetrical structure and variety of applications in cryptography, statistical design, scheduling and timetabling, tournament design and other (see [20], [23], [48]), MOLS are extensively studied by mathematicians and computer scientists over the past two centuries. It is well known that certain orders of MOLS, such as a pair of orthogonal Latin squares (OLS) of order 2 and 6 do not exist and there is a wide range of work dating back to L. Euler in 1849 (in [22]) providing proofs for the infeasible 6×6 case ([60], [62], [32], [68], [27], [3]). The next unresolved existence question in this field is the set of 3 MOLS of order 10. It is thought that this does not exist but despite numerous efforts ([53], [43], [50], [40], [21]) its existence remains an open question. Existing approaches for proving non-existence of certain orders of MOLS, rely primarily on exhaustive enumeration of cases and these proofs may deliver better results as computer power improves. However, to our knowledge limited work has been done to understand what is hidden in the structure of orders of MOLS that explain their existence or non-existence.

Our investigation of reasons for non-existence takes a different approach. We address a similar but smaller problem: determining whether an incomplete set of MOLRm of order n $(m < n)$ is completable or, in other words, if there exists a completed set of MOLRm of order n that contains the incomplete one. It is smaller than the general problem of finding a set of MOLS, in the sense that $a)$ there are fewer rows and $b)$ some cells already have values. To solve the MOLRm problem, it suffices to characterise all sets of MOLRm of a particular order that are not completable. Minimal such sets define circuits of the associated independence system (formal definitions appear in Chapter 2) and the collection of circuits is known as the clutter of circuits. Therefore an incomplete set of MOLRm is completable if and only if it contains no circuit and complete knowledge of the clutter of circuits answers the completability question.

There are antecedents to our approach. Mann in [49] provides a necessary but not sufficient condition for the completability of the general OLS case. R. Euler et al. in [30] initiated the study of circuits for the problem of completing Latin squares and this work was continued in [29] and [31] where the completability of 2-row and 3-row Latin rectangles is studied. In this thesis, we extend R. Euler's work to pairs of orthogonal rectangles for the first time. We fully characterise the clutter of circuits for 2-row orthogonal Latin rectangles (OLR2) and also provide a partial description of the clutter of circuits for the set of 3 2-row mutually orthogonal Latin rectangles (MOLR2).

Since there can be no polynomial time algorithm (unless $\mathcal{P}= \mathcal{NP}$) generating the clutter of either bases or circuits of an arbitrary independence system [61], our work adds to the few independence systems in the literature, for which the clutter of circuits is fully characterised (see [47] and [61]). Moreover, deriving the family of circuits associated with an independence system is a special case of the hypergraph transversal problem as studied in [44], which has many applications in computer science. In addition, the results on circuits presented in this thesis have some further polyhedral implications that are also discussed, i.e., they easily give rise to lifted circuit inequalities for the polytope associated with OLS and MOLS. Last, our approach becomes evidently valuable for further work on the complete characterisation of both circuits and associated inequalities for MOLS and possibly for proving infeasibility of certain MOLS or other highly symmetric combinatorial problems.

This thesis is organised as follows. Chapter 1 outlines basic definitions and concepts needed for this work and provides a short dictionary of most commonly used acronyms, notation and terms. In Chapter 2, a formal definition for sets of MOLR is given in two ways which are in $1-1$ correspondence: as arrays and as sets. This serves to provide respectively a visual representation and a means for developing a set theoretical framework. In this chapter, generic definitions for independence systems associated with sets of MOLR2 and their related clutter of bases and circuits are established for the first time in the literature, setting the theoretical infrastructure for further work on this topic. We also define the availability matrix, a configuration conceived in [29] that enables structured and exhaustive proofs for characterising clutter of circuits for sets of MOLR. A list of the most recent applications of sets of MOLR in telecommunications wireless networks, the transmission of visual data and more are also discussed.

Chapter 3 starts by establishing a connection between a Latin square and a set of $n-1$ MOLR2, both of order n which is used to next prove that:
a) it is always possible to complete a 2-row Latin rectangle to a set of $n-1$ MOLR2 and
b) completing an incomplete set of $n-1$ MOLR2 is $\mathcal{NP}$-complete.

The rest of the chapter focuses on proving that any knowledge of the clutter of circuits for a set of t MOLR2 directly contributes to the characterisation of the clutter of circuits for a set of t' MOLR2 where $t < t' \leq n-1$. This links to the last section of Chapter 4 where it is shown that some of the derived circuits in fact contribute to the characterisation of the clutter of circuits related to the general MOLS problem. Finally, to get a better understanding of the size of this problem we count the members in the clutter of bases associated to 2-row Latin rectangles, OLR2, and MOLR2.

Characterising the clutter of circuits for OLR2 has been an important goal of this thesis. Though the nature of the problem required a highly enumerative approach to derive up to equivalence the complete list of circuits in this clutter, the results apply not just to a specific value of n but for any $n \geq 7$ (and for $n < 7$ a subset of circuits listed suffices). We start in Chapter 4 first by presenting the 5 non equivalent circuits derived in [29] for the case of 2-row Latin rectangles and then continue to describe and apply the methodology for deriving all circuits for OLR2. This involves the use of the availability matrix and an examination of all possible cells that can be emptied to derive minimal incomplete OLR2. In addition to R. Euler's circuits for Latin rectangles, we conclude that the OLR2 case consists of five non-equivalent classes of circuits containing in total twentyone non-equivalent circuits. This work is also extended in the first section of Chapter 6 where we present an initial list of seven new non-equivalent circuits with an interesting structure for the set of 3 MOLR2.

The MOLS problem is a special case of an assignment and this is the main focus of Chapter 5. The aim here is to derive a solution to the problem of finding or completing a set of MOLR, with use of optimisation techniques. Here we present a 3-index formulation for the OLR case which is attributed to R. Euler and continue to formulate all circuits derived in the previous chapter, as circuit inequalities. To strengthen the LP relaxation of the model we then maximally lift these circuit constraints that represent valid inequalities not only for the OLR case but also for the general MOLS problem. An extension of this formulation for the general MOLS case is presented in Chapter 6 for the fist time, followed by an extension of Gale's formulation (for the OLS case in [19]) to the general MOLS problem. The focus of this chapter is to present and analyse formulations for MOLR (MOLS) in general; we therefore also discuss new Constraint Programming (CP) formulations and introduce what are known as redundant models, for the case of MOLS, to improve constraint propagation and computational times.

Finally, Latin rectangles (squares) are known for their numerous applications in scheduling and timetabling. Chapter 7 starts by describing an application of Latin rectangles for a truck scheduling problem and then continues to describe a variation of this problem known as the multi-index bottleneck assignment problem (MBA). This is divided into three types, complete, mixed and arbitrary (detailed definitions are given in the chapter). Inspiration for addressing this problem was a statement by Burkard et al. in [13], that the complexity status of the complete-MBA problem is open. We prove that it is $\mathcal{NP}$-hard. After presenting IP and CP formulations for this problem we then describe three new heuristics, one for each type, and prove their approximation guarantee. This chapter consists of joint work with F. C. R. Spieksma and T. Dokka from the University of Leuven and the complexity results as well as the two approximation algorithms given for the arbitrary and mixed case are published in [26].

A concluding note both for the MOLR study and the MBA problem is included at the end of each chapter.

Contents

List of Tables

List of Figures

Chapter 1

Preliminaries

This chapter provides a preliminary note on the fundamentals used throughout this study, assuming that the reader is familiar with linear algebra. All concepts presented here are commonly used in the relevant literature and definitions borrowed are presented as in the most popular texts. More specifically, Section 1.1 introduces the basics on sets, Section 1.2 introduces independence systems, bases and circuits as described in [42]. Next, Section 1.3 describes the fundamentals of complexity theory classes as stated in [1] and [66] and Section 1.4 presents some basics on graph theory as described in [12], [24], and [37]. Sections 1.5 and 1.6 provide a brief introduction to Integer Programming and Constraint Programming as in [67] and [8], [39], [56], [65], [38]. Finally, Section 1.7 provides a dictionary for ready reference, of acronyms, terms and notation frequently used in this study.

1.1 Sets

A *set*, is a collection of elements where no two are identical. For example $E = \{e_1, e_2, e_3, e_4\}$ is a set, where e_1, e_2, e_3, e_4 are its *elements* or *members*. A *multiset* is a generalised concept of a set that relaxes this criterion. The order in which the elements of a set or a multiset appear is of no importance. If e_i is a member E this is denoted by $e_i \in E$ while if e_i is not a member of E it is denoted by $e_i \notin E$. A set with n elements is called an n-set and a set with no elements is denoted by $\emptyset$ and is known as an *empty set*. Finally, the number of elements in set E is denoted by $|E|$ and is known as the *cardinality* of E. For this study all sets are assumed to be finite.

We denote by $\mathbb{N}$ and $\mathbb{Z}$ the set of natural and integer numbers respectively, including zero and by $\mathbb{B} \subset \mathbb{Z}$ the set $\{0, 1\}$. The subscript '$+$' restricts the sets to numbers > 0, for example $\mathbb{Z}_+ = \{x \in \mathbb{Z} : x > 0\}$ is the set of all positive integer numbers. For a set of numbers E, we denote by $max(E)$ the *maximum* (i.e. greatest) element of E and by $min(E)$ the *minimum* (i.e. smallest) element of E. For e a real number, we denote by $\lceil e \rceil$ the least integer $\geq e$ and by $\lfloor e \rfloor$ the greatest integer $\leq e$.

Let E and F to be two sets, then $E \cup F$ is the *union* of E and F, $\{x : x \in E \text{ or } x \in F\}$; $E \cap F$ is the *intersection* of E and F, $\{x \;:\; x \in E \text{ and } x \in F\}$. Furthermore, $E \backslash F$ is the *difference* of E and F, $\{x \;:\; x \in E \text{ and } x \notin F\}$. Finally, if e_1 is an element of E then $E \backslash \{e_1\}$ is set E without element e_1.

Let F be a set that contains all elements of E, then this is denoted by $E \subseteq F$ and E is known as a *subset* of F and we often say that E is contained in F. Now if E is not a subset of F it is written as $E \nsubseteq F$. The notation $E \subset F$ indicates that E is a subset of F and $E \neq F$ with $E \neq \emptyset$. In this case we call E a *proper subset* of F. Two sets E and F for which $E \cap F = \emptyset$ are called *disjoint*.

A *partition* of a set E is a set of subsets S such that:

- all sets in S are pairwise disjoint
- the union of all sets of S forms the whole set E
- none of the sets in S are empty

Given a set E, the collection $\mathcal{I}$ of subsets of E is called a *family* of sets over E. A member of $\mathcal{I}$ is *maximal* if it is not a proper subset of any member of $\mathcal{I}$ and *minimal* if none of its proper subsets is a member of $\mathcal{I}$.

An ordered list of objects $G = (g_1, g_2, \dots, g_m)$ is called a *sequence* and just as for sets, $g_1, g_2, \dots, g_m$ are called its *elements* or *members*. However, unlike sets, the order in which the elements appear in a sequence is important and the same element can appear multiple times in different positions of the sequence. A k-*tuple* is a sequence of k elements. Let E_1, E_2, ..., E_k be k sets that do not necessarily contain the same number of elements. The set of all possible ordered k-tuples whose first element is a member of E_1, second element is a member of E_2 and so on, is called the *Cartesian product* of the k sets and is denoted by $E_1 \times E_2 \times \dots \times E_k = \{(e_1, e_2, \dots, e_k) \;:\; e_i \in E_i\}$.

1.2 Independence systems, bases and circuits

An *independence system* S is an ordered pair $(E, \mathcal{I})$ of a finite set E and a family $\mathcal{I}$ over E satisfying the following condition: Let I be a member of $\mathcal{I}$ and let J be a subset of I, then J also belongs to $\mathcal{I}$. Formally,

$$J \subset I \in \mathcal{I} \Rightarrow J \in \mathcal{I} \tag{1.2.1}$$

Given an independence system $S = (E, \mathcal{I})$, the set E is called the *ground set* of S and a member of $\mathcal{I}$ is known as an *independent* set of S. Furthermore, any subset of E not contained in $\mathcal{I}$ is called a *dependent* set of S.

Example 1.2.1

Let $E = \{e_1, e_2, e_3, e_4\}$ and $\mathcal{I} = \{\{e_1\}, \{e_2\}, \{e_3\}, \{e_1, e_2\}, \{e_2, e_3\}\}$. Thus, $\mathcal{I}$ contains all independent sets of the system $S = (E, \mathcal{I})$ and the dependent sets of S are:

$$\{\{e_4\}, \{e_1, e_3\}, \{e_1, e_4\}, \{e_2, e_4\}, \{e_3, e_4\}\} \cup \{X \subseteq E \;:\; |X| \geq 3\}$$

Definition 1.2.1. *Every maximal independent set $I \in \mathcal{I}$ is called a* basis*. Therefore I is such if,*

$$\nexists\, I' \in \mathcal{I} \;\; \textit{such that} \;\; I \subset I' \tag{1.2.2}$$

The collection $\mathcal{B}$ of all such sets is called the *clutter of bases*. It follows from the definition above that, every independence system $S = (E, \mathcal{I})$ induces a unique clutter of bases $\mathcal{B}$.

Definition 1.2.2. *Every minimal dependent set $C \in E$ is called a* circuit*. Therefore C is a circuit if it satisfies the properties,*

$$C \notin \mathcal{I} \tag{1.2.3}$$

and

$$\forall\; c \in C, \;\; C \backslash \{c\} \in \mathcal{I} \tag{1.2.4}$$

The collection $\mathcal{C}$ of all such sets is called the *clutter of circuits* and it follows from the definition above that,

Proposition 1.2.3. *Every independence system $S = (E, \mathcal{I})$ induces a unique clutter of circuits $\mathcal{C}$.*

Example 1.2.2
Continuing from Example 1.2.1, the clutter of bases is $\mathcal{B} = \{\{e_1, e_2\}, \{e_2, e_3\}\}$ and the clutter of circuits is $\mathcal{C} = \{\{e_4\}, \{e_1, e_3\}\}$. Notice that every independent set is contained in some member of $\mathcal{B}$ and every dependent set contains some member of $\mathcal{C}$.

Since $\mathcal{B}$ contains all maximal subsets of $\mathcal{I}$, then every subset of $\mathcal{I}$ that is not maximal is contained in some member of $\mathcal{B}$. Therefore, by having complete knowledge of $\mathcal{B}$ we can derive the independence system $(E, \mathcal{I})$ as follows:

$$\mathcal{I} = \{X \subseteq E \;:\; \exists\, B \in \mathcal{B} \;\; \text{such that} \;\; X \subseteq B\} \tag{1.2.5}$$

It is apparent from the statement above that,

Lemma 1.2.4. *Every clutter of bases $\mathcal{B}$ induces a unique clutter of circuits $\mathcal{C}$.*

therefore we can now rewrite the Definition 1.2.2 as follows:

Definition 1.2.5. *$C \in E$ is a circuit if*

$$\nexists\, B \in \mathcal{B} \;\; \textit{such that} \;\; C \subseteq B \tag{1.2.6}$$

and

$$\forall\, c \in C,\ \exists\, B \in \mathcal{B}\ \textit{such that}\ C\backslash\{c\} \subseteq B \tag{1.2.7}$$

Notice that (1.2.6) states that a circuit is excluded from any member of the clutter of bases; to simplify discussions in the following chapter we will refer to it as the *exclusion property*. We will also refer to (1.2.7) as the *minimality property* since every circuit C follows the definition of a minimal set i.e. if any element of C is removed, then the remaining set is a proper subset of some member of the clutter of bases.

Proposition 1.2.6. *A subset F of a set E is also a subset of some member of the clutter of bases $\mathcal{B}$ if and only if no member of the clutter of circuits $\mathcal{C}$ is a subset of F.*

1.3 Complexity

Let x (or y) denote an *input* (or *output*) of a given problem and let its *size* be $|x|$ (or $|y|$) equal to the number of bits in the encoding (or decoding) of x (or y) over an alphabet $\{0, 1\}$. Furthermore, we say that V is a *polynomial-time algorithm* if there exists a polynomial $p(n)$ such that the running time of V is $O(p(|x|))$. In this study we come across the following three types of computational problems,

Decision problems: We are required to verify whether an input satisfies a given property. Formally, given an input $x \in \{0, 1\}$ we are required to give a "Yes" or "No" answer.

Search problems: We are required not only to verify but also produce an answer. Formally, given an input $x \in \{0, 1\}$ we need to compute an answer $y \in \{0, 1\}$ which if it exists, is in some relation to x.

Optimisation problems: We are required to produce the best possible answer given some properties or prove that no such exists. Formally, given an input $x \in \{0, 1\}$ we are required to find $max_y\{|y| : (x, y) \in R\}$, where R is a polynomial time computational set such that for some polynomial p, $(x, y) \in R$ only if $|y| \leq p(|x|)$.

Problem B *reduces* to problem A, denoted by $B \leq_p A$, if there is a polynomial time computable function f such that $x \in B$ if $f(x) \in A$. Also, Problem B is *equivalent* to problem A, denoted by $A =_p B$, if $B \leq_p A$ and $A \leq_p B$.

Search and optimisation problems can be reduced to a corresponding decision version, we therefore define the two classes of problems: $\mathcal{P}$ (polynomial time) or $\mathcal{NP}$ (non-deterministic polynomial time), with respect to decision problems. Class $\mathcal{P}$ contains all decision problems that have polynomial-time algorithms; class $\mathcal{NP}$contains all decision problems such that for any "Yes" (or "No") instance of the problem, there is a short, polynomial proof, that the answer is "Yes" (or " No").

Clearly $\mathcal{P}$$\subseteq$$\mathcal{NP}$; however whether $\mathcal{P}$=$\mathcal{NP}$ is a huge open problem in complexity theory. If the statement were true, it would mean that any problem for which a solution can be easily verified can also be found easily.

Definition 1.3.1. *A decision problem A is $\mathcal{NP}$-hard, if for every problem $B \in \mathcal{NP}$, $B \leq_p A$.*

Definition 1.3.2. *A decision problem A is $\mathcal{NP}$-complete if it is $\mathcal{NP}$-hard and it belongs to $\mathcal{NP}$.*

Our approach for deriving complexity results in Sections 3.1 and 7.2, emerges from the fact that polynomial-time reductions are transitive. Analytically, for three problems A, B and C it is known that,

Theorem 1.3.3. *If $A \leq_p B$ and $B \leq_p C$ then also $A \leq_p C$.*

Corollary 1.3.4. *If problem A is in $\mathcal{NP}$, problem B is $\mathcal{NP}$-complete and $B \leq_p A$, then A is also $\mathcal{NP}$-complete.*

Therefore, in order to prove that a problem A is $\mathcal{NP}$-complete we must first prove that it is in $\mathcal{NP}$ and then reduce a known $\mathcal{NP}$-complete problem to A. Moreover, the most commonly used method to show that a problem A is $\mathcal{NP}$-hard, is to chose a known $\mathcal{NP}$-complete problem and prove that it reduces to A.

Many combinatorial optimisation problems are proven to be $\mathcal{NP}$-hard and it is unlikely that an algorithm can be found for any of them, that runs in polynomial time and finds an optimal solution (or shows that no such exists) for all instances. In order to tackle this issue, one can focus only on finding exact solutions for particular instances or alternatively, relax the optimality condition and construct algorithms that run in polynomial time and find close to optimal solutions. These are known as approximation algorithms. Formally,

Definition 1.3.5. *A p-approximation algorithm for a minimisation (maximisation) problem is an algorithm that finds a solution with an objective function value at most p times the optimum (at least $1/p$ for maximisation) for any instance of the problem.*

The ratio $p \geq 1$ is called the *performance ratio* of the algorithm. In some cases, it can be proven that for small p it is not even possible to have a p-approximation algorithm unless $\mathcal{P}= \mathcal{NP}$. These are known as *inapproximability* results.

1.4 Graph Theory

A *graph* $G(V, E)$ is defined by a set V of elements called *nodes* and a set E of elements called *edges* which are unordered 2-element subsets of V. The two nodes of an edge are joined and said to be *adjacent*. We will only be concerned with *finite* graphs, those in which sets V and E are both finite.

The *degree* of a node is the total number of adjacent edges.

A graph $G(V, E)$ is *bipartite* if V is the union of two disjoint sets V_1, V_2 such that each edge member of E consists of one node in A and the other in B. Such a graph can be written as $G(V_1, V_2, E)$.

Let $I = \{1, \ldots, m\} \subset \mathbb{N}$, with $i \in I$. A graph $G(V, E)$ is m*-partite* if $V = \bigcup_{i \in I} V_i$ and $V_1, ..., V_m$ are m disjoint sets such that each edge member of E consists of one node in V_i and the other in $V_{i'}$, with $i \neq i' \in I$.

A *matching* of graph $G(V, E)$ is a subset of edges which contains each member of V at most once. The *size* of a matching M is equal to the number of edges it contains.

A matching is
- *maximum* if it has the largest possible size
- *perfect* if it contains all members of V

Theorem 1.4.1 ([37])**.** *Given a bipartite graph $G(V_1, V_2, E)$ with node-sets V_1, V_2 and an edge-set E, G has a perfect matching if and only if for every subset S of V_1 the number of distinct nodes adjacent to some member of S is at least $|S|$.*

1.5 Integer Programming

In general, an Integer Linear Programming problem, simply refered to as Integer Programming and written as IP, is one that finds the minimum of a linear function, over a set of integer vectors that satisfy a collection of linear constraints. More specifically an IP is of the form,

$$Max\ \{c^T x \text{ subject to } Ax \leq b,\ x \in \mathbb{Z}^n\}$$

where A is an $m \times n$ matrix of real numbers, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and $x \in \mathbb{Z}^n$. The special case where x takes values in $\mathbb{B}^n$ is known as a 0-1 IP. Problems that require a “yes” or “no” answer are often formulated as a 0-1 IP and these are often graph theoretical or combinatorial problems that are considered difficult to solve.

A common approach for tackling 0-1 IPs, is to drop the integrality constraints, turning the problem into a Linear Programming problem (LP) for which efficient algorithms are known. In this case, x takes values in $\{0, ..., 1\}$. In general, the feasible region of the relaxation produced by dropping the integrality constraints is considerably larger than the convex hull (the smallest convex set that includes the original feasible set). However, there exist a number of algorithms by which valid inequalities are added to the problem in a systematic way, in order to reduce the feasible region and better approximate the convex hull. These valid inequalities can be “strengthened” with a procedure called *lifting* and is described below.

Let $N = \{1, ..., n\}$ and consider a set

$$X = \{x \in \mathbb{B}^n : \sum_{j}^{n} a_j x_j \leq b\}$$

With no loss of generality we assume $a_j \geq 0$ and $b > 0$.

Definition 1.5.1. *A set $C \subseteq N$ is a cover for X, if $\sum_{j \in C} a_j > b$.*

Definition 1.5.2. *A cover is minimal if $C \backslash \{j\}$ is not a cover for any $j \in C$.*

Thus minimal covers are covers that have no proper subset that is a cover. To conclude,

Proposition 1.5.3. *If $C \subseteq N$ is a cover for X, then the following inequality is valid for X*

$$\sum_{j \in C} x_j \leq |C| - 1$$

Proposition 1.5.4. *If C is a cover for X, the extended cover inequality*

$$\sum_{j \in E(C)} x_j \leq |C| - 1$$

is valid for X, where $E(C) = C \cup \{j : \ a_j \geq a_i$ for all $i \in C\}$

Extended cover inequalities are stronger inequalities because they dominate their corresponding cover inequalities. This is because more variables are added without changing the right hand side of the inequality.

Given a cover C for X the goal is to find the best possible values for the coefficient λ_j where $j \in N \backslash C$ such that the inequality

$$\sum_{j \in C} x_j + \sum_{j \in N \backslash C} \lambda_j x_j \leq |C| - 1$$

is valid for X.
Suppose that the set $L \subseteq N \backslash C$ contains the indices of the variables that have already been lifted. Thus,

$$\sum_{j \in C} x_j + \sum_{j \in L} \lambda_j x_j \leq |C| - 1$$

is valid for set $\{x \in \mathbb{B}^{|C|+|L|} : \ \sum_{j \in C \cup L} a_j x_j \leq b\}$. Suppose further that the next variable to be lifted is $k \in N \backslash C \cup L$. The aim is to find coefficient λ_k for variable x_k. For this we solve the following problem,

$$z_k = \max \sum_{j \in C} x_j + \sum_{j \in L} \lambda_j x_j + \lambda_k x_k$$

$$\sum_{j \in C \cup L} a_j x_j \leq b - a_k$$

$$x_j \in \{0,1\} \text{ for all } j \in C \cup L \cup \{k\}$$

Set $\lambda_k = |C| - 1 - z_k$ which gives the lifted inequality:

$$\sum_{j \in C} x_j + \sum_{j \in L} \lambda_j x_j + \lambda_k x_k \leq |C| - 1$$

We can now proceed to the next variable until all variables are lifted.

1.6 Constraint Programming

Constraint programming (CP) is a powerful tool for solving a wide range of combinatorial problems. It was developed in the $80s$ by the artificial intelligence community and has since found application in areas such as scheduling, vehicle routing, assembling, networks, bio-informatics and more (see [56]). Additionally, CP was adapted to solve IP problems by J. Hooker [39] and P. Williams [65].

More specifically, CP is an expressive modelling language with efficient solver implementations. A CP problem consists of a set of variables, each with a specific domain of values and a set of relations between subsets of these variables that can be expressed as constraints. Essentially, constraints restrict the values each variable can take, hence restrict the variables' domain. To make this more clear, take variables x_1 and x_2 both having domain $\{1, 2, 3\}$ and their relation is described by constraint $x_1 - x_2 \geq 0$; then the instances $(x_1, x_2) = (1, 2), (1, 3), (2, 3)$ are forbidden. A not-equal constraint $x_1 \neq x_2$ forbids instances $(x_1, x_2) = (1, 1), (2, 2), (3, 3)$.

Moreover, an objective function is not required in order to derive a solution and this explains the focus of CP literature on feasibility problems rather that optimality ones. The following definitions have been borrowed from [39].

Domain consistency, means that for each element v in the domain of any variable x_j, there is at least one feasible solution in which $x_j = v$.

Filtering, are algorithms that remove inconsistent values from domains.

Constraint propagation, is the process of obtaining smaller variable domains by filtering a constraint to become the starting point for filtering another constraint. It should be noted that constraint propagation does not necessarily achieve domain consistency for the problem as a whole, even if the filtering algorithms achieve it for every individual constraint.

Constructive search, explores the search tree by tightening the domain of one variable at a time. It is usually used when constraint propagation is unable to further reduce the domains and in practice, it performs a search in the space of partial solutions.

There are several constraint types in CP; we present the most commonly used in the literature and in this study, as introduced in [38]. The `alldifferent` constraint, written as `alldiff(X)`, is a combination of not-equal constraints that forbid variables of a set, to take pairwise equal values.

Definition 1.6.1. *Let x_i, $i \in N = \{1, ..., n\}$ be a variable with finite domain $D(x_i)$ then,*

$$\texttt{alldiff}(x_i \,:\, i \in N) = \{(d_i \,:\, i \in N) : d_i \in D(x_i),\ d_i \neq d_{i'} \textit{ for } i, i' \in N\} \tag{1.6.1}$$

Example 1.6.1
Given sets $N = \{1, 2, 3\}$ and $L = \{a, b, c\}$, we want to assign a letter from L to each cell of a 3×3 matrix such that no same letter appears in the same row. We introduce variable x_{ij} as the value in cell (i, j) with domain $D(x_{ij}) = L$ and $i, j \in N$. The constraints can be expressed as,

$$x_{ij} \neq x_{ij'}, \quad \text{for all } \ i,\ j,\ j' \in N, \ \text{with } j' > j \tag{1.6.2}$$

The constraint states that all pairs of cells in the same row must contain different values and one of the possible solutions to this problem is,

a	b	c
a	b	c
a	b	c

Constraint (1.6.1) can be transformed to an `alldiff` constraint as,

$$\texttt{alldiff}(x_{ij} \,:\, j \in N), \quad \text{for all } \ i \in N \tag{1.6.3}$$

1.7 Notation dictionary

This section is only presented as a dictionary for the acronyms notation and terms frequently used throughout this study. Further detail and definitions for specific terminology will be presented when needed in the chapters that follow.

Dictionary for Chapters 1-6

$T = \{1, ..., |T|\}$, with $t \in T$ and $|T| \leq n - 1$ is a set of mutually orthogonal latin rectangles of order n

- for $|T| = 1$ we have a Latin rectangle of order n
- for $|T| = 2$ we have an m-row orthogonal Latin rectangle of order n written as: OLRm; if $m = n$ then we have an orthogonal Latin square of order n written as: OLS

- for $|T| \geq 3$ we have a set of $|T|$ m-row orthogonal Latin rectangles of order n written as: a set of $|T|$ MOLRm; if $m = n$ then we have a set of $|T|$ orthogonal Latin squares of order n written as: a set of $|T|$ $MOLS$

R_t where $t \in T$, is a Latin rectangle
R_{t-} where $t \in T$, is an incomplete rectangle

For a rectangle R_t, $I = \{i_1, ..., i_n\}$ is the set of rows, $J = \{j_1, ..., j_n\}$, is the set of columns and $K^t = \{k_1^t, ..., k_n^t\}$, is the set of values, where $t \in T$

For the majority of this study we focus on the case of two rows where $I = \{i_1, i_2\}$ and we use the following notation:

$G^t = I \times J \times K^t$ is the superset containing all combinations of rows, columns and values for each R_t and its elements are written as (i, j, k^t), where $t \in T$
$G^{|T|} = \bigcup_{t \in T} G_t$ is the ground set

$S^{\mathcal{B}_{|T|}}$ is the independence system
$\mathcal{I}^{\mathcal{B}_{|T|}}$ is the family of independent sets
$\mathcal{B}_{|T|}$ is the clutter of basis
$\mathcal{C}_{|T|}$ is the clutter of circuits

$A(R_{t-}, i_1)$ is the availability matrix for row i_1 of R_{t-}
$A[R_{t-}, i_2]$ is the availability matrix for row i_2 of R_{t-}

Dictionary for Chapter 7

MBA is the m-dimensional bottleneck assignment problem
MBA3 is MBA for $m = 3$
OPT is the optimal solution to the MBA problem

$I = \{1, ..., m\}$ is the set of days, with $i \in I$
$J = \{1, ..., n\}$ is the set of shifts, with $j \in J$

V_i is the set of shifts that need to be carried out in day i
$V = \bigcup_{i \in I} V_i$ is the set of shifts across all days
$w(v)$ is a measure of cost for shift $v \in V$
D_j is the duty; a set of shifts to be carried out by a driver
cD_j is the cost of duty j

Chapter 2

The independence system of 2-row Latin rectangles, OLR2 and sets of $|T|$ MOLR2

This chapter is about $m \times n$ rectangles (with $m \leq n$) that follow the Latin rectangle and orthogonality definitions as described in [48]. The main focus is case of $m = 2$, i.e. 2-row Latin rectangles, 2-row orthogonal Latin rectangles (OLR2) and sets of $|T|$ mutually orthogonal 2-row Latin rectangles (MOLR2). These structures are defined in two ways: as arrays and as sets, which are in $1 - 1$ correspondence. The purpose of this dual interpretation is to provide respectively a visual representation of these structures and a means to develop a theoretical framework based on independence systems.

Section 2.1 sets the theoretical framework by presenting first basic definitions and then continuing to introduce the independence systems, clutter of bases and clutter of circuits for Latin rectangles, OLR2 and MOLR2. The collection of all Latin rectangles of an order n (OLR2 or MOLR2) constitute the Latin rectangle (OLR2 or MOLR2) clutter of bases. Circuits are minimal dependent sets that do not appear in any member of the clutter of bases. If at least one cell of a rectangle is empty, then it is called incomplete and if it contains no circuit it is completable otherwise incompletable. For purposes of establishing a methodology to fill empty cells in a systematic way, the available values for each cell are presented with a configuration called availability matrix.

Section 2.2 describes some basic problems this study aims to address, one of which is whether a set of $|T|$ incomplete rectangles can be completed to a set of $|T|$ MOLR2 and if not then identify the circuits that forbid completion. This can only be accomplished if the corresponding clutter of circuits is described fully. However, it is well known that there exists no polynomial time algorithm to describe the clutter of circuits for an arbitrary independence system, unless $P = NP$ [61]. Consequently, this study adds to the few independence systems in the literature, for which the clutter of circuits is fully characterised (see [61] and [47]).

Finally, it is worth mentioning that the general problem of Latin squares and their extensions define an entire field in mathematics with applications in finite geometry, statistical design and more (see [22], [23], [48]). Section 2.3 presents most recent applications for this work in telecommunications, wireless networks and the transmission of visual data as well as finding efficient ways to code and decode messages.

2.1 Latin rectangles, OLR2 and sets of $|T|$ MOLR2

2.1.1 Basic definitions

Definitions in this section are borrowed from [48].

Definition 2.1.1. *An m-row Latin rectangle R of order n, is an $m \times n$ array where $m < n$, in which each value $1, ..., n$ appears exactly once in every row and at most once in every column.*

For $m = n$ we have the case of *Latin squares*, where each value $1, ..., n$ appears exactly once in every row and column. An example of 2-row Latin rectangle of order 4 is shown in Table 2.1.1.

1	2	3	4
2	3	4	1

R_1

Table 2.1.1: 2-row Latin rectangle of order 4

Definition 2.1.2. *A Latin rectangle is said to be normalised, if values $1, ..., n$ occur in the first row in natural order.*

To avoid confusion with similar definitions for Latin rectangles, such as "reduced", "standardised", "in standard form" and other, we clarify that our definition of "normalised" refers to Latin rectangles for which values only of the first row are in natural order, in contrast to other definitions that also consider the first column to be in natural order.

Definition 2.1.3. *Two m-row Latin rectangles R_1 and R_2 of order n, form an orthogonal pair (OLR or OLRm) if and only if when superimposed each of the n^2 ordered pairs of values $(1, 1), (1, 2), ..., (n, n)$ appears at most once.*

An example of a 2-row orthogonal Latin rectangles (OLR2) of order 4 is shown in Table 2.1.2. Also note that for $m = n$ we have the case of *orthogonal Latin squares (OLS)* where each of the n^2 ordered pairs of values $(1, 1), (1, 2), ..., (n, n)$ appears exactly once, when the two squares are superimposed.

1	2	3	4
2	3	4	1

R_1

1	2	3	4
3	4	1	2

R_2

Table 2.1.2: OLR2 of order 4

The definition for OLR naturally extends to a set of $|T|$ m-row Latin rectangles of order n, known as *mutually orthogonal Latin rectangles (MOLR or MOLRm)*, if and only if all Latin rectangles are pairwise orthogonal. An example of a set of 3 2-row mutually orthogonal Latin rectangles (MOLR2) of order 4 is presented in Table 2.1.3. In this study, we are mainly interested in 2 rows and unless otherwise stated, whenever we refer to rectangles it is implied that they have two rows. It is also assumed from now on that all Latin rectangles are of order n.

1	2	3	4
2	3	4	1

R_1

1	2	3	4
3	4	1	2

R_2

1	2	3	4
4	1	2	3

R_3

Table 2.1.3: A set of 3 MOLR2 of order 4

Before continuing to introduce clutters of bases and circuits we first formalise notation. For a given order n, let $T = \{1, ..., |T|\}$, where $|T| \leq (n-1)$ is the number of MOLR2. For $|T| = 1$ we have a 2-row Latin rectangle and for $|T| = 2$ we have an OLR2 and so on. The two sets $I = \{i_1, i_2\}$, $J = \{j_1, ..., j_n\}$, correspond to the rows and columns of each Latin rectangle, while the $|T|$ disjoint sets $K^t = \{k_1^t, ..., k_n^t\}$ with $t \in T$ define the n values appearing in the t^{th} Latin rectangle comprising the MOLR2. Define $G^{|T|} = \bigcup_{t \in T} G^t$ to be the *ground set*, where $G^t = I \times J \times K^t$ and $t \in T$. Notice that each G^t contains $2\,n^2$ triples (i, j, k^t) thus $G^{|T|}$ contains a total of $2|T|\,n^2$ triples. Therefore, based on this notation, Tables 2.1.1, 2.1.2 and 2.1.3 are revised in Tables 2.1.4, 2.1.5 and 2.1.6.

	j_1	j_2	j_3	j_4
i_1	k_1^1	k_2^1	k_3^1	k_4^1
i_2	k_2^1	k_3^1	k_4^1	k_1^1

R_1

Table 2.1.4: The Latin rectangle of Table 2.1.1

	j_1	j_2	j_3	j_4
i_1	k_1^1	k_2^1	k_3^1	k_4^1
i_2	k_2^1	k_3^1	k_4^1	k_1^1

R_1

j_1	j_2	j_3	j_4
k_1^2	k_2^2	k_3^2	k_4^2
k_3^2	k_4^2	k_1^2	k_2^2

R_2

Table 2.1.5: The OLR2 of Table 2.1.2

	j_1	j_2	j_3	j_4
i_1	k_1^1	k_2^1	k_3^1	k_4^1
i_2	k_2^1	k_3^1	k_4^1	k_1^1

R_1

j_1	j_2	j_3	j_4
k_1^2	k_2^2	k_3^2	k_4^2
k_3^2	k_4^2	k_1^2	k_2^2

R_2

j_1	j_2	j_3	j_4
k_1^3	k_2^3	k_3^3	k_4^3
k_4^3	k_1^3	k_2^3	k_3^3

R_3

Table 2.1.6: The set of 3 MOLR2 of Table 2.1.3

We give some examples in order to establish a coherent connection between the representation of a rectangle both as an array and as a set. We can denote with i the rows, with j the columns and with k^t the value in cell

(i, j) of the array. Thus for $t \in T$, each Latin rectangle can be represented as a set R_1, which is a subset of G_t thus every triple (i, j, k^t) of R_1 represents a row, column and value of the corresponding array. For Latin rectangles, consider the illustrated array in Table 2.1.4, where $|T| = 1$; cell (i_1, j_1) containing value k_1^1 can be written as triple (i_1, j_1, k_1^1). Hence,

$$R_1 = \{(i_1, j_1, k_1^1), (i_1, j_2, k_2^1), (i_1, j_3, k_3^1, (i_1, j_4, k_4^1), (i_2, j_1, k_2^1), (i_2, j_2, k_3^1), (i_2, j_3, k_4^1), (i_2, j_4, k_1^1)\}$$

For OLR2 where $n = 4$ and $|T| = 2$, we represent the illustrated pair of arrays in Table 2.1.5 as a union of two sets, $R_1 \cup R_2$ where $R_1 \in G^1$, $R_2 \in G^2$. To clarify, sets I and J denote the rows and columns respectively of the two arrays comprising the pair; set K^1 denotes the values of the first rectangle and set K^2 the values of the second hence,

$$\begin{aligned} R_1 \cup R_2 =& \{(i_1, j_1, k_1^1), (i_1, j_2, k_2^1), (i_1, j_3, k_3^1), (i_1, j_4, k_4^1), (i_2, j_1, k_2^1), (i_2, j_2, k_3^1), (i_2, j_3, k_4^1, (i_2, j_4, k_1^1), \\ & (i_1, j_1, k_1^2), (i_1, j_2, k_2^2), (i_1, j_3, k_3^2), (i_1, j_4, k_4^2), (i_2, j_1, k_3^2), (i_2, j_2, k_4^2), (i_2, j_3, k_1^2), (i_2, j_4, k_2^2)\} \end{aligned}$$

Lastly, for MOLR2 where $n = 4$ and $|T| = 3$, we represent the illustrated set of arrays in Table 2.1.6 as a union of sets $R_1 \cup R_2 \cup R_3$ where $R_1 \in G^1$, $R_2 \in G^2$ and $R_3 \in G^3$, hence,

$$\begin{aligned} & R_1 \cup R_2 \cup R_3 = \\ & \{(i_1, j_1, k_1^1), (i_1, j_2, k_2^1), (i_1, j_3, k_3^1), (i_1, j_4, k_4^1), (i_2, j_1, k_2^1), (i_2, j_2, k_3^1), (i_2, j_3, k_4^1, (i_2, j_4, k_1^1), \\ & (i_1, j_1, k_1^2), (i_1, j_2, k_2^2), (i_1, j_3, k_3^2), (i_1, j_4, k_4^2), (i_2, j_1, k_3^2), (i_2, j_2, k_4^2), (i_2, j_3, k_1^2), (i_2, j_4, k_2^2), \\ & (i_1, j_1, k_1^3), (i_1, j_2, k_2^3), (i_1, j_3, k_3^3), (i_1, j_4, k_4^3), (i_2, j_1, k_4^3), (i_2, j_2, k_1^3), (i_2, j_3, k_2^3), (i_2, j_4, k_3^3)\} \end{aligned}$$

We call two MOLR2 (or two Latin rectangles) *equivalent*, if one can be obtained from the other by permuting elements of one or more of the sets T, I, J, K^1, ..., $K^{|T|}$. For example, OLR2 of Table 2.1.5 is equivalent to the one presented in Table 2.1.9. In the first table, we can perform the permutations $i_1 \leftrightarrow i_2$, $j_3 \leftrightarrow j_4$, $k_1^1 \leftrightarrow k_3^1$ on elements of sets I, J, K^1 in the sequence presented, to obtain the second table. These permutations are shown in Tables 2.1.7 to 2.1.9. Note that directly interchanging the roles of sets I and J is not allowed since we stipulate the existence of only 2 rows but n columns; the same applies to the interchange of I or J with a set G^t , $t \in T$. In contrast, interchanging two elements t_1, $t_2 \in T$ is an indirect role interchange of sets K^{t_1} and K^{t_2}.

	j_1	j_2	j_3	j_4	j_1	j_2	j_3	j_4
i_1	k_2^1	k_3^1	k_4^1	k_1^1	k_3^2	k_4^2	k_1^2	k_2^2
i_2	k_1^1	k_2^1	k_3^1	k_4^1	k_1^2	k_2^2	k_3^2	k_4^2
	R_1				R_2			

Table 2.1.7: Performing permutation $i_1 \leftrightarrow i_2$ on Table 2.1.5

	j_1	j_2	j_3	j_4		j_1	j_2	j_3	j_4
i_1	k_2^1	k_3^1	k_1^1	k_4^1		k_3^2	k_4^2	k_2^2	k_1^2
i_2	k_1^1	k_2^1	k_4^1	k_3^1		k_1^2	k_2^2	k_4^2	k_3^2
	R_1					R_2			

Table 2.1.8: Performing permutation $j_3 \leftrightarrow j_4$ on Table 2.1.7

	j_1	j_2	j_3	j_4		j_1	j_2	j_3	j_4
i_1	k_2^1	k_1^1	k_3^1	k_4^1		k_3^2	k_4^2	k_2^2	k_1^2
i_2	k_3^1	k_2^1	k_4^1	k_1^1		k_1^2	k_2^2	k_4^2	k_3^2
	R_1					R_2			

Table 2.1.9: Performing permutation $k_1^1 \leftrightarrow k_3^1$ on Table 2.1.8

2.1.2 The independence systems

For the case of two rows, we present the independence system of Latin rectangles and for the general case of sets of $|T|$ MOLR2. For these we also define, the clutter of bases and clutter of circuits.

Definition 2.1.4. *Every subset R of the ground set G^1 that forms a 2-row Latin rectangle is called a 2-row Latin rectangle basis. The collection of all such subsets, constitutes the clutter of 2-row Latin rectangle bases,*

$$\mathcal{B}_1 = \{R \subset G^1 \ : \ R \textit{ forms a 2-row Latin rectangle}\} \tag{2.1.1}$$

Notice that $|R| = 2n$. Table 2.1.4 illustrates a basis, i.e. a member of $\mathcal{B}_1$.

From $\mathcal{B}_1$ we can derive the 2*-row Latin rectangle independence system*, denoted by $S^{\mathcal{B}_1} = (G^1, \mathcal{I}^{\mathcal{B}_1})$, where $\mathcal{I}^{\mathcal{B}_1}$ contains all independent sets,

$$\mathcal{I}^{\mathcal{B}_1} = \{X \subset G^1 \ : \ \exists\, R \in \mathcal{B}_1 \text{ such that } X \subseteq R\} \tag{2.1.2}$$

It is known, from Proposition 1.2.3, that $S^{\mathcal{B}_1}$ induces a unique *Clutter of* 2*-row Latin rectangle circuits* denoted by $\mathcal{C}_1$ which contains all subsets of G^1 following the definition,

Definition 2.1.5. *Every minimal dependent set of $S^{\mathcal{B}_1}$ is called a 2-row Latin rectangle circuit, hence,*

$$\mathcal{C}_1 = \{C \subset G^1 \ : \ C \notin \mathcal{I}^{\mathcal{B}_1}, \ \ C \backslash \{c\} \in \mathcal{I}^{\mathcal{B}_1} \textit{ for all } \ c \in C\} \tag{2.1.3}$$

Similarly, for the MOLR2 case we have,

Definition 2.1.6. *Every subset B of the ground set $G^{|T|}$ that forms a set of $|T|$ MOLR2, is called an MOLR2 basis and the collection of all such subsets, forms the clutter of MOLR2 bases which is*

$$\mathcal{B}_{|T|} = \{B \subset G^{|T|} \ : \ B \textit{ forms a MOLR2}\} \tag{2.1.4}$$

From $\mathcal{B}_{|T|}$ we can derive the *MOLR*2 *independence system*, denoted by $S^{\mathcal{B}_{|T|}} = (G^{|T|}, \mathcal{I}^{\mathcal{B}_{|T|}})$ where,

$$\mathcal{I}^{\mathcal{B}_{|T|}} = \{X \subset G^{|T|} \ : \ \exists\, B \in G^{|T|} \text{ such that } X \subseteq B\} \tag{2.1.5}$$

Every minimal dependent set of $S^{\mathcal{B}_{|T|}}$ is called a circuit hence, $C \subset G^{|T|}$ is a circuit if it satisfies the properties,

$$\mathcal{C}_{|T|} = \{C \subset G^{|T|} \ : \ C \notin \mathcal{I}^{\mathcal{B}_{|T|}}, \ \ C\backslash\{c\} \in \mathcal{I}^{\mathcal{B}_{|T|}} \text{ for all } c \in C\} \tag{2.1.6}$$

The definition can be rewritten as follows:

Definition 2.1.7. *$\mathcal{C}_{|T|}$ is the collection of all $C \subset G^{|T|}$ that satisfy the following two properties:*

- *Exclusion property: C is not contained in any member of $\mathcal{B}_{|T|}$ thus,*

$$C \nsubseteq B, \ \textit{for all } \ B \in \mathcal{B}_{|T|} \tag{2.1.7}$$

- *Minimality property: removing any element of C will make the remainder a subset of some member of $\mathcal{B}_{|T|}$ thus,*

$$\textit{for all } \ c \in C, \ \exists\, B \in \mathcal{B}_{|T|} \ \textit{such that } \ C\backslash\{c\} \subseteq B \tag{2.1.8}$$

2.1.3 Incomplete rectangles

	j_1	j_2	j_3	j_4
i_1		k^1_1	k^1_3	k^1_4
i_2	k^1_2	k^1_4	k^1_1	k^1_3

R_{1-}

Table 2.1.10: An incomplete rectangle of order 4 that is incompletable

An *incomplete* rectangle, denoted by R_{1-} is an $m \times n$ array (with $m < n$) whose cells receive values 1 to n but may also be empty. We use the '$-$' minus sign in this notation to indicate that there are empty cells in the rectangle. Note, that incomplete rectangles can violate the definition of Latin rectangles in the sense that, a cell (i, j) may contain more than one values and same values may appear in a row and/or column. Such an example is shown in Table 4.1.2. An incomplete rectangle R_{1-} is *completable* if there exists $R'_{1-} \in G^1$ such that $R_{1-} \cup R'_{1-}$ forms a Latin rectangle, i.e. $R_{1-} \cup R'_{1-} \in \mathcal{B}_1$ and *incompletable* otherwise.

An example of an incomplete rectangle R_{1-} is presented in Table 2.1.10. Notice that cell (i_1, j_1) is empty and only value k^1_2 can be placed in the cell. An attempt to complete this structure will violate the definition of a Latin rectangle, as value k^1_2 will be repeated in column j_1; it is therefore incompletable. Conversely the rectangle of Table 2.1.11 is incomplete but completable; here $R_{1-} = \{(i_1, j_2, k^1_1), (i_1, j_3, k^1_3), (i_1, j_4, k^1_2), (i_2, j_1, k^1_1), (i_2, j_2, k^1_3), (i_2, j_4, k^1_4)\}$ and there exists $R'_{1-} = \{(i_1, j_1, k^1_4), (i_2, j_3, k^1_2)\} \in G^1$ such that $R_{1-} \cup R'_{1-} \in \mathcal{B}_1$.

	j_1	j_2	j_3	j_4
i_1		k_1^1	k_3^1	k_2^1
i_2	k_1^1	k_3^1		k_4^1

R_{1-}

Table 2.1.11: An incomplete rectangle of order 4 that is completable

In a similar fashion, a set of $|T|$ rectangles $\{R_{t-},\, t \in T\}$ is called *incomplete* if any such rectangle is incomplete and *completable* if and only if there are $\{R'_{t-},\, t \in T\}$ such that $\bigcup_{t\in T}(R_{t-} \cup R'_{t-}) \in \mathcal{B}_{|T|}$, i.e. if rectangles can be completed to a set of $|T|$ MOLR2 and *incompletable* otherwise.

Notice in Table 2.1.12 that R_{1-} can be completed to a Latin rectangle by placing k_4^1 and k_2^1 in cells (i_1, j_1) and (i_2, j_3) respectively. In fact, the resulting pair $R_1 \cup R_2$ is a member of $\mathcal{B}_2$ since it follows the Latin rectangle definition. Therefore, $R_{1-} \cup R_2$ as shown in Table 2.1.12 is completable to an OLR2.

	j_1	j_2	j_3	j_4
i_1		k_1^1	k_3^1	k_2^1
i_2	k_1^1	k_3^1		k_4^1

R_{1-}

j_1	j_2	j_3	j_4
k_1^2	k_2^2	k_3^2	k_4^2
k_3^2	k_4^2	k_1^2	k_2^2

R_{2-}

Table 2.1.12: A pair of incomplete rectangles of order 4

Equivalence also applies to (a set of) incomplete rectangles exactly as for (a set of) Latin rectangles, hence incomplete rectangles are called equivalent if one can be obtained form the other by permuting elements in T, I, J, K^1, ..., $K^{|T|}$.

Clearly $\{R_{t-},\, t \in T\}$ being completable implies that it is a subset of some $B \in \mathcal{B}_{|T|}$ thus containing no circuit. However, if incompletable and therefore not contained in any $B \in \mathcal{B}_{|T|}$, it contains some $C \in \mathcal{C}_{|T|}$. Hence the following,

Proposition 2.1.8. *An incomplete set $\{R_{t-},\, t \in T\}$ is completable to an MOLR2 if and only if it does not contain a circuit member of $\mathcal{C}_{|T|}$.*

Notice that any $C \in \mathcal{C}_{|T|}$ is itself an incomplete set of rectangles that is also incompletable; in that respect incomplete sets of rectangles that are incompletable are exactly the dependent subsets of $G^{|T|}$. We denote the rectangles comprising the circuits with R_{1-}, ..., $R_{|T|-}$.

2.1.4 The availability matrix $A(R_{1-}, i)$

	j_1	j_2	j_3	j_4
i_1		k_2^1		k_4^1
i_2		k_4^1	k_1^1	k_2^1

R_{1-}

Table 2.1.13: An incomplete rectangle of order 4

Consider the incomplete rectangle R_{1-} of Table 2.1.13. To fill the empty cell (i_1, j_1) of the array, a value must be chosen that does not violate the definition of a Latin rectangle. Notice that the *available* triples are k_1^1, k_3^1 and for this cell we can select any of the two. In general, the collection of all available values for a particular row i constitute set,

$$A_i = \{(i, j, k^1) \in G^1 \;:\; a \cup R_{1-} \text{ does not violate the Latin rectangle definition}\} \tag{2.1.9}$$

For the purpose of establishing a simple and comprehensive methodology for filling empty cells, we visually represent A_i with a configuration first described in [29] known as the *availability matrix* of row i.

Definition 2.1.9. *Let R_{1-} denote an incomplete rectangle with n columns, $S(i)$ denote the set of symbols appearing in row i and $J(i)$ the set of column indices of the v empty cells in that row, where $v < n$. The availability matrix $A(R_{1-}, i)$ is the $v \times v$ matrix obtained from the $n \times n$ matrix*

$$\begin{array}{c} \begin{array}{ccc} \mathrm{j}_1 & \cdots & \mathrm{j}_\mathrm{n} \end{array} \\ \begin{pmatrix} \mathrm{k}_1^1 & \cdots & \mathrm{k}_1^1 \\ \mathrm{k}_2^1 & \cdots & \mathrm{k}_2^1 \\ \cdots & \cdots & \cdots \\ \mathrm{k}_\mathrm{n}^1 & \cdots & \mathrm{k}_\mathrm{n}^1 \end{pmatrix} \end{array}$$

after deleting from the matrix all rows corresponding to elements of $S(i)$ and all columns that are not members of $J(i)$. We mark an element of $A(R_{1-}, i)$ in column j with the symbol '$*$' to indicate that the value is not available, if and only if that element appears in column j of R_{1-}.

The *dimension* of an availability matrix is simply the number of its rows *or* columns. Notice that for any availability matrix the number of rows is equal to the number of columns.

We employ an availability matrix for a particular incomplete rectangle only if non-empty cells in that rectangle do not violate the Latin rectangle definition. In particular, if in an incomplete rectangle there appears a repetition of values in the same row or column, or a cell contains more than one entry, then clearly this structure is incompletable. In such cases, the violation of the definition is explicit and therefore using the availability matrix to demonstrate incompletability is unnecessary.

Also, we use the curved brackets () to denote the availability matrix of the first row, $A(R_{1-}, i_1)$ and the square brackets [] to denote the availability matrix of the second row, $A[R_{1-}, i_2]$. Notice that there is a one-to-one correspondence between A_i and $A(R_{1-}, i)$, therefore in terms of set notation, for a particular row i, every combination of column j and value k^1 of the availability matrix represents a member (i, j, k^1) of A_i. Therefore together $A(R_{1-}, i_1)$ and $A[R_{1-}, i_2]$ denoted simply by A can be expressed as a subset of G^1 . This is demonstrated in the example below.

Example 2.1.1
For the first row of Table 2.1.13, $S(i_1) = \{k_2^1, k_4^1\}$ and $J(i_1) = \{j_1, j_3\}$ thus,

$$A(R_{1-}, i_1) = \begin{array}{c} \begin{array}{cc} \mathrm{j}_1 & \mathrm{j}_3 \end{array} \\ \begin{pmatrix} \mathrm{k}_1^1 & \mathrm{k}_1^{1*} \\ \mathrm{k}_3^1 & \mathrm{k}_3^1 \end{pmatrix} \end{array}$$

and shows that values k_1^1 and k_3^1 are available in columns j_1 and j_3.

For the second row of R_{1-} sets $S(i_2) = \{k_1^1, k_2^1, k_4^1\}$ and $J(i_2) = \{j_1\}$, therefore since there is only one value available for the empty cell,

$$A[R_{1-}, i_2] = \begin{array}{c} \mathrm{j}_1 \\ \begin{bmatrix} \mathrm{k}_3^1 \end{bmatrix} \end{array}$$

Finally notice that $A = \{(i_1, j_1, k_1^1), (i_1, j_1, k_3^1), (i_1, j_3, k_3^1), (i_2, j_1, k_3^1)\}$. Element (i_1, j_3, k_1^1) is not included in the set as it has an '$*$' indicating it is not available.

Remark 2.1.10. *If there exists $R'_{1-} \in A$ such that $R_{1-} \cup R'_{1-}$ forms a Latin rectangle, then R_{1-} is completable, otherwise incompletable.*

It is easy to complete the incomplete rectangle R_{1-} of Table 2.1.13, as shown in Table 2.1.14. Notice from $A[R_{1-}, i_2]$, that k_3^1 is the only available value for cell (i_2, j_1) and consequently k_1^1 becomes the only available value for cell (i_1, j_1). In general, to complete any R_{1-}, a single value must be selected from every row and column in $A(R_{1-}, i_1)$ and $A[R_{1-}, i_2]$ such that the definition of a Latin rectangle is not violated. More specifically, the value selected in column j of $A(R_{1-}, i_1)$ must be different from the one selected in column j of $A[R_{1-}, i_2]$.

	j_1	j_2	j_3	j_4
i_1	k_1^1	k_2^1	k_3^1	k_4^1
i_2	k_3^1	k_4^1	k_1^1	k_2^1

R_1

Table 2.1.14: Completed rectangles of Table 2.1.13

To highlight the selected entry of $A(R_{1-}, i_1)$ we use a **circle** and similarly for the selected entry of $A[R_{1-}, i_2]$ we use a **square**. For illustration reasons, we merge the two availability matrices into one figure denoted by A

$$\begin{array}{c} \begin{array}{cc} \mathrm{j}_1 & \mathrm{j}_3 \end{array} \\ \begin{pmatrix} \enclose{circle}{\mathrm{k}_1^{1*}} & \mathrm{k}_1^{1*} \\ [\,\boxed{\mathrm{k}_3^1}\,] & \enclose{circle}{\mathrm{k}_3^1} \end{pmatrix} \\ \mathrm{A} \end{array}$$

In summary, an incomplete rectangle R_{1-} is completable if,

1) A single entry can be selected from each row and column of $A(R_{1-}, i_1)$ and $A[R_{1-}, i_2]$ (entries with an '$*$' cannot be selected)

2) For same columns of $A(R_{1-}, i_1)$ and $A[R_{1-}, i_2]$ the selected entry is different

2.2 Motivation

It is shown in [47] that,

Theorem 2.2.1. *Unless $\mathcal{P} = \mathcal{NP}$, there exists no polynomial total time algorithm that generates the clutter of bases of any independence system.*

In [47] one can find a list of particular independence systems for which it is possible to characterise their clutter of bases. This list includes knapsack problems, set packing problems, complete k -partite sub-graphs and other.

It is also proven in [61] that,

Corollary 2.2.2. *Unless $\mathcal{P} = \mathcal{NP}$, there exists no polynomial total time algorithm that generates the clutter of circuits of any independence system.*

The clutter of circuits $\mathcal{C}_1$ has been fully described by R. Euler in [29]. Our goal in this study is to obtain a characterisation of the complete list of circuit members of $\mathcal{C}_2$. The difficulty of this task is evident from Theorem 2.2.1 and there exist few examples in the literature, some of which are in [61], [47], that characterise the clutter of circuits for a particular independence system. Our work therefore adds to the few independence systems in the literature for which the clutter of circuits is fully characterised. The constructive proof we present in Chapter 4 to list all circuits as well as the methodology presented in this study sets the grounds for further research in this direction. Some potential research problems that have also motivated this work are presented below as well as some interesting applications of Latin rectangles, OLR and MOLR in the next section.

Problem 1: We are interested in determining whether an incomplete pair of rectangles R_{1-}, R_{2-} is completable to an orthogonal pair. Set $R_{1\_} \cup R_{2\_}$ can potentially form a basis member of $\mathcal{B}_2$ if and only if it does not contain a member of $\mathcal{C}_2$. Therefore, complete knowledge of $\mathcal{C}_2$ as well as the ability to recognise all circuits of $\mathcal{C}_2$ in any pair of incomplete rectangles, will allow us to determine whether they are completable. In other words, for $|T| = 2$ we are interested in knowing if there exist $R'_{1\_} \cup R'_{2\_} \in G^2$ such that $\bigcup_{t \in T}(R_{t-} \cup R^{'}_{t-}) \in \mathcal{B}_2$ and if not, then recognise the circuit(s) that cause incompletability.

Completing pairs of rectangles to form OLR2 is part of a much larger problem, that of completing pairs of incomplete squares to form OLS. It is obvious that if an incomplete pair of rectangles is incompletable then so is any pair of squares that contains it. The opposite however does not hold. For example, even though there exists a pair of OLR2 of order 6, a pair of OLS of order 6 does not exist. This proves, that Theorem 3.1.2 by Hall [36] is not generalisable for the case of orthogonal Latin rectangles, and to our knowledge no equivalent statement exists, that determines the completability of pairs of squares based on the completability of pair of rectangles. Nevertheless, establishing a methodology that helps generate circuits for pairs of OLR2, may shed light on the much larger problem; that of finding the clutter of circuits for pairs of OLS of any order n or it may help to prove that this is not possible in polynomial time. This is of particular interest, since Proposition 2.1.8 generalises to the MOLS case and finding the clutter of circuits for this problem would answer any completability question. However, one must consider that the circuits found for OLR2 are not necessarily circuits of the OLS problem, as the minimality property may not preserve.

Another possible direction would be to address the problem of MOLR, to investigate why certain incomplete structures are incompletable. It will also be shown further on that circuits of OLR2 are also circuits for MOLR2, therefore knowing all members of $\mathcal{C}_2$ adds to the knowledge of $\mathcal{C}_{|T|}$.

Problem 2: For pairs of Latin rectangles that are not orthogonal we are interested in finding the minimum number of value swaps in order for the pair to become orthogonal. In detail, let us define a swap, as the act of selecting two cells in the same row and swapping their values. Let us consider the case for which two superimposed Latin rectangles R_1, R_2 violate the definition of orthogonality i.e. a pair of values is repeated. We are interested to determine the minimum number of swaps that need to be made in order for the pair to become orthogonal. Remember that R_1, R_2 will become orthogonal when all circuits are eliminated. With the complete knowledge of $\mathcal{C}_2$ and the ability to recognise all members of $\mathcal{C}_2$ that exist in $R_1 \cup R_2$ we can then make a minimum number of swaps such that all circuits are eliminated and no new ones are created, thus leading to the creation of an orthogonal pair. Furthermore, this problem can be rephrased in various interesting ways for example: “If R_1 and R_2 are not orthogonal, find a basis $B \in \mathcal{B}_2$ that has the maximum number of common elements with $R_1 \cup R_2$”, or “If R_1 and R_2 are not orthogonal, find the maximum subset of $R_1 \cup R_2$ that can be found in a basis $B \in \mathcal{B}_2$ ”.

2.3 Applications

Latin squares and orthogonal Latin squares are known to have a wide range of applications. In this section we focus on some of the more recent ones that mainly refer to Latin rectangles and OLR.

Incomplete rectangles for minimising interference in transmission of data: With the advent of fast communication via mobiles and other devices, designing fast and reliable wireless networks has become very important. Traditional network nodes (e.g. mobile phones and mobile services suppliers' towers) could only forward or replicate an incoming packet (of data). In contrast, network coding can create and handle additional packets by algebraically combining received packets. This realisation has led to a method known as Code Division Multiple Access (CDMA) for transmitting multiple coded digital signals simultaneously over the same frequency in a network. For mobile networks each conversation is assigned a unique code before entering a packet which is allowed to be transmitted over the full bandwidth and decoded at the end node. As a result, efficient methods of coding which maximise the size and number of packets handled and minimise interference have become a hot topic of research. In 2006 [54] and [69] introduced the concept of physical layer network coding which has developed in to a sub-field of network coding with new results in the domains of wireless communication, wireless information theory and wireless networking. One branch of this new field works with de-noise-and-forward-protocol in the network coding maps that satisfy a requirement called the exclusive law which reduces the impact of multiple access interference. In [64] it is established that the network coding maps that satisfy the exclusive law are obtainable by the completion of partially filled Latin Rectangles. Isotopic and transposed Latin Squares are also used to create network coding maps with particular desirable characteristics.

MOLR and MOLS for transmitting optical data: A big part of data transmission in the modern era has to do with visual data (pictures, photos, videos etc). This has led to fibre-optic signal processing techniques for Optical Code Division Multiple Access (OCDMA), first studies in [58], which deliver multi-access optical networks for fibre-optic communications. One important type of OCDMA system is known as an Optical Orthogonal Code (OOC). An OOC is a family of $(0, 1)$ sequences with good auto and cross-correlation properties, i.e., fast and low interference transmission properties. One major branch of OCDs is Spectral Amplitude Coding (SAC). In [25] the authors propose two new coding schemes capable of cancelling the multi-user interference for certain SAC systems. Three major advantages of the proposed OOC families are claimed to be: $1)$ large flexibility in choosing number of users, $2)$ simplicity of construction and $3)$ suitability to all important transmission technologies including SAC. One of these novel schemes is based on MOLR and MOLS.

MOLR and MOLS for Low Density Parity Codes (LDPC): In fast communication of vast data sets, a big problem is noise or corruption of data. LDPC codes were invented to deal with this problem by changing the process of coding and decoding messages; they have revolutionised the accuracy of data transmission. LDPC codes are now the lead technology used in hard disk drive read channels, wireless 10-GB, DVB-S2 digital TV receivers, computers with third generation wireless and more recently in flash SSD as well as in communicating with space probes sent out by NASA. Pseudo-random approaches and combinatorial approaches are the two main techniques for the construction of a specific LDPC code based on finite geometries was fist studied in [45]. In [63] and [46] a different construction is devised, based on balanced incomplete block designs constructed from MOLR and MOLS.

2.4 Concluding remarks

This chapter established a set theoretical framework for Latin rectangles and OLR2. The independence systems, clutters of bases and circuits were defined. Next the concept of completable and incompletable rectangles and pairs of rectangles was discussed and how this can be illustrated with the availability matrix. The chapter concluded with a number of motivational reasons for this study such as answering the completability question for any given incomplete pair of rectangles and finding the minimum number of cells that need to be emptied in order to allow for non-orthogonal pair of Latin rectangles to become orthogonal. Some of the most recent applications of Latin rectangles and MOLRm were also discussed, for example in telecommunications and wireless networks.

It would be interesting to extend this work to establish a similar theoretical framework for the OLRm case and even for the more general case MOLRm where $m > 2$.

The next chapter continues to present theoretical and complexity results.

Chapter 3

Theoretical results

This Chapter presents some initial theoretical findings that are essential for the proofs carried out in Chapter 4 that deals with the characterisation of the clutter of circuits $\mathcal{C}_2$.

Section 3.1 starts by essentially establishing a connection between a Latin square and a set of $n-1$ MOLR over two rows, both of order n. In fact, it proves that a Latin square can be written as a set of $n-1$ MOLR2 and vice versa. This realisation sets the grounds for proving a simple but fundamental finding of this work, that any set $|T|$ of MOLR2 (where $|T| < n-1$) can be completed to a set of $n-1$ MOLR2.

Based on these findings Section 3.2 utilises Coulbroun's theorem in [18] for Latin squares, to establish that completing a set of $|T|$ MOLR2 is $\mathcal{NP}$-complete. The section concludes that the clutter of circuits $\mathcal{C}_t$ is a subset of $\mathcal{C}_{t+1}$, where $t \in T$ and $t < |T|$; and this reveals how by characterising a clutter of circuits for a set of MOLR2 directly improves knowledge for all greater number of sets of MOLR2.

Finally, in order to establish an understanding of size with respect to Latin rectangles, OLR2 and sets of 3 MOLR2, Section 3.3 establishes a methodology to count the members of their corresponding clutters of bases and provides tables with exact calculations for various orders of n.

3.1 From Latin squares to sets of $n-1$ MOLR2 and vice versa

Proposition 3.1.1. *Any set of $|T|$ normalised MOLR2 of order n can be represented as a normalised $(|T|+1)$-row Latin rectangle of order n and vice versa.*

Before proving the proposition we illustrate for $n=4$ and $|T|=2$ in the example below:

Example 3.1.1
Let us consider the case of $|T| = 2$. Table 3.1.1 illustrates a normalised OLR2 of order $n = 4$. In the second row of R_2, n distinct values appear and each in a different column than in the row above. Since the normalised first row in both R_1 and R_2 establishes the occurrence of the n pairs of values

$(k_1^1, k_1^2), (k_2^1, k_2^2), (k_3^1, k_3^2), (k_4^1, k_4^2)$, values of the second row of R_2 must each be in different columns than in the second row of R_1; this is to avoid repetition of pairs of values. Since the first row of R_1 and the first row of R_2 are the same, the second row of R_2 can be placed as a third row in R_1 and we will now obtain a 3-row Latin rectangle as shown in Table 3.1.2. For $m = n$ we can say that every normalised Latin square can be represented as a set of $|T| = (n-1)$ MOLR2 of order n.

	j_1	j_2	j_3	j_4
i_1	k_1^1	k_2^1	k_3^1	k_4^1
i_2	k_2^1	k_3^1	k_4^1	k_1^1

R_1

j_1	j_2	j_3	j_4
k_1^2	k_2^2	k_3^2	k_4^2
k_3^2	k_4^2	k_1^2	k_2^2

R_2

Table 3.1.1: An OLR2 of order 4

	j_1	j_2	j_3	j_4
i_1	k_1	k_2	k_3	k_4
i_2	k_2	k_3	k_4	k_1
i_3	k_3	k_4	k_1	k_2

R

Table 3.1.2: A 3-row Latin rectangle of order 4

Proof. The $(|T|+1)$-row latin rectangle of order n, denoted as R, is obtained from the set of $|T|$ normalised MOLR2 by placing the entries of the second row of R_t $(t \in T)$ at the $(t+1)^{th}$ row of R; that is, if the value of cell (i_2, j) of R_t is k_l^t then cell $(t+1, j)$ recieves value k_l^t (see Example 3.1.1). It follows that every value appears once in each row of R, since it appears once in every row of R_t $(t \in T)$. It remains to show that every value occurs at most once in each column of R.

Since all $|T|$ rectangles in MOLR2 are normalised, for any two of them, say R_t and $R_{t'}$, the n pairs of values $(k_1^t, k_1^{t'})$, ..., $(k_n^t, k_n^{t'})$ appear in the first row. Then, to avoid repeating a pair, value k_l^1 $(l = 1, ..., n)$ occurring in the second row of R_t $(t \in T)$ is bound to be at a different column of $R_{t'}$ $(t' \in T)$. But then, value k_l^1 $(l = 1, ..., n)$ appears at most once per column of R.

It becomes easy to see that the construction is applicable in the inverse direction, i.e. given a normalised $(|T|+1)$-row Latin rectangle of order n, one can obtain a set of $|T|$ normalised MOLR2 of the same order. □

Hall in [36] presents the following,

Theorem 3.1.2. *[36] Every m-row Latin rectangle of order n can be completed to a Latin square of order n.*

We adapt this theorem to MOLR2 as follows,

Corollary 3.1.3. *Any set of $|T|$ MOLR2 $(1 \leq |T| \leq n-1)$ of order n can be completed to a set of $n-1$ MOLR2 of order n.*

Clearly for $|T| = 1$ we have the case of a 2-row Latin rectangle that can be completed to a set of $n-1$ MOLR2, since according to Theorem 3.1.2, any 2-row Latin rectangle can be completed to a Latin square and from Proposition 3.1.1 we know that this Latin square can be represented as set of $n-1$ MOLR2. For $|T| = 2$ we have the case of an OLR2 that can be completed to a set of $n-1$ MOLR2. Similarly it follows for $|T| > 2$.

Moreover, a Latin rectangle of order n can have at most n rows, in which case it would be a Latin square. Hence, Proposition 3.1.1 implies that there exists a set of $n-1$ MOLR2 of order n; the latter directly yields that there can be at most a set of $n-1$ MOLS; an alternative proof for Theorem 2.1 in [48].

3.2 Complexity results

We continue with an examination of the complexity class of the problem of completing a given incomplete set of $|T|$ 2-row rectangles of order n to a set of $n-1$ MOLR2. It is trivial to construct a set of $|T|$ MOLR2 of order n with no prior restrictions. One can easily place values in natural order in the first row of all $|T|$ rectangles and then in a cyclic manner shift elements by one value to the right in the second row of every rectangle. However, for the case where some cells already have values, deciding whether $|T|$ incomplete 2-row rectangles are completable to a set of $|T|$ MOLR2, is not that straight forward. We are faced with the following decision problem, which we will prove to be $\mathcal{NP}$-complete.

Input: $|T|$ incomplete 2-row rectangles of order n

Question: Are they completable to a set of $|T|$ MOLR2?

Clearly this problem is in $\mathcal{NP}$, since given a solution we can easily verify its correctness by simply listing all pairs of values obtained from the pairwise superimposed rectangles, and checking whether there appears a repetition of a pair. To complete the proof, we will show that our problem is equivalent to the problem of completing an incomplete square of order n to a Latin square; this has been proven to be $\mathcal{NP}$-complete by Colbourn in [18].

Notice, that while an arbitrary incomplete square can be represented as a set of $n-1$ incomplete 2-row rectangles, the opposite is not always true. Such an example for $n = 4$, of a set of $n-1$ incomplete 2-row rectangles that cannot be written as an incomplete square due to the position of empty/filled cells, is shown in Table 3.2.1.

	j_1	j_2	j_3	j_4
i_1	k_1^1			
i_2		k_2^1		

R_1

	j_1	j_2	j_3	j_4
i_1	k_1^2	k_2^2		
i_2			k_3^2	

R_2

	j_1	j_2	j_3	j_4
i_1	k_1^3	k_2^3	k_3^3	
i_2				k_4^3

R_3

Table 3.2.1: A set of $n-1$ incomplete 2-row rectangles

Let us consider the case of $|T| = n - 1$. We construct $n-1$ incomplete rectangles, where all filled cells in the first row of all arrays contain the same values and there is no requirement for the second row. Such an example is shown in Table 3.2.2. The incomplete rectangles can be completed if and only if the corresponding Latin square, as shown in Table 3.2.3, is completable.

If the incomplete rectangles $R_{1-}, ..., R_{n-1-}$ are completable, elements in sets $K^1, ..., K^{n-1}$ can be permuted so that the MOLR is normalised. It then follows from Proposition 3.1.1 that this represents a Latin square. Conversely, if the incomplete square is completed then it follows again from Proposition 3.1.1 that we also have a set of MOLR2. We have shown that,

Corollary 3.2.1. *Deciding whether $n-1$ incomplete 2-row rectangles of order n are completable to a set of $n-1$ MOLR2, is $\mathcal{NP}$-complete.*

	j_1	j_2	j_3	j_4
i_1			k_3^1	k_4^1
i_2			k_4^1	k_1^1

R_1

	j_1	j_2	j_3	j_4
i_1			k_3^2	k_4^2
i_2	k_3^2			k_2^2

R_2

	j_1	j_2	j_3	j_4
i_1			k_3^3	k_4^3
i_2	k_4^3	k_1^3		

R_3

Table 3.2.2: A set of 3 incomplete 2-row rectangles of order 4

	j_1	j_2	j_3	j_4
i_1			k_3	k_4
i_2			k_4	k_1
i_3	k_3			k_2
i_4	k_4	k_1		

R

Table 3.2.3: An incomplete square of order 4

Now consider the following problem: complete a set of $|T|$ MOLR2 to a set of $|T|+1$ MOLR2, where $|T| \leq n-2$. Finding that additional $(|T|+1)^{th}$ Latin rectangle can be translated to the problem of adding a $(|T|+2)^{th}$ row to a $(|T|+1)$-row Latin rectangle. For $|T| = 1$ we present a short algorithm that solves this problem and clearly generalises for $|T| \leq n-2$.

For $|T| = 1$ we have the problem of adding a third row to a 2-row Latin rectangle in order to make it a 3-row Latin rectangle. We first normalise the Latin rectangle, then add an empty row and lastly attempt to complete it. Let $A(R_{1-}, i_3)$ be the $n \times n$ availability matrix of row i_3 of the incomplete 3-row rectangle R_{1-}. Since cells of the first two rows of R_{1-} contain a value, then in each column of $A(R_{1-}, i_3)$, $n-2$ values will be available (i.e. will not have a '*' symbol) and each value from k_1^1 to k_n^1 will be available in exactly $n-2$ columns. The following steps demonstrate that it is possible to select a value in each column.

Step 1**:** Let $k_1^1 ... \, k_{n-2}^1$ be the $n-2$ available values in column j_1. Select k_1^1 in column j_1.

Step 2**a:** If there exists a column identical to column j_1 then select k_2^1 in that column and the remaining $k_3^1 ... \, k_{n-2}^1$ values in any of the $n-4$ remaining columns in which they are available, otherwise

Step 2**b:** Select the remaining $k_2^1 ... \, k_{n-2}^1$ values in any of the $n-3$ remaining columns in which they are available.

Step 3**:** Select k_{n-1}^1 and k_n^1 in the remaining two columns. With no loss of generality we can assume these to be columns j_{n-1} and j_n.

To prove the correctness of the algorithm, it needs to be shown that in Step 3 a selection of k_{n-1}^1 and k_n^1 is always possible. Values k_{n-1}^1 and k_n^1 will each be available at least once in j_{n-1} or j_n since none of them are available in column j_1 and also they must be available a total of $n-2$ times in $A(R_{1-}, i_3)$. Moreover, none of them can appear in one of the last columns j_{n-1} or j_n, because in that case one of the last two columns would be identical to j_1 and according to Step $2a$ it would be considered earlier. For an example consider Table 3.2.4 in which a 3^{rd} empty row is added and is completed in Table 3.2.5 which shows the corresponding availability matrix (selected values are circled). The process presented in these two tables can be used to find R_2 of Table 2.1.5, given R_1.

	j_1	j_2	j_3	j_4
i_1	k_1^1	k_2^1	k_3^1	k_4^1
i_2	k_2^1	k_3^1	k_4^1	k_1^1
i_3				

R_1

Table 3.2.4: An incomplete 3-row rectangle

$$A(R_{1-}, i_3) = \begin{array}{c} \begin{array}{cccc} \mathrm{j}_1 & \mathrm{j}_2 & \mathrm{j}_3 & \mathrm{j}_4 \end{array} \\ \left(\begin{array}{cccc} \mathrm{k}_1^{1*} & \mathrm{k}_1^1 & \enclose{circle}{\mathrm{k}_1^1} & \mathrm{k}_1^{1*} \\ \mathrm{k}_2^{1*} & \mathrm{k}_2^{1*} & \mathrm{k}_2^1 & \enclose{circle}{\mathrm{k}_2^1} \\ \enclose{circle}{\mathrm{k}_3^1} & \mathrm{k}_3^{1*} & \mathrm{k}_3^{1*} & \mathrm{k}_3^1 \\ \mathrm{k}_4^1 & \enclose{circle}{\mathrm{k}_4^1} & \mathrm{k}_4^{1*} & \mathrm{k}_4^{1*} \end{array} \right) \end{array}$$

Table 3.2.5: Availability matrix for row i_3 of R_{1-} in Table 3.2.4

Since every set of $|T|$ MOLR2 with $|T| \leq n-1$ can be completed to a set of $n-1$ MOLR2 it becomes clear that the independence system of the first is a subset of the latter and the same holds for the clutter of circuits. Hence,

Corollary 3.2.2. $\mathcal{I}^{\mathcal{B}_t} \subset \mathcal{I}^{\mathcal{B}_{t+1}}, \quad t = 1, ..., n-2$

Corollary 3.2.3. $\mathcal{C}_t \subset \mathcal{C}_{t+1}, \quad t = 1, ..., n-2$

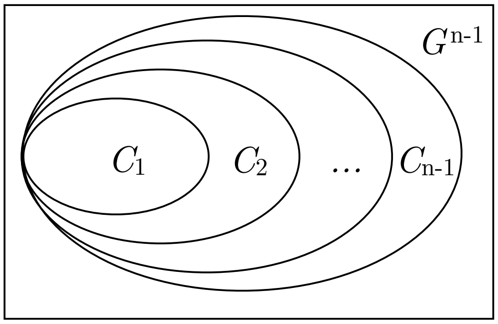


Figure 3.2.1: Venn diagram illustrating Corollary 3.2.3

Notice from Figure 3.2, that $\mathcal{C}_1 \subset \mathcal{C}_2 \subset \; ... \; \subset \mathcal{C}_{n-1}$ and therefore $\mathcal{C}_{n-1} = \mathcal{C}_1 \cup (\mathcal{C}_2 \backslash \mathcal{C}_1) \cup (\mathcal{C}_3 \backslash \mathcal{C}_2) \cup \; ... \; \cup (\mathcal{C}_{n-1} \backslash \mathcal{C}_{n-2})$.

3.3 The size of clutters $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$

Counting Latin rectangles is a topic broadly studied in combinatorics; some examples listed in chronological order are [51], [35], [28], [57]. In this Section we count the members of $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ i.e. the total number of 2-row Latin rectangles, OLR2 and sets of 3 MOLR2.

We first count members of $\mathcal{B}_1$. Normalised 2-row Latin rectangles of order n are known in the literature as *derangements* because they can each be interpreted as a permutation without a fixed point. Enumerating all derangements is described by Ryser in [57] as a classical numerical problem known as “le problème des rencontres”, for which he provides the following formula. Let $r_m(n)$ denote the number of normalised m-row Latin rectangles of order n.

$$r_2(n) = n! \sum_{q=0}^{n} (-1)^q \frac{1}{q!} \tag{3.3.1}$$

Every Latin rectangle can be normalised by permuting its columns (elements in J); one can permute the n

columns in $n!$ ways hence,

$$|\mathcal{B}_1\,| = n!\, r_2(n) \tag{3.3.2}$$

n	$\lvert\mathcal{B}_1\,\rvert$
3	12
4	216
5	5,280
6	190,800
7	598,066,560
8	598,672,880
9	48,443,028,480
10	4,844,306,476,800

Table 3.3.1: The number of members in $\mathcal{B}_1$ for various orders n

From Proposition 3.1.1 we know that instead of counting the unique number of sets of $|T|$ normalised MOLR2, one can count the total number of $(|T| + 1)$-row Latin rectangles. However, the latter count includes $|T|!$ permutations of the last $|T|$ rows which should not be included in the first count, therefore the number of normalised sets of $|T|$ MOLR2 $(|T| \leq n - 1)$ of order n, is equal to the total number of normalised $(|T| + 1)$-row Latin rectangles of order n divided by $|T|!$.

We now continue to count members of $\mathcal{B}_2$. For $|T| = 2$ we know that every OLR2 can be normalised by permuting elements in sets J and K^2. These actions yield a total of $(n!)^2$ permutations. To derive the total number of normalised OLR2 we first calculate the number of normalised 3-row Latin rectangles. From the existing formulae for this computation, some of which can be found in [34], [28], [11], we use (3.3.3) given by Riordan [55].

$$r_3(n) = \frac{1}{2}\sum_{p=0}^{n} \binom{n}{p}\, r_2(n\text{-}p)\, r_2(p)\, z(n\text{-}2p) \tag{3.3.3}$$

where,

$$z(n) = 2n\sum_{q=0}^{n} \frac{(-1)^q \binom{2n-q}{q} (n-1)!}{2n-q} \tag{3.3.4}$$

hence,

$$|\mathcal{B}_2\,| = (n!)^2\, \frac{r_3(n)}{2!} \tag{3.3.5}$$

n	$\vert\mathcal{B}_2\vert$	
3	36	
4	6,912	
5	39,744	$\times 10^2$
6	5,515,776	$\times 10^3$
7	13,637,611,008	$\times 10^3$
8	571,428,411,015,168	$\times 10^2$
9	381,406,944,838,917	$\times 10^6$
10	386,591,669,172,110	$\times 10^{10}$

Table 3.3.2: The number of members in $\mathcal{B}_2$ for various orders n

Finally, we count members of $\mathcal{B}_3$. For $|T| = 3$ we know that any set of 3 MOLR2 can be normalised by permuting elements in sets J, K^2 and K^3. These actions yield a total of $(n!)^3$ permutations. To derive the total number of normalised 3MOLR2 we first calculate the number of normalised 4-row Latin rectangles. McKay and Wanless in [51] present a table of values for $r_m(n)$ when $2 \leq m \leq n \leq 11$, which was obtained by computer enumerations. Table 3.3.3 presents for various values of order n, the total number of $r_4(n)$ as presented in [51].

n	$r_4(n)$
4	4
5	56
6	6,552
7	1,293,216
8	4,20,909,504
9	207,624,560,256
10	147,174,521,059,584

Table 3.3.3: $r_4(n)$ for various orders n

Therefore,

$$|\mathcal{B}_3| = (n!)^3 \, \frac{r_4(n)}{3!} \qquad (3.3.6)$$

n	$\vert\mathcal{B}_3\vert$	
4	9,216	
5	16,128	$\times 10^3$
6	407,586,816	$\times 10^3$
7	27,593,794,658,304	$\times 10^3$
8	4,598,318,530,415,300	$\times 10^6$
9	1,653,547,114,909,490	$\times 10^{12}$
10	1,172,115,689,907,780	$\times 10^{18}$

Table 3.3.4: The number of members in $\mathcal{B}_3$ for various orders n

3.4 Concluding remarks

In this chapter it was shown that any set of $|T|$ normalised MOLR2 of order n can be represented as a normalised $(|T|+1)$-row Latin rectangle of order n and vice versa. It was also established that deciding whether $n-1$ incomplete 2-row rectangles of order n are completable to a set of $n-1$ MOLR2 is $\mathcal{NP}$-complete. It was also proved that any circuit found for a set of $|T|$MOLR2 adds to the knowledge of the clutter of circuits associated with all sets of MOLR2 greater than $|T|$. Moreover, a polynomial time algorithm was provided for completing a set of $|T|$ MOLR2 to a set of $|T|+1$MOLR2. In the last section, the clutters $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ were counted up to $n=10$, demonstrating how large this problem is, even for the reduced case of rectangles over only two rows.

The natural next step, in terms of further research, would be to device complexity results for set of $|T|$ MOLR2 for $|T| < n-1$.

The next chapter includes the main findings of this thesis, i.e. the complete description of cutter $\mathcal{C}_2 \backslash \mathcal{C}_1$.

Chapter 4

On the clutter of circuits $\mathcal{C}_2$

This chapter focuses on the complete description of the clutter of circuits $\mathcal{C}_2$. To achieve this, it suffices to characterise all incomplete pairs of 2-row Latin rectangles that are not completable to an $OLR2$. Minimal such incomplete pairs define circuits of the independence system associated with $OLR2$ of order n and in that system, an incomplete pair of 2-row Latin rectangles is independent if and only if it is completable to an $OLR2$.

All circuits for the independence system associated with 2-row Latin rectangles and denoted by $\mathcal{C}_1$ are fully described in [29]. We take this work a step further to describe all circuits in $\mathcal{C}_2 \setminus \mathcal{C}_1$. Since there can be no polynomial time algorithm (unless $\mathcal{P} = \mathcal{NP}$) generating the clutter of either bases or circuits of an arbitrary IS [61], our work adds to the (few) independence systems in the literature, for which the clutter of circuits is fully characterised (see [61] and [47]). It can be seen as a first valuable step for the characterisation of circuits for $MOLR$, $MOLS$ and possibly other highly symmetric combinatorial problems.

The chapter is organised as follows. Section 4.1 presents up to equivalence all representatives of $\mathcal{C}_1$ as they are described by R. Euler in [29] and also counts all members in the clutter. The main findings are presented in Section 4.2 where $\mathcal{C}_2 \setminus \mathcal{C}_1$ is fully characterised. The clutter is first separated into five distinct classes and then an exhaustive procedure is followed that finds up to equivalence all representative circuits for each class. The procedure utilises the availability matrix configuration, as described in Section 2.1.4, whose symmetrical and illustrative nature helps structure the proofs. This work sets the scene for further research in this direction and the foundation for the introduction to circuits for 3 sets of MOLR2 that follows in Chapter 5. Finally, Section 4.3 presents a list of circuits in $\mathcal{C}_2$ that are also members of every clutter of circuits $\mathcal{C}_{|T|}$ where $|T| \leq n-1$.

A representation of the clutter of circuits for a set of $|T|$ MOLR2 denoted by $\mathcal{C}_{|T|}$, where $|T| \leq n-1$ is illustrated in Figure 4.0.1 in which notation $\mathcal{C}_{|T|,d}$ is used to denote circuits of type d in $\mathcal{C}_{|T|}$. It shows what was essentially concluded in Section 3.1; that $\mathcal{C}_{|T|} = \mathcal{C}_1 \cup (\mathcal{C}_2 \setminus \mathcal{C}_1) \cup \ ... \ \cup (\mathcal{C}_{|T|} \setminus \mathcal{C}_{|T|-1})$.

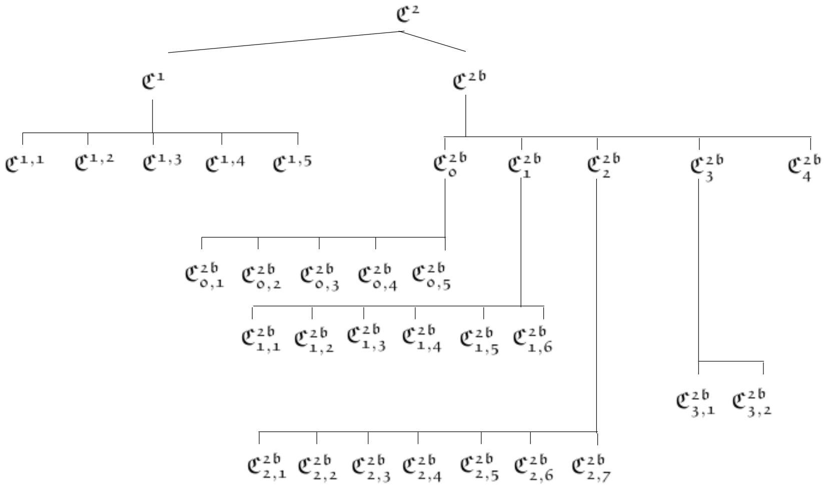

$\mathfrak{C}^2$
$\mathfrak{C}^1$
$\mathfrak{C}^{2b}$
$\mathfrak{C}^{1,1}$
$\mathfrak{C}^{1,2}$
$\mathfrak{C}^{1,3}$
$\mathfrak{C}^{1,4}$
$\mathfrak{C}^{1,5}$
$\mathfrak{C}^{2b}_0$
$\mathfrak{C}^{2b}_1$
$\mathfrak{C}^{2b}_2$
$\mathfrak{C}^{2b}_3$
$\mathfrak{C}^{2b}_4$
$\mathfrak{C}^{2b}_{0,1}$
$\mathfrak{C}^{2b}_{0,2}$
$\mathfrak{C}^{2b}_{0,3}$
$\mathfrak{C}^{2b}_{0,4}$
$\mathfrak{C}^{2b}_{0,5}$
$\mathfrak{C}^{2b}_{1,1}$
$\mathfrak{C}^{2b}_{1,2}$
$\mathfrak{C}^{2b}_{1,3}$
$\mathfrak{C}^{2b}_{1,4}$
$\mathfrak{C}^{2b}_{1,5}$
$\mathfrak{C}^{2b}_{1,6}$
$\mathfrak{C}^{2b}_{3,1}$
$\mathfrak{C}^{2b}_{3,2}$
$\mathfrak{C}^{2b}_{2,1}$
$\mathfrak{C}^{2b}_{2,2}$
$\mathfrak{C}^{2b}_{2,3}$
$\mathfrak{C}^{2b}_{2,4}$
$\mathfrak{C}^{2b}_{2,5}$
$\mathfrak{C}^{2b}_{2,6}$
$\mathfrak{C}^{2b}_{2,7}$

Figure 4.0.1: The clutter of circuits $\mathcal{C}_{|T|}$

4.1 The clutter of circuits $\mathcal{C}_1$

A detailed study of the clutter $\mathcal{C}_1$ for $n \geq 2$, is given in [29], where it is shown that there are five equivalence classes in $\mathcal{C}_1$ denoted as $\mathcal{C}_{1,d}$, $d = 1, ..., 5$. Let us give a representative from each equivalence class and count $|\mathcal{C}_1|$, assuming without loss of generality that $I = \{1, 2\}$ and $K^1 = \{1, ..., n\}$.

The representative of a circuit $C \in \mathcal{C}_{1,1}$ is shown in Table 4.1.1, where $C = \{(1, j_1, 1), (2, j_1, 1)\}$. To see that C is a circuit, check first that both the exclusion and minimality properties hold. It is obvious that C is a dependent set of $S^{\mathcal{B}_1}$, since value 1 appears twice in the column, thus ensuring that C is not contained in any member of $\mathcal{B}_1$ because it violates the definition of a Latin rectangle. The removal of any element of $c \in C$ will allow for the rectangle to be completed to a Latin rectangle and therefore C is minimal. For example, if 1 is removed from cell $(2, j_1)$ of the array, all cells can be filled in multiple ways, one of which gives the Latin rectangle in Table 2.1.4. Similarly if element 1 is removed from cell $(1, j_1)$.

j_1	... j_n
1	
1	

R_-

Table 4.1.1: Representative of $\mathcal{C}_{1,1}$

The representative from each of the remaining equivalence families is presented in Table 4.1.2. For the last two circuits presented, the notation $K^1 \backslash \{1\}$ is used to show that in columns j_2 to j_n all values of set K^1

appear with the exception of 1.

j_1	j_2	··· j_n
1	1	

R_-, representative of $\mathcal{C}_{1,2}$

j_1	··· j_n
$1,2$	

R_-, representative of $\mathcal{C}_{1,3}$

j_1	··· j_n
	$K^1 \backslash \{1\}$
	$K^1 \backslash \{1\}$

R_-, representative of $\mathcal{C}_{1,4}$

j_1	··· j_n
	$K^1 \backslash \{1\}$
1	

R_-, representative of $\mathcal{C}_{1,5}$

Table 4.1.2: Representative of $\mathcal{C}_{1,2}$, $\mathcal{C}_{1,3}$, $\mathcal{C}_{1,4}$ and $\mathcal{C}_{1,5}$

Lemma 4.1.1.

$$|\mathcal{C}_1| = n^2 \left(1 + 4(n-1) + (n-1)!^2 \sum_{q=0}^{n-1} (-1)^q \frac{1}{q!} + 2(n-1)! \right) \tag{4.1.1}$$

Proof. Family $\mathcal{C}_{1,1}$ includes n^2 circuits, since there is one such circuit per column and value i.e. per each member of J and K^1. To obtain a circuit in $\mathcal{C}_{1,2}$ (notice its representative in Table 4.1.2), there are 2 options for the row, n options for the value in K^1 and $\binom{n}{2}$ options for the two columns in which the value appears, i.e. a total of $n^2(n-1)$ options. For $\mathcal{C}_{1,3}$, the reasoning is that there are $n * (n-1)$ options for the value in each cell, n for each column and 2 for each row i.e. the total is $2(n-1)n^2$.

Regarding $\mathcal{C}_{1,4}$, notice that there are n options for the value and n options for the column which is left empty. Notice also that for columns in $J \backslash \{j_1\}$, the second row must be a derangement of the first in order to comply with the Latin rectangle structure; hence there are $r_2(n-1)$ options for filling the second row for each of the $(n-1)!$ options of filling the first one. A formula for $r_2(n)$ is given in Section 3.3. Overall, $\mathcal{C}_{1,4}$ contains $n^2 \cdot r_2(n-1) \cdot (n-1)!$ circuits. Last, $|\mathcal{C}_{1,5}| = 2n^2(n-1)!$ since there are two options for the row where a single value appears, n options for the value, n for the column and $(n-1)!$ options for filling the remaining row. The result follows from the fact that the five classes $\mathcal{C}_1$ are disjoint and therefore $|\mathcal{C}_1| = |\mathcal{C}_{1,1}| + |\mathcal{C}_{1,2}| + |\mathcal{C}_{1,3}| + |\mathcal{C}_{1,4}| + |\mathcal{C}_{1,5}|$. □

Table 4.1.3 presents for various values of order n, the total number of members in the Clutter of circuits $\mathcal{C}_1$.

n	$\|\mathcal{C}_1\|$
3	99
4	496
5	6,825
6	199,116
7	9,420,397
8	598,672,320
9	48,449,925,577
10	4,844,375,425,900

Table 4.1.3: Number of $\mathcal{C}_1$ members for various orders of n

4.2 The clutter of circuits $\mathcal{C}_2 \backslash \mathcal{C}_1$

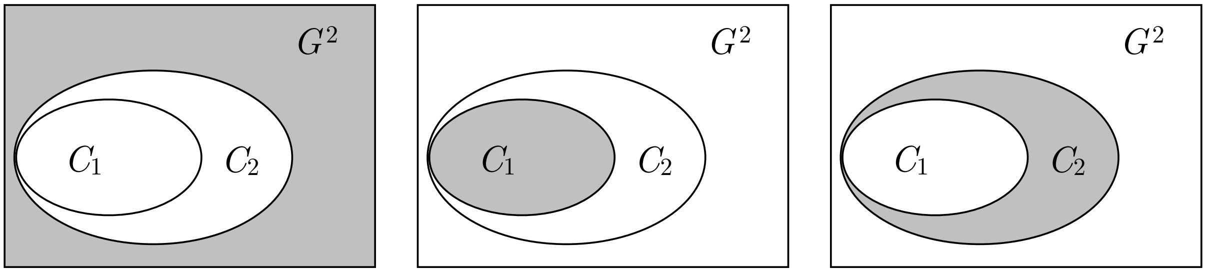
This section focuses on characterising the complete list of circuits in $\mathcal{C}_2$. A summary of all families constituting $\mathcal{C}_2$ is presented in the first two branches of Figure 4.0.1 (branches $\mathcal{C}_1$ and $\mathcal{C}_2 \backslash \mathcal{C}_1$).

Clutter $\mathcal{C}_1$ was discussed in the previous section, therefore in order to characterise all member of the Clutter $\mathcal{C}_2$, we only need to derive all circuits in $\mathcal{C}_2 \backslash \mathcal{C}_1$. These are minimal dependent subsets of $S^{\mathcal{B}_2}$, that are not members of $\mathcal{C}_1$. To examine how these different circuits affect completablilty, consider a pair of incomplete rectangles R_{1-}, R_{2-}. The following three scenarios are possible:

Scenario 1**:** The two rectangles are completable to an orthogonal pair indicating that $R_{1\_} \cup R_{2\_}$ does not violate the Latin rectangle structure (i.e does not contain a circuit in $\mathcal{C}_{1,1}, \mathcal{C}_{1,2}, \mathcal{C}_{1,3}$) and its completion will not violate this structure (i.e does not contain a circuit in $\mathcal{C}_{1,4}$, $\mathcal{C}_{1,5}$).

Scenario 2**:** Alternatively, it is possible that one or both of the rectangles comprising the pair are not individually completable to a Latin rectangle indicating that R_{1-} and/or R_{2-} contain at least one member of $\mathcal{C}_1$ and obviously the two rectangles cannot be completed to form an orthogonal pair.

Scenario 3**:** Lastly, both rectangles are individually completable to a Latin rectangle but no matter how this is done, there always appears a repetition of a pair of values when the rectangles are superimposed, thus they cannot form an orthogonal pair. This indicates that $R_{1\_} \cup R_{2\_}$ contains a member of $\mathcal{C}_2 \backslash \mathcal{C}_1$.



$G^2 \setminus \mathcal{C}_2$ (no circuit) $\mathcal{C}_1$ $\mathcal{C}_2 \setminus \mathcal{C}_1$

Figure 4.2.1: Venn diagram illustrating Scenario 1 (left), Scenario 2 (centre) and Scenario 3 (right)

To clearly demonstrate the differences between the two clutters $\mathcal{C}_1$ and $\mathcal{C}_2 \setminus \mathcal{C}_1$ we note,

Remark 4.2.1. *A circuit $C \in \mathcal{C}_2$ is,*

- *a member of $\mathcal{C}_1$ if for some $t \in T$, $C \cap G^t$ is not a member of $\mathcal{I}^{\mathcal{B}_1}$*
- *a member of $\mathcal{C}_2 \setminus \mathcal{C}_1$ if for all $t \in T$, $C \cap G^t$ is a member of $\mathcal{I}^{\mathcal{B}_1}$*

4.2.1 Class $\mathcal{C}_{2,4}$

We will now start to describe circuits in $\mathcal{C}_2 \backslash \mathcal{C}_1$. It is known that an OLR2 of order $n = 2$ (i.e. an OLS of order 2) does not exist hence there do not exist corresponding clutters of bases and circuits. Therefore circuits derived in this chapter are for orders $n > 2$. In fact, we derive circuits whose corresponding incomplete rectangles have at least 7 columns, hence $\mathcal{C}_2$ represents the complete list of circuits for orders $n \geq 7$ and for orders $2 < n < 7$ a subset of $\mathcal{C}_2$ suffices.

Two incomplete rectangles $R_{1-} \cup R_{2-}$ containing a member of $\mathcal{C}_2 \backslash \mathcal{C}_1$ implies that the completion of R_{1-} and R_{2-} will force a pair of values to be repeated. Equivalence yields that we may assume without loss of generality that this pair of values is (k_1^1, k_1^2) and that it appears twice in columns j_1 and j_2. Table 4.2.1 shows this by assuming $k_1^1 = k_1^2 = 1$; in fact, to simplify our exposition, let us hereafter assume that $I = \{1, 2\}$ and $K^1 = K^2 = \{1, \ldots, n\}$. We conclude with no loss of generality that any completion of $R_{1-} \cup R_{2-}$ that contains a member of $\mathcal{C}_2 \backslash \mathcal{C}_1$ is bound to include the set $E = \{(1, j_1, 1), (2, j_2, 1), (1, j_1, 1), (2, j_2, 1)\}$.

j_1	j_2	
1		
	1	

R_{1-}

j_1	j_2	
1		
	1	

R_{2-}

Table 4.2.1: Set E

Lemma 4.2.2. *E belongs to $\mathcal{C}_2 \backslash \mathcal{C}_1$.*

Proof. We first need to show that this is a circuit. This is true if E satisfies the exclusion and minimality property (see Definition 2.1.7). By definition it is evident that E is not contained in any member of $\mathcal{B}_2$. On the other hand, if we remove any element c of E then there does exist $B \in \mathcal{B}_2$ such that $E \backslash \{c\} \subset B$. An example is shown in Table 4.2.2 where $c = \{(2, j_2, 1)\}$. A similar example can be shown if any other element of E is removed. We have now established that $E \in \mathcal{C}_2$ and next need to show that it is not a member of $\mathcal{C}_1$. For this, it is easy to see that each incomplete rectangle in Table 4.2.1 is completable to a Latin rectangle.

j_1	j_2	...		j_n
1	2	...		n
2	3	...	n	1

R_1

j_1	j_2	j_3	...	j_n
1	2		...	n
n	1	2	...	$n-1$

R_2

Table 4.2.2: An OLR2

□

Since $|E| = 4$ then up to equivalence, any circuit member of $\mathcal{C}_2 \backslash \mathcal{C}_1$, may contain 0 up to 4 elements of E, leading to five different classes of circuits in $\mathcal{C}_2 \backslash \mathcal{C}_1$. More formally,

Definition 4.2.3. *Let $d = 0, \ldots, 4$ define the class of a circuit and $\mathcal{C}_{2,d} \subset \mathcal{C}_2 \backslash \mathcal{C}_1$, where $\mathcal{C}_{2,d} = \{C \in \mathcal{C}_2 \backslash \mathcal{C}_1 : C$ is equivalent to some C' such that $|C' \cap E| = d\}$; evidently the classes $\{\mathcal{C}_{2,d},\ d = 0, \ldots, 4\}$ form a partition of $\mathcal{C}_2 \backslash \mathcal{C}_1$ hence,*

$$\mathcal{C}_2 \backslash \mathcal{C}_1 = \mathcal{C}_{2,4} \cup \mathcal{C}_{2,3} \cup \mathcal{C}_{2,2} \cup \mathcal{C}_{2,1} \cup \mathcal{C}_{2,0} \tag{4.2.1}$$

For class 4 the obvious result follows,

Theorem 4.2.4. *The incomplete pair of rectangles of Table 4.2.1 comprises up to equivalence, the single member of $\mathcal{C}_{2,4}$.*

For all other classes we provide a list of all non-equivalent circuits (i.e., we catalogue all sub-classes that each class splits into and list one representative per sub-class). For such cases, let $\mathcal{C}^v_{2,d}$ denote the subfamily containing the v^{th} circuit of class d. For example $\mathcal{C}^1_{2,2}$ denotes the 1^{st} circuit in class 2.

We have established that class 4 contains, up to equivalence, only one member. We continue to find all circuit members of classes 0 to 3. It turns out that as summarised in Figure 4, classes $0,\ 1,\ 2,\ 3$ respectively have , $5,\ 6,\ 7$ and 2 sub-families of circuits. We start by investigating class 0.

4.2.2 Class $\mathcal{C}_{2,0}$

Proposition 4.2.5. *Let $\mathcal{Q}$ denote the collection of all pairs of Latin rectangles R_1, R_2 that do not form an orthogonal pair due to the repetition of at least one pair of values in the first two columns. Then for each $C \in \mathcal{C}_2 \backslash \mathcal{C}_1$ there exists $Q \in \mathcal{Q}$ such that $C \subset Q$.*

Proof. We know that for all $C \in \mathcal{C}_2 \backslash \mathcal{C}_1$, there exist $R_1, R_2 \in \mathcal{B}_1$ such that $C \subset (R_1 \cup R_2)$ and $E \subset (R_1 \cup R_2)$. Therefore, the collection of all pairs of Latin rectangles Q, in which there appears a repetition of a pair of values, contains the complete list of $C \in \mathcal{C}_2 \backslash \mathcal{C}_1$. □

Proposition 4.2.6. *For all $Q \in \mathcal{Q}$, $Q \backslash E$ contains a circuit member of class* 0.

Proof. To obtain a circuit of class $\mathcal{C}_{2,0}$ we start with a member of $\mathcal{Q}$ and let this be $Q = R_1 \cup R_2$. Notice that the set $(R_1 \cup R_2) \backslash E$ continues to satisfy the exclusion property, since completing each of the rectangles corresponding to $R_1 \backslash E$ and $R_2 \backslash E$ forces the repetition of pair $(1, 1)$ in cells $(1, j_1)$ and $(2, j_2)$ (see Tables 4.2.3 and 4.2.4). Therefore $(R_1 \cup R_2) \backslash E$ contains a circuit and due to the absence of all elements of E, we know that it contains a circuit in $\mathcal{C}_{2,0}$ (class 0). □

j_1	j_2	...	j_n
1	$K^1 \backslash \{1\}$		
k_1	1	$K \backslash \{k_1, 1\}$	

R_1

j_1	j_2	...	j_n
1	$K^2 \backslash \{1\}$		
k_2	1	$K^2 \backslash \{k_2, 1\}$	

R_2

Table 4.2.3: A pair of Latin rectangles in $\mathcal{Q}$

j_1	j_2	...	j_n
	$K^1 \backslash \{1\}$		
k_1		$K^1 \backslash \{k_1, 1\}$	

$R_1 \backslash E$

j_1	j_2	...	j_n
	$K^2 \backslash \{1\}$		
k_2		$K^2 \backslash \{k_2, 1\}$	

$R_2 \backslash E$

Table 4.2.4: Removing E from Table 4.2.3

It is also easy to see from Table 4.2.4 that $(R_1 \cup R_2) \backslash E$, does not satisfy the minimality property, since if element $(2, j_1, k_1)$ is removed (or in other words cell $(2, j_1)$ is emptied from R_1 in Table 4.2.4), then the exclusion property still holds. However, if we continue to remove elements of $(R_1 \cup R_2) \backslash E$, eventually the minimality property will be met and we will have a circuit in $\mathcal{C}_{2,0}$. To make our search more concise, we first observe that in attempting completion after removing the additional elements from $R_1 \backslash E$ or $R_2 \backslash E$, some value must be forced to appear twice in the first two columns, i.e., in cells $(1, j_1), (1, j_2), (2, j_1)$ and $(2, j_2)$. For convenience, let us introduce the following.

Definition 4.2.7. *An incomplete rectangle R_- is called* ***pink*** *if :*

i) $\exists R'_-$ such that $R_- \subset R'_-$ and R'_- includes elements $(1, j_1, 1), (2, j_2, 1)$ and

ii) any completion of R_- forces some value to appear twice in the first two columns

Notice that by Definition 4.2.3, for a pink rectangle R_- in class 0, $R_- \cap E = \emptyset$. Our exhaustive, yet concise, procedure, to reveal all non-equivalent circuits in $\mathcal{C}_{2,0}$ is the following:

Step 1**:** Identify all pairwise **non-equivalent pink** (incomplete) rectangles by systematically emptying cells form Table 4.2.4

Step 2**:** Combine them to obtain incomplete **pairs** of pink rectangles in all possible ways

Step 3**:** Finally, omit any incomplete pair that is not **minimal**

After completing the 3-step procedure we will obtain all circuits of class 0. Notice that Steps 1 and Step 2 will be performed on one rectangle and then as indicated in Step 3 pairs of incomplete rectangles will be created. We introduce the following definition to describe the different types of rectangles derived from Step 1.

Definition 4.2.8. *A pink incomplete rectangle R_- is of* ***type I**, **II** or **III**, if its completion forces the same value to appear respectively as follows,*

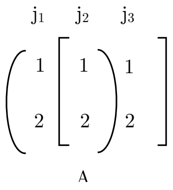
type I: *always and only in cells $\{(1, j_1), (2, j_2)\}$, irrespective of how R_- is completed*

type II: *either in cells $\{(1, j_1), (2, j_2)\}$ and/or in cells $\{(1, j_2), (2, j_1)\}$, depending on how R_- is completed*

type III: *always in both, cells $\{(1, j_1), (2, j_2)\}$ and in cells $\{(1, j_2), (2, j_1)\}$, irrespective of how R_- is completed*

The difference between type II and type III is that irrespective of the way in which an incomplete rectangle of type III is completed, a repetition of a values will always appear in cells $\{(1, j_1), (2, j_2)\}$ **and** $\{(1, j_2), (2, j_1)\}$, which is not the case for an incomplete rectangle of type II. Example 4.2.1 helps to clarify the three types of pink rectangles.

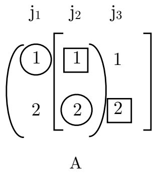
Example 4.2.1
Table 4.2.5 depicts a type I pink rectangle R_-. Notice that in cell $(1, j_1)$ of the array we can place value 1 or 2. The value we choose will be forced to also appear in cell $(2, j_2)$, therefore R_- is type I. We present the corresponding availability matrix next to it.



j_1	j_2	j_3	$\cdots$	j_n
		$K\backslash\{1,2\}$		
n			$K\backslash\{1,2\}$	

R_-

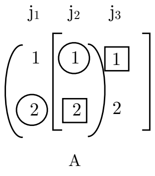
Table 4.2.5: R_- of type I and corresponding availability matrix



j_1	j_2	j_3	...	j_n
1	2	$K\backslash\{1,2\}$		
n	1	2	$K\backslash\{1,2\}$	

R

or

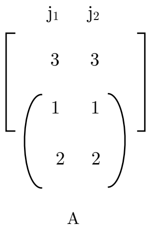


j_1	j_2	j_3	...	j_n
2	1	$K\backslash\{1,2\}$		
n	2	1	$K\backslash\{1,2\}$	

R

Table 4.2.6: All possible ways of completing R_- of Table 4.2.5

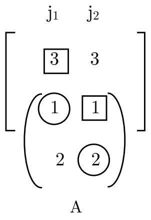
Table 4.2.7 depicts a type II pink rectangle R_-. Notice that value 1 can be placed either in cells $(1, j_1)$, $(2, j_2)$ or in cells $(1, j_2)$, $(2, j_1)$; therefore R_- is type II.



j_1	j_2	...	j_n
		$K\backslash\{1,2\}$	
		$K\backslash\{1,3\}$	

R_-

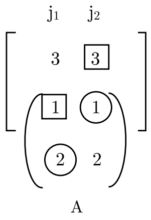
Table 4.2.7: R_- of type II and corresponding availability matrix



j_1	j_2	...	j_n
1	2	$K\backslash\{1,2\}$	
3	1	$K\backslash\{1,3\}$	

R

or



j_1	j_2	...	j_n
2	1	$K\backslash\{1,2\}$	
1	3	$K\backslash\{1,3\}$	

R

Table 4.2.8: All possible ways of completing R_- of Table 4.2.7

Finally, Table 4.2.9 depicts a type III rectangle R_-. One can easily observe that no matter how values are placed in the empty cells, they will always form a Latin square of order 2. Since same values will always appear in cells $(1, j_1)$, $(2, j_2)$ and cells $(1, j_2)$, $(2, j_1)$, R_- is type III.

$$\begin{array}{c} \begin{array}{cc} \mathrm{j}_1 & \mathrm{j}_2 \end{array} \\ \left[\left(\begin{array}{cc} 1 & 1 \\ 2 & 2 \end{array}\right)\right] \\ \mathrm{A} \end{array}$$

j_1	j_2	... j_n
		$K\backslash\{1,2\}$
		$K\backslash\{1,2\}$

R_-

Table 4.2.9: R_- of type III and corresponding availability matrix

$$\begin{array}{c} \begin{array}{cc} \mathrm{j}_1 & \mathrm{j}_2 \end{array} \\ \left[\left(\begin{array}{cc} \textcircled{1} & \boxed{1} \\ \boxed{2} & \textcircled{2} \end{array}\right)\right] \\ \mathrm{A} \end{array}$$

j_1	j_2	... j_n
1	2	$K\backslash\{1,2\}$
2	1	$K\backslash\{1,2\}$

R

or

$$\begin{array}{c} \begin{array}{cc} \mathrm{j}_1 & \mathrm{j}_2 \end{array} \\ \left[\left(\begin{array}{cc} \boxed{1} & \textcircled{1} \\ \textcircled{2} & \boxed{2} \end{array}\right)\right] \\ \mathrm{A} \end{array}$$

j_1	j_2	... j_n
2	1	$K\backslash\{1,2\}$
1	2	$K\backslash\{1,2\}$

R

Table 4.2.10: All possible ways of selecting values in A of Table 4.2.9

Lemma 4.2.9. *Two pink rectangles form a pair whose completion forces a repetition of a pair of values in the first two columns if and only if both are of type I or at least one is of type III.*

Thus, two pink rectangles R_- and R'_- comply with Lemma 4.2.9 if:

i) both R_- and R'_- are of type I or

ii) R_- is of type I and R'_- is of type III or

iii) R_- is of type II and R'_- is of type III or (iv) both R_- and R'_- are of type III.

Based on the fact that the type of pink rectangles in any pair $R_- \cup R'_-$ determines whether their completion will result in a repetition of a pair of values in the first two columns, we can derive a necessary condition for minimality (see Remark 4.2.11). To that end, we use the concept of a *dominated* pink rectangle as defined below.

Definition 4.2.10. *A pink rectangle R_- is called* ***dominated*** *if there is some pink rectangle $R'_- \subset R_-$ such that R_- and R'_- are both of the same type (i.e., I, II, III).*

Remark 4.2.11. *In order for a dependent set $R_- \cup R'_-$ to be minimal, it is necessary that neither R_- nor R'_- are dominated.*

Let us list a few last observations, to be utilized in the proof that follows.

Remark 4.2.12. *Up to equivalence, for any two R_- and R'_-, $R_- \subset R'_-$ if and only if $A' \subset A$.*

Remark 4.2.13. *If two pairs of incomplete rectangles $(R_- \cup R'_-), (R''_- \cup R'''_-) \in G^2$ both satisfy the exclusion property and $(R_- \cup R'_-) \subset (R''_- \cup R'''_-)$ then $R''_- \cup R'''_-$ is not minimal and consequently not a circuit. For the corresponding availability matrices it follows that if $(A'' \cup A''') \subset (A \cup A')$, then $R''_- \cup R'''_-$ is not minimal.*

We proceed to find all circuits of $\mathcal{C}_{2,0}$, by first following Step 1. To derive all pink rectangles, we will progressively empty cells from the left rectangle of Table 4.2.4. To avoid enumerating equivalent (i.e., symmetric) cases, we assume that the number of cells emptied from the first row are less than or equal to the number of cells emptied from the second row. For each case, we illustrate the availability matrix A_z of the rectangle R_{z-} obtained after emptying cells. An '$*$' besides a value in A_z denotes that that value is not available (e.g., due to its occurrence in a non-empty cell in the same column); we emphasize the occurrence of some '$*$' in A_z by writing, instead, A^*_z and R^*_z. We make the following observations for Step 1:

Observation 1**:** The resulting two availability matrices (one for each row) must share at least one row containing value 1, since E has been removed from $R_- \cup R'_-$ and therefore value 1 is now available in both rows.

Observation 2**:** Remember that $A(R_-, 1)$ denotes the availability matrix of the first row and $A[R_-, 2]$ of the second and $A = A(R_-, 1) \cup A[R_-, 2]$. It follows from the definition of a Latin rectangle, that an available value in column j of $A(R_-, 1)$ cannot be selected if it already appears in cell $(2, j)$ of the corresponding array. The same can be said for $A[R_-, 2]$ and cell $(1, j)$. Hence,

2.1 An '$*$' appears in column j of $A(R_-, 1)$ only if column j does not appear in $A[R_-, 2]$. This is evident, since cell $(2, j)$ of the array is filled and therefore there is no available value. Similarly for an '$*$' appearing in $A[R_-, 2]$.

2.2 An '$*$' appears on an element of $A(R_-, 1)$ only if this element does not appear in $A[R_-, 2]$. This is clear, since an '$*$' in $A(R_-, 1)$ indicates that element appears in row 2 of the corresponding array and therefore it is not available, hence it will not appear in $A[R_-, 2]$. Similarly for an '$*$' appearing in $A[R_-, 2]$.

2.3 At most one '$*$' can appear in each column of $A(R_-, 1)$ (or $A[R_-, 2]$). This is also clear, since more than one '$*$' in column j of $A(R_-, 1)$ indicates that cell $(2, j)$ of the corresponding array contains more than one elements and therefore the definition of a Latin rectangle is violated.

Remark 4.2.14. *Consider an incomplete rectangle R_- for which $A(R_-, 1)$ has dimension p and $A[R_-, 2]$ has dimension q, where $p \leq q$. Then the two availability matrices can share* 1 *up to p rows and* 0 *up to p columns.*

Proposition 4.2.15. *Tables 4.2.11 and 4.2.12 contain all pink rectangles, which share no element with E.*

j_1	j_2	j_3	...	j_n
	2	$K^1\backslash\{1,2\}$		
k_1			$K^1\backslash\{1,2,k_1\}$	

R^*_{1-}

, $A^*_1 = \{(1,j_1,1),(2,j_2,1),(2,j_3,1)(2,j_3,2)\}$

j_1	j_2	...	j_n
	k_2	$K^1\backslash\{1,k_2\}$	
		$K^1\backslash\{1,2\}$	

R_{2-}

, $A_2 = \{(1,j_1,1),(2,j_1,1),(2,j_1,2),(2,j_2,1),(2,j_2,2)\}$

j_1	j_2	...	j_n
	2	$K^1\backslash\{1,2\}$	
		$K^1\backslash\{1,2\}$	

R^*_{2-}

, $A^*_2 = \{(1,j_1,1),(2,j_1,1),(2,j_1,2),(2,j_2,1)\}$

j_1	j_2	j_3	j_n
	2	3	$K^1\backslash\{1,2,3\}$
			$K^1\backslash\{1,2,3\}$

R^*_{4-}

, $A^*_4 = \{(1,j_1,1),(2,j_1,1),(2,j_1,2),(2,j_1,3),(2,j_2,1),$
$(2,j_2,3),(2,j_3,1),(2,j_3,2)\}$

j_1	j_2	j_3	j_4	...	j_n
	3		$K_1\backslash\{1,2,3\}$		
2		n		$K_1\backslash\{1,2,3,n\}$	

R^*_{6-}

, $A^*_6 = \{(1,j_1,1),(1,j_3,1),(1,j_3,2),(2,j_2,1),$
$,(2,j_4,1),(2,j_4,3)\}$

j_1	j_2	j_3	...	j_n
		$K_1\backslash\{1,2\}$		
2			$K_1\backslash\{1,2,3\}$	

R^*_{7-}

, $A^*_7 = \{(1,j_1,1),(1,j_2,1),(1,j_2,2),(2,j_2,1),(2,j_2,3),$
$(2,j_3,1),(2,j_3,3)\}$

j_1	j_2	...	j_n
		$K^1\backslash\{1,2\}$	
		$K^1\backslash\{1,3\}$	

R_{8-}

, $A_8 = \{(1,j_1,1),(1,j_1,2),(1,j_2,1),(1,j_2,2),(2,j_1,1)$
$(2,j_1,3),(2,j_2,1),(2,j_2,3)\}$

j_1	j_2	j_3	...	j_n
		$K_1\backslash\{1,2\}$		
n			$K_1\backslash\{1,2,n\}$	

R_{10-}

, $A_{10} = \{(1,j_1,1),(1,j_1,2),(1,j_2,1),(1,j_2,2),(2,j_2,1),$
$(2,j_2,2),(2,j_3,1),(2,j_3,2)\}$

j_1	j_2	...	j_n
		$K_1\backslash\{1,2\}$	
		$K_1\backslash\{1,2\}$	

R_{11-}

, $A_{11} = \{(1,j_1,1),(1,j_1,2),(1,j_2,1),(1,j_2,2),(2,j_1,1)$
$(2,j_1,2),(2,j_2,1),(2,j_2,2)\}$

j_1	j_2	j_3	...	j_n
		k_1	$K_1\backslash\{1,2,k_1\}$	
			$K_1\backslash\{1,2,3\}$	

R_{15-}

, $A_{15} = \{(1,j_1,1),(1,j_1,2),(1,j_2,1),(1,j_2,2),(2,j_1,1),$
$(2,j_1,2),(2,j_1,3),(2,j_2,1),(2,j_2,2),(2,j_2,3),$
$(2,j_3,1),(2,j_3,2),(2,j_3,3)\}$

Table 4.2.11: Pink rectangles that share no element with E (continued in Table 4.2.11)

j_1	j_2	j_3	... j_n
		3	$K_1\backslash\{1,2,3\}$
			$K_1\backslash\{1,2,3\}$

R^*_{15-} ,

$$A^*_{15} = \{(1,j_1,1),(1,j_1,2),(1,j_2,1),(1,j_2,2),(2,j_1,1),(2,j_1,2),(2,j_1,3),(2,j_2,1),(2,j_2,2),(2,j_2,3),(2,j_3,1),(2,j_3,2)\}$$

j_1	j_2	j_3	j_4	... j_n
			$K_1\backslash\{1,2,3\}$	
		n		$K_1\backslash\{1,2,3,n\}$

R_{23-} ,

$$A_{23} = \{(1,j_1,1),(1,j_1,2),(1,j_1,3),(1,j_2,1),(1,j_2,2),(1,j_2,3),(1,j_3,1),(1,j_3,2),(1,j_3,3),(2,j_1,1),(2,j_1,2),(2,j_1,3),(2,j_2,1),(2,j_2,2),(2,j_2,3),(2,j_4,1),(2,j_4,2),(2,j_4,3)\}$$

j_1	j_2	j_3	... j_n
			$K_1\backslash\{1,2,3\}$
			$K_1\backslash\{1,2,3\}$

R_{24-} ,

$$A_{24} = \{(1,j_1,1),(1,j_1,2),(1,j_1,3),(1,j_2,1),(1,j_2,2),(1,j_2,3),(1,j_3,1),(1,j_3,2),(1,j_3,3),(2,j_1,1),(2,j_1,2),(2,j_1,3),(2,j_2,1),(2,j_2,2),(2,j_2,3),(2,j_3,1),(2,j_3,2),(2,j_3,3)\}$$

Table 4.2.12: Pink rectangles that share no element with E

Proof. We proceed by following Step 1 and systematically emptying cells from Table 4.2.4, giving rise to five cases.

Case 1. *Emptying* 1 *cell in row* 2

Without loss of generality we assume that the cell emptied is either $(2, j_3)$ or $(2, j_1)$ and that the two values missing from row 2 are $\{1, 2\}$. Emptying cell $(2, j_3)$ gives rise to R_{1-} or R^*_{1-} (Tables 4.2.11 and 4.2.13 respectively) depending on whether value 2 appears in cell $(1, j_2)$; emptying cell $(2, j_1)$ results, in a similar manner, to R_{2-} or R^*_{2-} (Tables 4.2.11 and 4.2.13). Please notice the corresponding availability matrices in Figure 4.2.2 and the '$*$' in matrix A^*_1 regarding value 2 for the second row and column j_2 (and similarly in A^*_2); notice also that, for each matrix, we illustrate the number of rows and columns that are common to the availability matrix of both rows as indicated in Remark 4.2.14 (at least one row is common since value 1 is missing from both rows). In Figure 4.2.2 the comment "A_1: Row 1, Col 0 " explains that $A(R_{1-}, 1)$ and $A[R_{1-}, 2]$ have 1 row in common and 0 columns in common; this becomes more visible in Tables 4.2.11 and 4.2.13 where it is clear that value 1 is the only one that does not appear in both rows, and rows 1 and 2 share no columns of empty cells.

$$\begin{array}{cc} j_1 & \begin{array}{cc} j_2 & j_3 \end{array} \\ (1) & \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix} \end{array}$$

A$_1$: Row 1, Col 0

$$\begin{array}{cc} j_1 & \begin{array}{cc} j_2 & j_3 \end{array} \\ (1) & \begin{bmatrix} 1 & 1 \\ 2^* & 2 \end{bmatrix} \end{array}$$

A$^*_1$: Row 1, Col 0

$$\begin{array}{c} \begin{array}{cc} j_1 & j_2 \end{array} \\ \begin{bmatrix} (1) & 1 \\ 2 & 2 \end{bmatrix} \end{array}$$

A$_2$: Row 1, Col 1

$$\begin{array}{c} \begin{array}{cc} j_1 & j_2 \end{array} \\ \begin{bmatrix} (1) & 1 \\ 2 & 2^* \end{bmatrix} \end{array}$$

A$^*_2$: Row 1, Col 1

Figure 4.2.2: Availability matrices for Case 1

j_1	j_2	j_3	j_4 ... j_n
	k_2	$K^1\backslash\{1,k_2\}$	
k_1			$K^1\backslash\{1,2,k_1\}$

R_{1-}

j_1	j_2	j_3	j_4	j_n
	k_2	k_3	k_4	$K^1\backslash\{1,k_2,k_3,k_4\}$
k_1				$K^1\backslash\{1,2,3,k_1\}$

R_{3-}

j_1	j_2	j_3	j_4	j_n
	2	3	k_4	$K^1\backslash\{1,2,3,k_4\}$
k_1				$K^1\backslash\{1,2,3,k_1\}$

R^*_{3-}

j_1	j_2	j_3	j_n
	k_2	k_3	$K^1\backslash\{1,k_2,k_3\}$
			$K^1\backslash\{1,2,3\}$

R_{4-}

j_1	j_2	j_3	j_4	j_n
	2	3	4	$K^1\backslash\{1,2,3,4\}$
				$K^1\backslash\{1,2,3,4\}$

R^*_{5-}

j_1	j_2	j_3	... j_n
		$K^1\backslash\{1,2\}$	
n			$K^1\backslash\{1,3,n\}$

R_{7-}

j_1	j_2	j_3	j_4	j_n
	k_1		$K^1\backslash\{1,2,k_1\}$	
k_2		k_3		$K^1\backslash\{1,2,k_2,k_3\}$

R_{9-}

j_1	j_2	j_3	j_4	j_n
		3	4	$K^1\backslash\{1,2,3,4\}$
k_1				$K^1\backslash\{1,3,4,k_1\}$

R^*_{12-}

j_1	j_2	j_3	j_n
		3	$K^1\backslash\{1,2,3\}$
			$K^1\backslash\{1,3,4\}$

R^*_{13-}

j_1	j_2	j_3	j_4	j_n
		3	$K^1\backslash\{1,2,3\}$	
k_1				$K^1\backslash\{1,2,3,k_1\}$

R^*_{14-}

j_1	j_2	j_3	j_4	j_5	j_n
			4	5	$K^1\backslash\{1,2,3,4,5\}$
2		3			$K^1\backslash\{1,2,3,4,5\}$

R^*_{16-}

j_1	j_2	j_3	j_4	j_n
			5	$K^1\backslash\{1,2,3,5\}$
2				$K^1\backslash\{1,2,4,5\}$

R^*_{17-}

j_1	j_2	j_3	j_n
			$K^1\backslash\{1,2,3\}$
			$K^1\backslash\{1,4,5\}$

R_{18-}

j_1	j_2	j_3	j_4	j_5	j_n
			4	$K^1\backslash\{1,2,3,4\}$	
3		k_1			$K^1\backslash\{1,2,3,4,k_1\}$

R^*_{19-}

j_1	j_2	j_3	j_4	j_n
			k_1	$K^1\backslash\{1,2,3,k_1\}$
		k_2		$K^1\backslash\{1,2,k_2,4\}$

R_{20-}

j_1	j_2	j_3	j_4	j_n
			4	$K^1\backslash\{1,2,3,4\}$
		3		$K^1\backslash\{1,2,3,4\}$

R^*_{20-}

j_1	j_2	j_3	j_n
			$K^1\backslash\{1,2,3\}$
			$K^1\backslash\{1,2,4\}$

R_{21-}

j_1	j_2	j_3	j_4	j_5	j_n
			$K^1\backslash\{1,2,3\}$		
k_1		k_2			$K^1\backslash\{1,2,3,k_1,k_2\}$

R_{22-}

Table 4.2.13: Non-pink rectangles that share no element with E

It is easy to see from Figure 4.2.3 that R_{1-} is not pink and therefore is excluded from our analysis. Notice that value 1 is selected in column j_1 of the first matrix (i.e. first row of R_{-}) and in column j_3 of the second matrix (i.e. second row of R_{-}). This is a non-pink structure as value one is not forced to be selected in both j_1 and j_2.

$$\begin{array}{cc} & \begin{array}{ccc} j_1 & j_2 & j_3 \end{array} \\ (\textcircled{1}) & \begin{bmatrix} 1 & \boxed{1} \\ \boxed{2} & 2 \end{bmatrix} \end{array}$$

A_1: Row 1, Col 0

Figure 4.2.3: Completion of non-pink R_{1-}

For completion of R^*_{1-}, R_{2-} and R^*_{2-} we note that value 1 will appear in the first two columns with any selection, as shown in Figure 4.2.4; therefore all three rectangles are pink. Figure 4.2.4 illustrates all possible selection of values and the corresponding completed rectangles are shown in Tables 7.5 to A.4 in Appendix A.

$$\begin{array}{cc} & \begin{array}{ccc} j_1 & j_2 & j_3 \end{array} \\ (\textcircled{1}) & \begin{bmatrix} \boxed{1} & 1 \\ 2^* & \boxed{2} \end{bmatrix} \end{array}$$

A^*_1: Row 1, Col 0

$$\begin{array}{c} \begin{array}{cc} j_1 & j_2 \end{array} \\ \begin{bmatrix} (\textcircled{1}) & \boxed{1} \\ \boxed{2} & 1 \end{bmatrix} \end{array}$$

A_2: Row 1, Col 1

$$\begin{array}{c} \begin{array}{cc} j_1 & j_2 \end{array} \\ \begin{bmatrix} (\textcircled{1}) & \boxed{1} \\ \boxed{2} & 2^* \end{bmatrix} \end{array}$$

A^*_2: Row 1, Col 1

Figure 4.2.4: Completion of pink R^*_{1-}, R_{2-}, R^*_{2-}

Case 2. *Emptying* 2 *cells in row* 2

We assume the cells emptied are $(2, j_3)$ and either $(2, j_4)$ or $(2, j_1)$ and that the three values missing from row 2 are $\{1, 2, 3\}$. Emptying cells $(2, j_3)$ and $(2, j_4)$ gives rise to R_{3-} or R^*_{3-} (in Table 4.2.13) depending on whether values 2 and 1 appear in cells $(1, j_2)$ and $(1, j_3)$; emptying cells $(2, j_1)$ and $(2, j_3)$ results, in a similar manner, to R_{4-} or R^*_{4-} (in Tables 4.2.13 and 4.2.11 respectively). The corresponding availability matrices are presented in Figure 4.2.5.

$$\begin{array}{cc} & \begin{array}{cccc} j_1 & j_2 & j_3 & j_4 \end{array} \\ (\ 1\ ) & \begin{bmatrix} 1 & 1 & 1 \\ 2^* & 2 & 2 \\ 3 & 3^* & 3 \end{bmatrix} \end{array}$$

A_3: Row 1, Col 0

$$\begin{array}{cc} & \begin{array}{cccc} j_1 & j_2 & j_3 & j_4 \end{array} \\ (\ 1\ ) & \begin{bmatrix} 1 & 1 & 1 \\ 2^* & 2 & 2 \\ 3 & 3^* & 3 \end{bmatrix} \end{array}$$

A^*_3: Row 1, Col 0

$$\begin{array}{c} \begin{array}{ccc} j_1 & j_2 & j_3 \end{array} \\ \begin{bmatrix} (\ 1\ ) & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix} \end{array}$$

A_4: Row 1, Col 1

$$\begin{array}{c} \begin{array}{ccc} j_1 & j_2 & j_3 \end{array} \\ \begin{bmatrix} (\ 1\ ) & 1 & 1 \\ 2 & 2^* & 2 \\ 3 & 3 & 3^* \end{bmatrix} \end{array}$$

A^*_4: Row 1, Col 1

Figure 4.2.5: Availability matrices for Case 2

It becomes easy to see that R_{3-} is not pink since it is completable by placing 1 in cells $(1, j_1)$ and $(2, j_4)$, 2 in cell $(2, j_3)$ and 3 in cell $(3, j_2)$; in fact, R^*_{3-} (that is more 'restricted' than R_{3-} since $A^*_{3-} \subset A_{3-}$) is not pink either, since completable in exactly the same manner. This fact gives us a useful rule to avoid examining some rectangles: if R^*_{t-} is non-pink, then so is R_{t-}.

The opposite does not hold. For example, R_{4-} is not pink since completable by placing 1 in cells $(1, j_1)$ and $(2, j_3)$, 3 in cell $(2, j_1)$ and 2 in cell $(2, j_2)$; to the contrary R^*_{4-} is completable only by placing either 1 in cells $(1, j_1)$ and $(2, j_2)$ or 2 in cells $(1, j_2)$ and $(2, j_1)$ thus being pink. An illustration is presented in Figure 4.2.7. The availability matrices of the rectangles, shown in Figure 4.2.6, illustrate the completion of non-pink rectangles R_{3-}, R^*_{3-} and R_{4-} and corresponding completed rectangles are shown in Tables 7.5 to A.4 in Appendix A.

$$\begin{array}{c|ccc|} j_1 & j_2 & j_3 & j_4 \\ (\textcircled{1}) & 1 & 1 & \boxed{1} \\ & 2 & \boxed{2} & 2 \\ & \boxed{3} & 3 & 3 \end{array}$$

A$_3$: Row 1, Col 0

$$\begin{array}{c|ccc|} j_1 & j_2 & j_3 & j_4 \\ (\textcircled{1}) & 1 & 1 & \boxed{1} \\ & 2^* & \boxed{2} & 2 \\ & \boxed{3} & 3^* & 3 \end{array}$$

A*$_3$: Row 1, Col 0

$$\begin{array}{ccc} j_1 & j_2 & j_3 \\ (\textcircled{1}) & 1 & \boxed{1} \\ \boxed{2} & 2 & 2 \\ 3 & \boxed{3} & 3 \end{array}$$

A$_4$: Row 1, Col 1

Figure 4.2.6: Completion of non-pink rectangles R_{3-}, R^*_{3-}, R_{4-}

$$\begin{array}{ccc} j_1 & j_2 & j_3 \\ (\textcircled{1}) & \boxed{1} & 1 \\ 2 & 2^* & \boxed{2} \\ \boxed{3} & 3 & 3^* \end{array} \quad \text{or} \quad \begin{array}{ccc} j_1 & j_2 & j_3 \\ (\textcircled{1}) & 1 & \boxed{1} \\ \boxed{2} & 2^* & 2 \\ 3 & \boxed{3} & 3^* \end{array}$$

A*$_4$: Row 1. Col 1 A*$_4$: Row 1. Col 1

Figure 4.2.7: Completion of pink R^*_{4-}

Case 3. *Emptying at least* 3 *cells in row* 2

This case yields only non-pink rectangles. To see this, observe that the most restricted rectangle is the one arising after emptying exactly 3 cells in row 2, namely $(2, j_1), (2, j_3)$ and $(2, j_4)$, and in addition, having values 2 and 3 appearing in cells $(1, j_2)$ and $(1, j_3)$, respectively. The availability matrix shown in Figure 4.2.8 illustrates that this rectangle, namely R^*_{5-} is non-pink, i.e., its selected entries show how R^*_{5-} (in Table 4.2.13 is completable without any value appearing twice in the first two columns (in fact, the completion is made as for R_{3-} and R^*_{3-} in Case 2). Once again, the corresponding completed rectangles are shown in Tables 7.5 to A.4 in Appendix A and in fact, this will be assumed for all availability matrices from now on.

$$\begin{array}{cccc} j_1 & j_2 & j_3 & j_4 \\ \left[\begin{array}{cccc} (\textcircled{1}) & 1 & \boxed{1} & 1 \\ 2 & 2^* & 2 & \boxed{2} \\ 3 & \boxed{3} & 3^* & 3 \\ \boxed{4} & 4 & 4 & 4 \end{array}\right] \end{array}$$

A^*_5: Row 1, Col 1

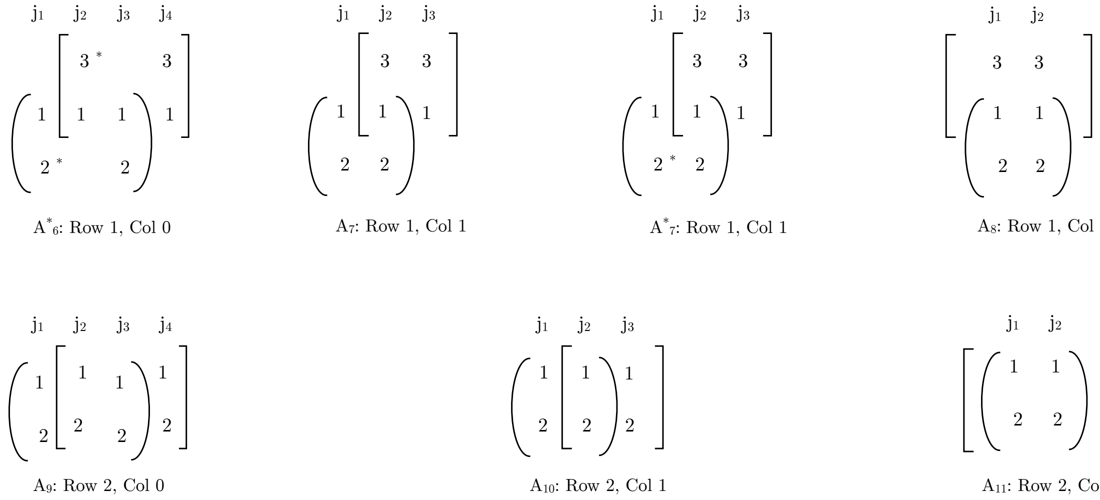
Figure 4.2.8: Completion of non-pink R^*_{5-}

No further cases need to be examined in which more than 1 cell is emptied form the first row and more than 3 values are emptied from the second row.

Case 4. *Emptying* 1 *cell in row* 1 *and* 1 *cell in row* 2

For all rectangles in this case, notice that row 1 has two empty cells hence two missing values $\{1, 2\}$; the same applies to row 2, apart from the fact that the second missing value may be 2 or not, i.e., the values missing from row 2 can be either $\{1, 2\}$ or $\{1, 3\}$. Thus, there is a 2×2 availability matrix per row and these two matrices share 1 or 2 rows (if the second value missing from row 2 is 2 or 3, respectively) and $0, 1$ or 2 columns (depending on which cells are empty at each row). In total, the possible availability matrices (and hence rectangles) to be examined as shown at Figure 4.2.9.

Notice that A_6 (i.e., A^*_6 without any '$*$') is not listed because R_{6-} is easily completable without a value appearing twice in the first two columns; in contrast, R^*_{6-} is pink (see Table 4.2.11 and Figure 4.2.11). Based on the selected entries of A_7 we observe that R_{7-} is not pink, whereas R^*_{7-} is (see Table 4.2.13 and Figure 4.2.11). There is no R^*_{t-} for $t = 8, 9, 10, 11$: observe that columns j_1 and j_2 are empty at both rows in R_{8-} and R_{11-} in Table 4.2.11 (thus no value is forbidden at some row because of its occurrence in the other row), while values 1 and 2 are missing from both rows in R_{9-} and R_{10-}. In Figure 4.2.10, the selected entries of A_9 show that R_{9-} is not pink. Thus this case includes the pink rectangles $R^*_{6-}, R^*_{7-}, R_{8-}, R_{10-}$ and R_{11-} and selection of values is depicted in Figure 4.2.11.



A$^*_6$: Row 1, Col 0

A$_7$: Row 1, Col 1

A$^*_7$: Row 1, Col 1

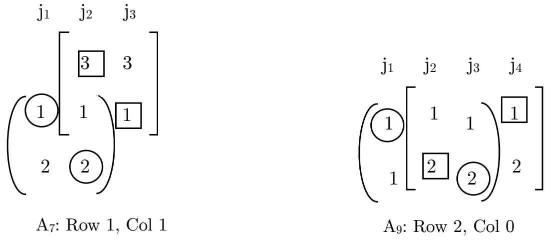
A$_8$: Row 1, Col 2

A$_9$: Row 2, Col 0

A$_{10}$: Row 2, Col 1

A$_{11}$: Row 2, Col 2

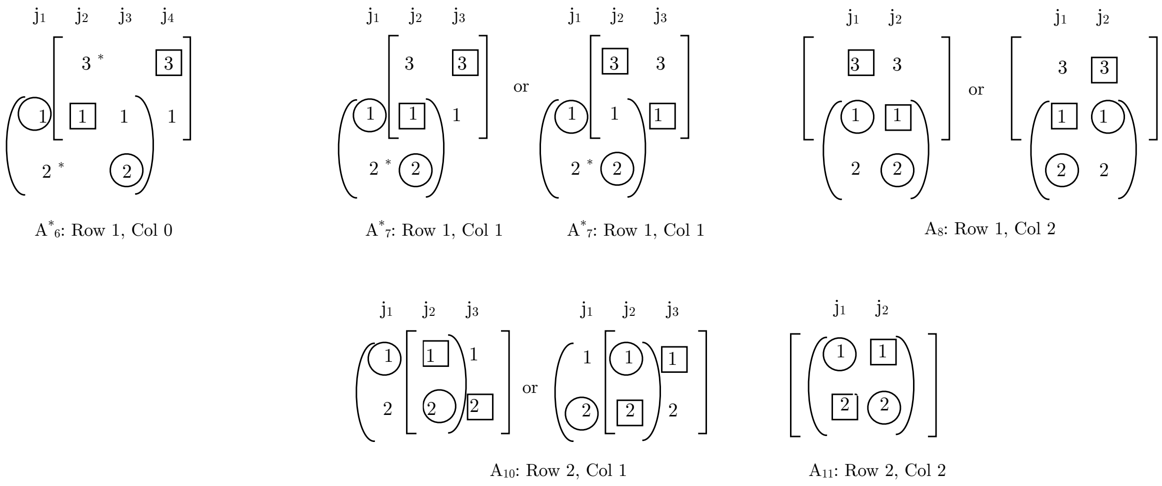
Figure 4.2.9: Availability matrices for Case 4



A$_7$: Row 1, Col 1

A$_9$: Row 2, Col 0

Figure 4.2.10: Completion of non-pink R_{7-}, R_{9-}



A$^*_6$: Row 1, Col 0

A$^*_7$: Row 1, Col 1

A$^*_7$: Row 1, Col 1

A$_8$: Row 1, Col 2

A$_{10}$: Row 2, Col 1

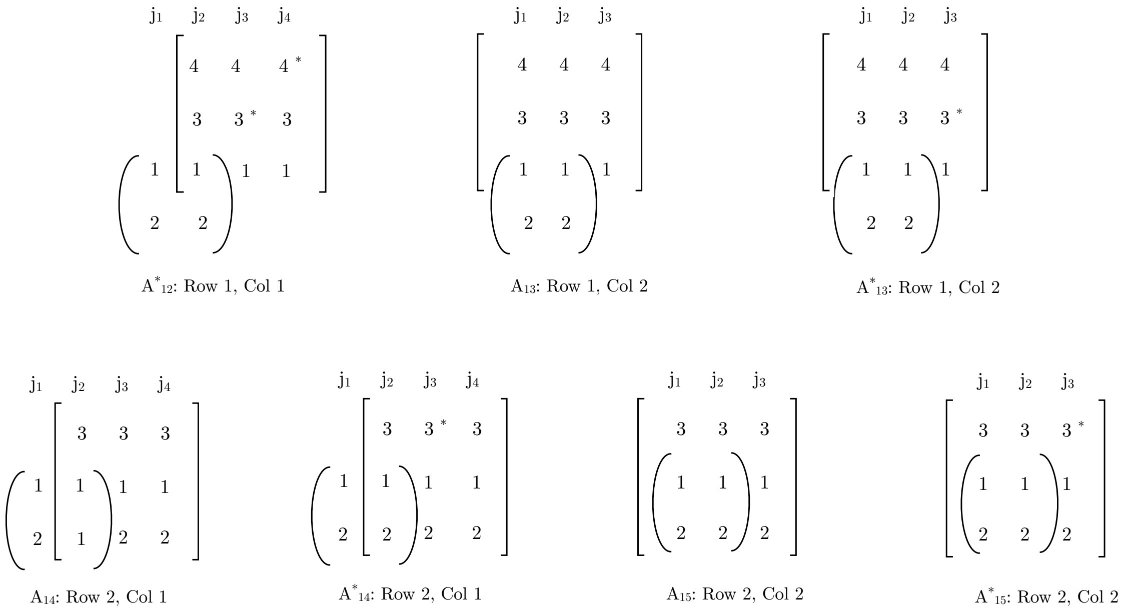
A$_{11}$: Row 2, Col 2

Figure 4.2.11: Completion of pink R^*_{6-}, R^*_{7-}, R_{8-}, R_{10-}, R_{11-}

Case 5. *Emptying* 1 *cell in row* 1 *and* 2 *cells in row* 2

Here, row 1 has 2 empty cells hence two missing values $\{1, 2\}$, whereas row 2 has 3 empty cells thus its missing values are either $\{1, 3, 4\}$ or $\{1, 2, 3\}$; hence there is a 2×2 availability matrix for row 1 and a 3×3 such matrix for row 2. These two matrices share 1 or 2 rows (depending on whether value 2 is missing from row 2) and 1 or 2 columns (depending on which cells are empty at each row); notice that should these matrices share 0 columns, any corresponding rectangle would not be pink.

The possible availability matrices (and hence rectangles) to be examined are shown in Figure 4.2.12, demonstrating that R^*_{12-} (and hence R_{12-}) is not pink, the same applying to R_{13-}, R^*_{13-} and to R_{14-}, R^*_{14-} (all presented in Table 4.2.13). Hence this case yields the pink rectangles R_{15-} and R^*_{15-} in Tables 4.2.11 and 4.2.12.



A^*_{12}: Row 1, Col 1

A_{13}: Row 1, Col 2

A^*_{13}: Row 1, Col 2

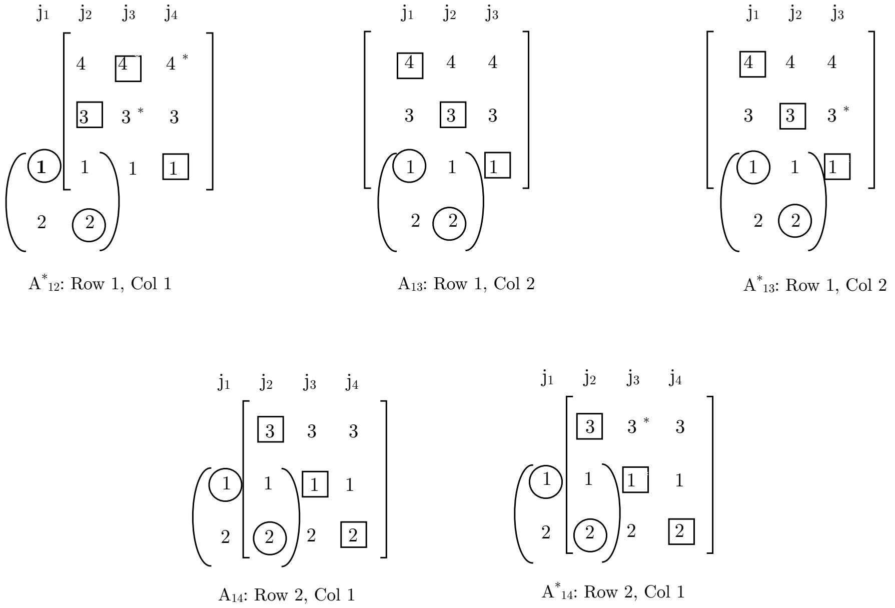
A_{14}: Row 2, Col 1

A^*_{14}: Row 2, Col 1

A_{15}: Row 2, Col 2

A^*_{15}: Row 2, Col 2

Figure 4.2.12: Availability matrices for Case 5



A*12: Row 1, Col 1

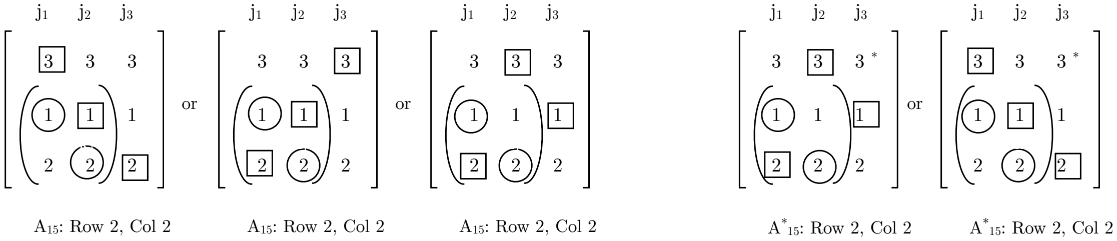
A13: Row 1, Col 2

A*13: Row 1, Col 2

A14: Row 2, Col 1

A*14: Row 2, Col 1

Figure 4.2.13: Completion of non-pink R^*_{12}, R_{13-}, R^*_{13-}, R_{14-}, R^*_{14-}



A15: Row 2, Col 2

A15: Row 2, Col 2

A15: Row 2, Col 2

A*15: Row 2, Col 2

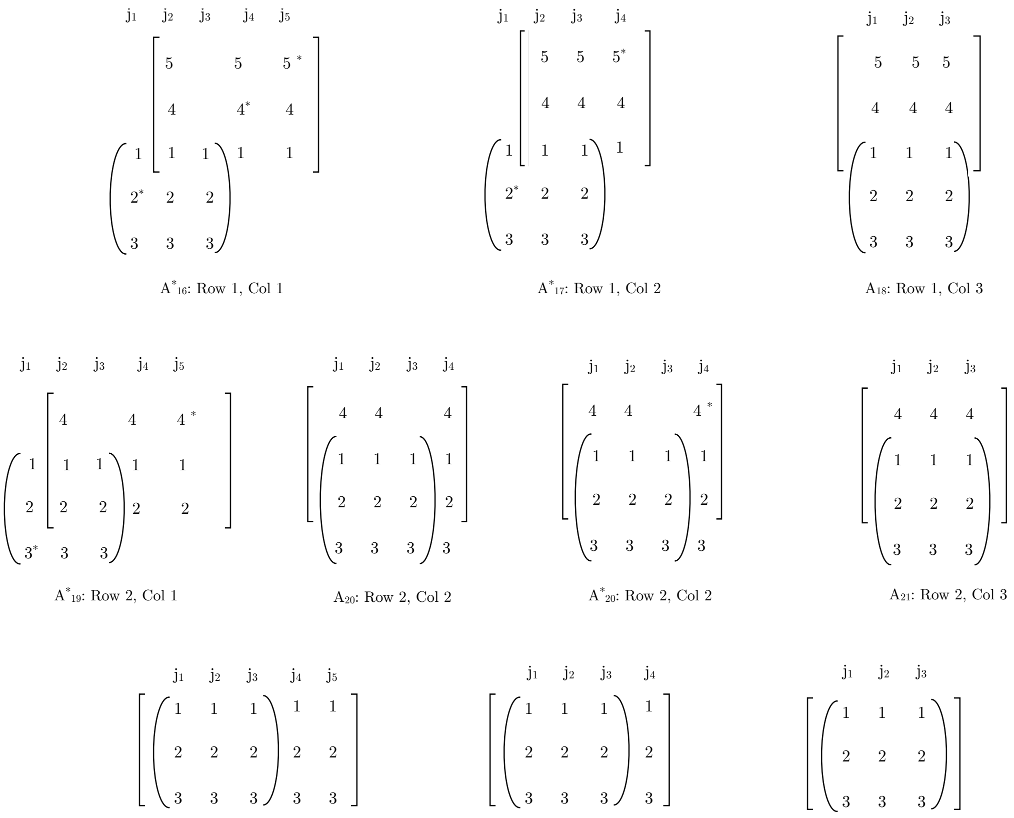
A*15: Row 2, Col 2

Figure 4.2.14: Completion of pink R_{15-}, R^*_{15-}

Case 6. *Emptying* 2 *cells in row* 1 *and* 2 *cells in row* 2

In this case, row 1 has 3 empty cells thus its missing values are $\{1, 2, 3\}$, whereas row 2 also has 3 empty cells but its missing values can be $\{1, 4, 5\}$ or $\{1, 3, 4\}$ or $\{1, 2, 3\}$; hence there is a 3×3 availability matrix for row 1 and the two matrices share 1 up to 3 rows and 0 up to 3 columns; notice that should these matrices share 0 columns, any corresponding rectangle would not be pink.

The possible availability matrices (and hence rectangles) to be examined are shown in Figure 4.2.15, with the selected entries showing that all corresponding rectangles except for R_{23-} and R_{24-} in Table 4.2.12, are non-pink. Notice also that non-applicability of an '$*$' in all matrices except for A_{18}, since columns j_1, j_2, j_3 are all empty in both rows regarding R_{18-}, R_{21-} and R_{24-} (thus no value is forbidden in some row because of its occurrence in the other row), while values $1, 2, 3$ are all missing from both rows regarding R_{22-} and R_{23-}.

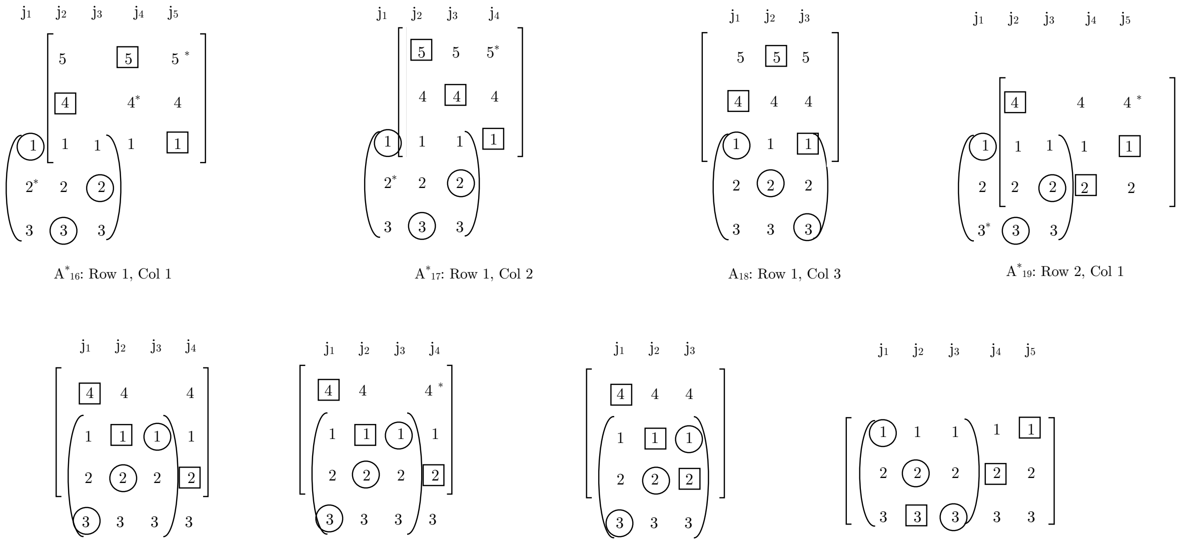


A^*_{16}: Row 1, Col 1 A^*_{17}: Row 1, Col 2 A_{18}: Row 1, Col 3

A^*_{19}: Row 2, Col 1 A_{20}: Row 2, Col 2 A^*_{20}: Row 2, Col 2 A_{21}: Row 2, Col 3

A_{22}: Row 3, Col 1 A_{23}: Row 3, Col 2 A_{24}: Row 3, Col 3

Figure 4.2.15: Availability matrices for Case 6



A^*_{16}: Row 1, Col 1 A^*_{17}: Row 1, Col 2 A_{18}: Row 1, Col 3 A^*_{19}: Row 2, Col 1

A_{20}: Row 2, Col 2 A^*_{20}: Row 2, Col 2 A_{21}: Row 2, Col 3 A_{22}: Row 3, Col 1

Figure 4.2.16: Completion of non-pink R^*_{16-}, R^*_{17-}, R_{18-}, R^*_{19-}, R_{20}, R^*_{20-}, R_{21-}, R_{22-}

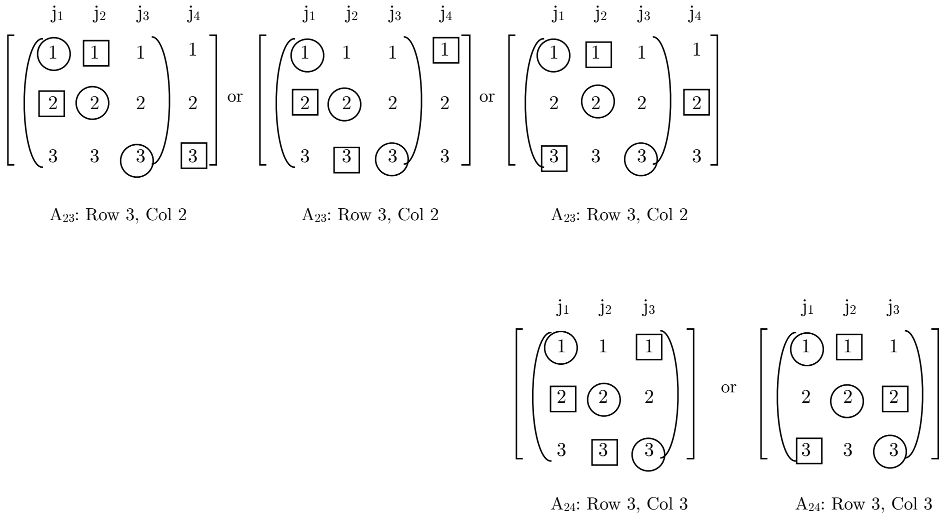


Figure 4.2.17: Completion of pink R_{23-}, R_{24-}

It is now clear that no further cases need to be examined for this class, hence all pink rectangles that share no elements with E are found and presented in Tables 4.2.11 and 4.2.12. □

Proposition 4.2.16. *The non-dominated pink rectangles, which share no element with E, are R^*_{6-}, R_{10-}, R_{11-}, R_{23-} and R_{24-}.*

Proof. In order to create pairs, as indicated in Step 2, we first need to categorise the availability matrices by type and then remove the dominated cases (see Definition 4.2.10). Up to this point, all combinations of emptied cells for a single rectangle have been found and from the completion patterns in Figures 4.2.4, 4.2.7, 4.2.11, 4.2.14 and 4.2.17 we derive the type of each availability matrix as described in Definition 4.2.8; these are presented in Table 4.2.14.

Type	Incomplete rectangle
I	$R^*_{1-},\ R_{2-},\ R^*_{6-},\ R_{10-}$
II	$R^*_{4-},\ R^*_{7-},\ R_{8-},\ R_{15-}, R^*_{15-},\ R_{23-},\ R_{24-}$
III	$R^*_{2-},\ R_{11-}$

Table 4.2.14: All pink rectangles that share no element with E

From set notation we can observe,

$$A_1^* \subset A_{10} \tag{4.2.2}$$

$$A_2 \subset A_{10} \tag{4.2.3}$$

$$A_6^* \subset A_{23} \tag{4.2.4}$$

$$A_4^* \subset A_{24} \tag{4.2.5}$$

$$A_7^* \subset A_{24} \tag{4.2.6}$$

$$A_8 \subset A_{24} \tag{4.2.7}$$

$$A_{10} \subset A_{24} \tag{4.2.8}$$

$$A_2^* \subset A_2 \subset A_{11} \subset A_{15}^* \subset A_{15} \subset A_{24} \tag{4.2.9}$$

Therefore we can eliminate dominated cases and Table 4.2.14 can be revised for each type of availability matrix to produce Table 4.2.15.

Type	Incomplete rectangles
I	$R_{6-}^*,\ R_{10-}$
II	$R_{23-},\ R_{24-}$
III	R_{11-}

Table 4.2.15: Non-dominated pink rectangles of Table 4.2.14

□

Theorem 4.2.17. *The incomplete pairs of rectangles of Tables 4.2.16 to 4.2.20 comprise, up to equivalence, the complete list of circuit members of $\mathcal{C}_{2,0}$.*

j_1	j_2	j_3	j_4	...	j_n
	3		$K^1\backslash\{1,2,3\}$		
2		n		$K^1\backslash\{1,2,3,n\}$	

R_{6-}^*

j_1	j_2	j_3	j_4	...	j_n
	3		$K^2\backslash\{1,2,3\}$		
2		n		$K^2\backslash\{1,2,3,n\}$	

R_{6-}^*

Table 4.2.16: Representative of $\mathcal{C}_{2,0}^1$

j_1	j_2	j_3	j_4	...	j_n
	3		$K^1\backslash\{1,2,3\}$		
2		n		$K^1\backslash\{1,2,3,n\}$	

R_{6-}^*

j_1	j_2	j_3	...	j_n
		$K^2\backslash\{1,2\}$		
n			$K^2\backslash\{1,2,n\}$	

R_{10-}

Table 4.2.17: Representative of $\mathcal{C}_{2,0}^2$

j_1	j_2	j_3	...	j_n
		$K^1\backslash\{1,2\}$		
n			$K^1\backslash\{1,2,n\}$	

R_{10-}

j_1	j_2	j_3	...	j_n
		$K^2\backslash\{1,2\}$		
n			$K^2\backslash\{1,2,n\}$	

R_{10-}

Table 4.2.18: Representative of $\mathcal{C}^3_{2,0}$

j_1	j_2	j_3	j_4	...	j_n
			$K^1\backslash\{1,2,3\}$		
		n		$K^1\backslash\{1,2,3,n\}$	

R_{23-}

j_1	j_2	...	j_n
		$K^2\backslash\{1,2\}$	
		$K^2\backslash\{1,2\}$	

R_{11-}

Table 4.2.19: Representative of $\mathcal{C}^4_{2,0}$

j_1	j_2	j_3	...	j_n
			$K^1\backslash\{1,2,3\}$	
			$K^1\backslash\{1,2,3\}$	

R_{24-}

j_1	j_2	j_3	...	j_n
		$K^2\backslash\{1,2\}$		
		$K^2\backslash\{1,2\}$		

R_{11-}

Table 4.2.20: Representative of $\mathcal{C}^5_{2,0}$

Proof. Up to this point we focused on a single rectangle and we now continue to create combinations of two pink rectangles. From Lemma 4.2.9 we know that rectangle R_- of type I can be matched with other rectangles R'_- of type I or II whereas rectangle R_- of type II, can only be matched with other R'_- of type III. Additionally, type III rectangles can also be matched with other of the same type. This is depicted in Table 4.2.21.

R_-	R'_-
type I	type I
type I	type III
type II	type III
type III	type III

Table 4.2.21: Permitted combinations of R and R'_-

From Tables 4.2.15 and Table 4.2.21 we derive the combinations $(R^*_{6-} \cup R^*_{6-})$, $(R^*_{6-} \cup R_{10-})$, $(R_{10-} \cup R_{10-})$, $(R_{23-} \cup R_{11-})$, $(R_{24-} \cup R_{11-})$, $(R^*_{6-} \cup R_{11-})$, $(R_{10-} \cup R_{11-})$ and $(R_{11-} \cup R_{11-})$. We continue to Step 3 to eliminate combinations we know are not minimal as per Remark 4.2.13. Hence, after taking into consideration (4.2.4), (4.2.8), (4.2.9) we conclude that the remaining combinations are, $(R^*_{6-} \cup R^*_{6-})$, $(R^*_{6-} \cup R_{10-})$, $(R_{10-} \cup R_{10-})$, $(R_{23-} \cup R_{11-})$, $(R_{24-} \cup R_{11-})$.

□

4.2.3 Class $\mathcal{C}_{2,1}$

We continue to find all members of $\mathcal{C}_{2,1}$ in a similar manner and now present proofs in a more concise manner.

Proposition 4.2.18. *Table 4.2.22 contains all non-dominated pink rectangles, which share one element with E.*

$$\begin{array}{|c|c|c|} \multicolumn{1}{c}{j_1} & \multicolumn{1}{c}{j_2} & \multicolumn{1}{c}{\cdots \quad j_n} \\ \hline & & K^1\backslash\{1,2\} \\ \hline & 1 & \\ \hline \multicolumn{3}{c}{R^*_{25-}} \end{array} \quad , \quad \begin{array}{c} A^*_{25} = \{(1,j_1,1),(1,j_1,2),(1,j_2,2), \\ (2,j_1,2),(2,j_1,3),...,(2,j_1,n), \\ (2,j_3,2),...,(2,j_3,n),...,(2,j_n,2),...,(2,j_n,n)\} \end{array}$$

$$\begin{array}{|c|c|c|} \multicolumn{1}{c}{j_1} & \multicolumn{1}{c}{j_2} & \multicolumn{1}{c}{\cdots \quad j_n} \\ \hline & & K^1\backslash\{1,2\} \\ \hline 2 & 1 & \\ \hline \multicolumn{3}{c}{R^*_{26-}} \end{array} \quad , \quad \begin{array}{c} A^*_{26} = \{(1,j_1,1),(1,j_2,2),(2,j_3,3),...,(2,j_3,n),..., \\ (2,j_n,3),...,(2,j_n,n)\} \end{array}$$

$$\begin{array}{|c|c|c|c|} \multicolumn{1}{c}{j_1} & \multicolumn{1}{c}{j_2} & \multicolumn{1}{c}{j_3} & \multicolumn{1}{c}{\cdots \quad j_n} \\ \hline & & & K^1\backslash\{1,2,3\} \\ \hline 2 & 1 & \multicolumn{2}{c|}{} \\ \hline \multicolumn{4}{c}{R^*_{27-}} \end{array} \quad , \quad \begin{array}{c} A^*_{27} = \{(1,j_1,1),(1,j_2,2),(1,j_3,1),(1,j_3,2),(1,j_3,3) \\ (2,j_3,3),...,(2,j_3,n),...,(2,j_n,3),...,(2,j_n,n)\} \end{array}$$

Table 4.2.22: Pink rectangles that share one element with E

Proof. By definition, a pair of rectangles $R_- \cup R'_-$ is a circuit of $\mathcal{C}_{2,1}$ only if $\left|\left(R_- \cup R'_-\right) \cap E\right| = 1$, thus assume without loss of generality that $|R_- \cap E| = 0$ and $\left|R'_- \cap E\right| = 1$. All possible availability matrices corresponding to pink R_- such that $|R_- \cap E| = 0$ were found in the previous section and are presented in Table 4.2.15. For R'_-, let us assume without loss of generality, that the element shared with E is $(2, j_2, 1)$. Our goal is to enforce a repetition of a pair in the first two columns, therefore since value 1 already appears in cell $(2, j_2)$ we can start by emptying all other cells in row 2.

Case 7. *Emptying* 1 *cell in row* 1 *and* $n-1$ *cells in row* 2

By definition cell $(1, j_1)$ is emptied in the first row resulting in an incomplete rectangle of type I, presented next in case 9. This rectangle is clearly dominated by the pink rectangle R^*_{26-} (type I).

Case 8. *Emptying* 2 *cells in row* 1 *and* $n-1$ *cells in row* 2

In the second row, all cells but $(2, j_2)$ are emptied and in the first row we can assume that the cells emptied are $(1, j_1)$ and $(1, j_2)$ with missing values $\{1, 2\}$. This gives rise to R^*_{25-} which is pink (type I), since value 1 is forbidden for the cell $(1, j_2)$ hence appearing with a '$*$' in the availability matrix of row 1, leaving value 1 as the only option for cell $(1, j_1)$. Emptying any cell other than $(1, j_2)$, e.g. $(1, j_3)$ will give rise to a non-pink rectangle, since value 1 can then be placed in cell $(1, j_3)$ and value 2 in cell $(1, j_1)$. Emptying additional cells in row 1 will result in a similar non-pink structure, unless an additional cell is filled in row 2 to enforce the selection of value 1 in cell $(1, j_1)$, hence Case 10.

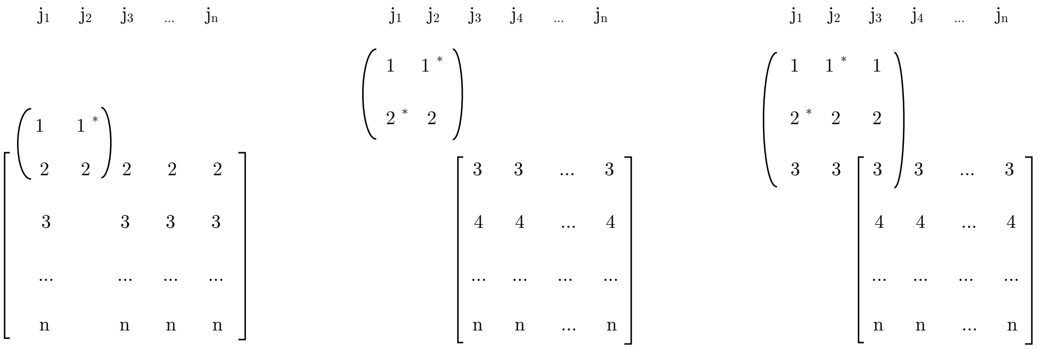
Case 9. *Emptying* 2 *cells in row* 1 *and* $n-2$ *cells in row* 2

In the second row, all cells but $(2, j_1)$ and $(2, j_2)$ can be emptied and in the first row we can assume that the cells emptied are $(1, j_1)$ and $(1, j_2)$ with missing values $\{1, 2\}$. Emptying cells $(1, j_1)$ and $(1, j_2)$ gives rise to R^*_{26-} which is pink (type II) since values 1 and 2 are forced to appear in cells $(1, j_1)$ and $(1, j_2)$ due to the '$*$' appearing in the availability matrices, indicating that cells $(2, j_1)$ and $(2, j_2)$ contain values 2 and 1 respectively.

Case 10. *Emptying* 3 *cells in row* 1 *and* $n-2$ *cells in row* 2

Here, emptied cells in the second row remain as per previous case, while in row 1 cells $(1, j_1)$, $(1, j_2)$ and $(1, j_3)$ are emptied with missing values $\{1, 2, 3\}$. This gives rise to R^*_{27-} which is pink (type I) which is completable either by placing value 1 in $(1, j_1)$ or value 2 in $(1, j_3)$ or both. Notice that emptying any other cell from row 1 leads to a non-pink rectangle.

Incomplete rectanlges are shown in Table 4.2.22, while their availability matrices are shown in Figure 4.2.18. Notice that although $A^*_{25} \subset A^*_{26}$ and $A^*_{27} \subset A^*_{26}$, hence by Remark 4.2.12 $R^*_{26-} \subset R^*_{25-}$ and $R^*_{26-} \subset R^*_{27-}$, R^*_{26-} is not dominated (recall Definition 4.2.10) since being of type III, whereas R^*_{25-} is of type I and R^*_{27-} is of type II.



A$^*_{25}$: Row 1, Col 1 A$^*_{26}$: Row 0, Col 0 A$^*_{27}$: Row 1, Col 1

Figure 4.2.18: Availability matrices corresponding to pink R^*_{25-}, R^*_{26-}, R^*_{27-}

Incomplete rectangles derived from the cases above are presented by type in Table 4.2.23. A detailed representation is shown in Table 4.2.22. However, not all fifteen combinations given by the five in Table4.2.15 and three in Table 4.2.23 give rise to minimal incompletable *pairs* of rectangles, as explained next in the proof of Theorem 4.2.19.

Type	Incomplete rectangles
I	R^*_{25-}
II	R^*_{27-}
III	R^*_{26-}

Table 4.2.23: Non-dominated pink rectangles that share one element with E

□

Theorem 4.2.19. *The incomplete pairs of rectangles of Tables 4.2.24 to 4.2.29 comprise, up to equivalence, the complete list of circuit members of $\mathcal{C}_{2,1}$.*

j_1	j_2	j_3	j_4	... j_n
	3		$K^1\backslash\{1,2,3\}$	
2		n		$K^1\backslash\{1,2,3,n\}$

R^*_{6-}

j_1	j_2	j_3 ... j_n
		$K^2\backslash\{1,2\}$
	1	

R^*_{25-}

Table 4.2.24: Representative of $\mathcal{C}^1_{2,1}$

j_1	j_2	j_3	... j_n
		$K^1\backslash\{1,2\}$	
n			$K^1\backslash\{1,2,n\}$

R_{10-}

j_1	j_2	j_3 ... j_n
		$K^2\backslash\{1,2\}$
	1	

R^*_{25-}

Table 4.2.25: Representative of $\mathcal{C}^2_{2,1}$

j_1	j_2	j_3 ... j_n
		$K^1\backslash\{1,2\}$
		$K^1\backslash\{1,2\}$

R_{11-}

j_1	j_2	j_3 ... j_n
		$K^2\backslash\{1,2\}$
	1	

R^*_{25-}

Table 4.2.26: Representative of $\mathcal{C}^3_{2,1}$

j_1	j_2	j_3	j_4	... j_n
			$K^1\backslash\{1,2,3\}$	
		n		$K^1\backslash\{1,2,3,n\}$

R_{23-}

j_1	j_2	j_3 ... j_n
		$K^2\backslash\{1,2\}$
2	1	

R^*_{26-}

Table 4.2.27: Representative of $\mathcal{C}^4_{2,1}$

j_1	j_2	j_3	...	j_n
			$K^1\backslash\{1,2,3\}$	
			$K^1\backslash\{1,2,3\}$	

R_{24-}

j_1	j_2	j_3	...	j_n
		$K^2\backslash\{1,2\}$		
2	1			

R^*_{26-}

Table 4.2.28: Representative of $\mathcal{C}^5_{2,1}$

j_1	j_2	j_3	...	j_n
		$K^1\backslash\{1,2\}$		
		$K^1\backslash\{1,2\}$		

R_{11-}

j_1	j_2	j_3	...	j_n
			$K^2\backslash\{1,2,3\}$	
2	1			

R^*_{27-}

Table 4.2.29: Representative of $\mathcal{C}^6_{2,1}$

Proof. Recall from Table 4.2.15 that R^*_{6-} and R_{10-} are of type I, R_{23-} and R_{24-} are of type II and R_{11-} is of type III. By Lemma 4.2.9, R^*_{6-} or R_{10-} can be paired with R^*_{25-} (since all are of type I), hence the circuit of Tables 4.2.24 and 4.2.25.

All five rectangles of Proposition 4.2.16 can be paired with R^*_{26-} since the latter is of type III. Notice, however, that $A^*_{26} \subset A^*_{25}$ hence the pairs $R^*_{6-} \cup R^*_{25-}$, $R_{10-} \cup R^*_{25-}$ and $R_{11-} \cup R^*_{25-}$ dominate, respectively, the pairs $R^*_{6-} \cup R^*_{26-}$, $R_{10-} \cup R^*_{26-}$ and $R_{11-} \cup R^*_{26-}$, which are therefore omitted. No other rectangles are dominated and for the remaining, we know that they are pink and all possible cells have been emptied. Therefore it follows that the pairs containing R^*_{26-} are $R_{23-} \cup R^*_{26-}$ and $R_{24-} \cup R^*_{26-}$ and satisfy both the exclusion and minimality property. These are the circuit of Table 4.2.26 and the circuit of Table 4.2.27.

Last, R_{11-} (being of type III) can be paired with all rectangles of Proposition 4.2.18; as above, $A^*_{26} \subset A^*_{25-}$ yields that the $R_{11-} \cup R^*_{26-}$ is not a circuit, hence the remaining two pairs containing R_{11-} are depicted in Tables 4.2.28 and 4.2.29.

In summary we have combinations $(R^*_{6-} \cup R^*_{25-})$, $(R_{10-} \cup R^*_{25-})$, $(R_{11-} \cup R^*_{25-})$, $(R_{23-} \cup R^*_{26-})$, $(R_{24-} \cup R^*_{26-})$, $(R_{11-} \cup R^*_{27-})$ as the six sub-cases of circuits in $\mathcal{C}_{2,1}$.

□

4.2.4 Class $\mathcal{C}_{2,2}$

We now continue to find all circuits of Class 2.

Proposition 4.2.20. *Table 4.2.30 contains all non-dominated pink rectangles, which share two element with E.*

$$\begin{array}{|c|c|c|} \multicolumn{1}{c}{j_1} & \multicolumn{1}{c}{j_2} & \multicolumn{1}{c}{\quad ... \quad j_n} \\ \hline 1 & & \qquad\qquad \\ \hline & 1 & \\ \hline \multicolumn{3}{c}{R_{28-}} \end{array} \quad , \quad \begin{array}{c} A_{28} = \{(1,j_2,2),...,(1,j_2,n),...,(1,j_n,2),...,(1,j_n,n) \\ (2,j_1,2),...,(2,j_1,n),(2,j_3,2),...,(2,j_3,n)..., \\ (2,j_n,3),...,(2,j_n,n)\} \end{array}$$

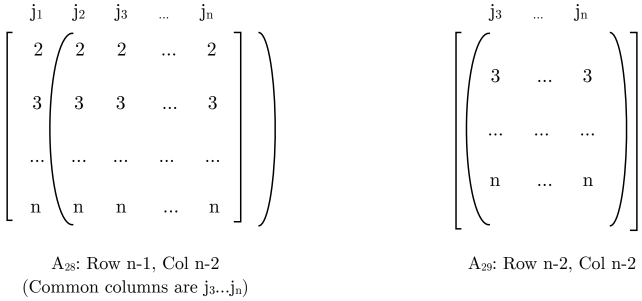
$$\begin{array}{|c|c|c|} \multicolumn{1}{c}{j_1} & \multicolumn{1}{c}{j_2} & \multicolumn{1}{c}{\quad ... \quad j_n} \\ \hline 1 & 2 & \qquad\qquad \\ \hline 2 & 1 & \\ \hline \multicolumn{3}{c}{R_{29-}} \end{array} \quad , \quad \begin{array}{c} A_{29} = \{(1,j_3,3),...,(1,j_3,n),...,(1,j_n,3),...,(1,j_n,n), \\ (2,j_3,3),...,(2,j_3,n),...,(2,j_n,3),...,(2,j_n,n)\} \end{array}$$

Table 4.2.30: Pink rectangles that share two elements with E

Proof. This proof is similar to Proposition 4.2.18. The two elements shared with E are $(1, j_1, 1)$ and $(2, j_2, 1)$; keeping only these two elements results in the first availability matrix of Figure 4.2.19, corresponding to R_{28-} which is of type I and clearly dominates any other incomplete rectangle of the same type that shares two elements with E but has fewer emptied cells. However, we may also include elements $(1, j_2, 2)$and $(2, j_1, 2)$, thus yielding the second matrix of Figure 4.2.19 that corresponds to R_{29-}. Although $A_{29} \subset A_{28}$, R_{29-} is not dominated since it is of type III, whereas R_{28-} is of type I; this is presented in Table 4.2.31.

Type	Incomplete rectangles
Type I	R_{28-}
Type III	R_{29-}

Table 4.2.31: Non-dominated pink rectangles that share two elements with E



A28: Row n-1, Col n-2
(Common columns are j3...jn)

A29: Row n-2, Col n-2

Figure 4.2.19: Availability matrices corresponding to R_{28-}, R_{29-}

□

Theorem 4.2.21. *The incomplete pairs of rectangles of Tables 4.2.32 to 4.2.38 comprise, up to equivalence, the complete list of circuit members of $\mathcal{C}_{2,2}$.*

j_1	j_2	j_3	j_4	...	j_n
	3		$K^1\backslash\{1,2,3\}$		
2		n		$K^1\backslash\{1,2,3,n\}$	

R^*_{6-}

j_1	j_2	... j_n
1		
	1	

R_{28-}

Table 4.2.32: Representative of $\mathcal{C}^1_{2,2}$

j_1	j_2	j_3	... j_n
		$K^1\backslash\{1,2\}$	
n			$K^1\backslash\{1,2,n\}$

R_{10-}

j_1	j_2	... j_n
1		
	1	

R_{28-}

Table 4.2.33: Representative of $\mathcal{C}^2_{2,2}$

j_1	j_2	... j_n
		$K\backslash\{1,2\}$
		$K\backslash\{1,2\}$

R_{11-}

j_1	j_2	... j_n
1		
	1	

R_{28-}

Table 4.2.34: Representative of $\mathcal{C}^3_{2,2}$

j_1	j_2	j_3	j_4	...	j_n
			$K^1\backslash\{1,2,3\}$		
		n		$K^1\backslash\{1,2,3,n\}$	

R_{23-}

j_1	j_2	j_3 ... j_n
1	2	
2	1	

R_{29-}

Table 4.2.35: Representative of $\mathcal{C}^4_{2,2}$

j_1	j_2	j_3	... j_n
			$K^1\backslash\{1,2,3\}$
			$K^1\backslash\{1,2,3\}$

R_{24-}

j_1	j_2	j_3 ... j_n
1	2	
2	1	

R_{29-}

Table 4.2.36: Representative of $\mathcal{C}^5_{2,2}$

j_1	j_2	j_3 ... j_n
		$K^1\backslash\{1,2\}$
	1	

R^*_{25-}

j_1	j_2	j_3 ... j_n
		$K^2\backslash\{1,2\}$
	1	

R^*_{25-}

Table 4.2.37: Representative of $\mathcal{C}^6_{2,2}$

j_1	j_2	j_3	···	j_n
		$K^1\backslash\{1,2\}$		
2	1			

R^*_{26-}

j_1	j_2	j_3	···	j_n
			$K^2\backslash\{1,2,3\}$	
2	1			

R^*_{27-}

Table 4.2.38: Representative of $\mathcal{C}^7_{2,2}$

Proof. By definition, a pair of rectangles $R_- \cup R'_-$ is a circuit of $\mathcal{C}_{2,2}$ only if $\left|\left(R_- \cup R'_-\right) \cap E\right| = 2$, thus we may consider that either $|R_- \cap E| = 0$ and $\left|R'_- \cap E\right| = 2$ or $|R_- \cap E| = \left|R'_- \cap E\right| = 1$.

For the former case ($|R_- \cap E| = 0$, $\left|R'_- \cap E\right| = 2$) we restrict ourselves to the five rectangles listed in Table 4.2.15 (that can play the role of R_-) and for R'_- we illustrate availability matrices in Table 4.2.19.

By Lemma 4.2.9, R^*_{6-} or R_{10-} can be paired with R_{28-} (since all are of type I), hence the first two circuits of Tables 4.2.32 and 4.2.33. By the same Lemma, R_{11-} can be paired with R_{28-} since the former is of type III, hence the circuit of Table 4.2.34. All five rectangles of Table 4.2.15 can be paired with R_{29-} since the latter is of type III; however, $A_{29-} \subset A_{28-}$ hence the pairs $R^*_{6-} \cup R_{29-}$, $R_{10-} \cup R_{29-}$ and $R_{11-} \cup R_{29-}$ are dominated by rectangles in Tables 4.2.32, 4.2.33 and 4.2.34. The only remaining pairs containing R_{29-} are $R_{23-} \cup R_{29-}$ and $R_{24-} \cup R_{29-}$, i.e., the circuits of Tables 4.2.35 and 4.2.36.

For the latter case ($|R_- \cap E| = \left|R'_- \cap E\right| = 1$), it suffices to examine the three rectangles of Proposition 4.2.18. By Lemma 4.2.9, R^*_{25-} (of type I) can be paired with itself and with R^*_{26-} (of type III); however, $(A^*_{25} \cup A^*_{26}) \subset (A^*_{25} \cup A^*_{25})$ since $A^*_{26} \subset A^*_{25}$; for the same reason $A^*_{26} \cup A^*_{26}$ (plausible by Lemma 4.2.9 since R^*_{26-} is of type III) is omitted since it is a subset of $A^*_{25} \cup A^*_{25-}$. Last, R^*_{26-} can be paired with R^*_{27-}. Overall, the case of $|R_- \cap E| = \left|R'_- \cap E\right| = 1$ leads to the circuits of Tables 4.2.37 and 4.2.38.

Therefore in summary, according to Table 4.2.21 we conclude with the following combinations: $(R^*_{6-} \cup R_{28-})$, $(R_{10-} \cup R_{28-})$, $(R_{11-} \cup R_{28-})$, $(R_{23-} \cup R_{29-})$, $(R_{24-} \cup R_{29-})$, $(R^*_{25-} \cup R^*_{25-})$ and $(R^*_{26-} \cup R^*_{27-})$. □

4.2.5 Class $\mathcal{C}_{2,3}$

Theorem 4.2.22. *The rectangles of Tables 4.2.39 and 4.2.40 comprise, up to equivalence, the complete list of circuit members of $\mathcal{C}_{2,3}$.*

j_1	j_2	j_3	···	j_n
		$K^1\backslash\{1,2\}$		
	1			

R^*_{25-}

j_1	j_2	j_3	···	j_n
1				
	1			

R_{28-}

Table 4.2.39: Representative of $\mathcal{C}^1_{2,3}$

j_1	j_2	j_3	...	j_n
			$K^1\backslash\{1,2,3\}$	
2	1			

R^*_{27-}

j_1	j_2	j_3	...	j_n
1	2			
2	1			

R_{29-}

Table 4.2.40: Representative of $\mathcal{C}^2_{2,3}$

Proof. By definition, a pair of rectangles $R_- \cup R'_-$ is a circuit of $\mathcal{C}_{2,3}$ only if $\left|\left(R_- \cup R'_-\right) \cap E\right| = 3$, thus we may consider without loss of generality that $|R_- \cap E| = 1$ and $\left|R'_- \cap E\right| = 2$. That is, we may restrict ourselves to the three rectangles of Proposition 4.2.18 and the two rectangles of Proposition 4.2.20 for the roles of R_- and R'_- respectively.

By Lemma 4.2.9, R^*_{25-} can be paired with R_{28-} (since both are of type I), hence the first circuit of Table 4.2.39. By the same Lemma, all three rectangles of Proposition 4.2.18 can be paired with R_{29-} since the latter is of type III; observe, however, $A_{29} \subset A_{28}$ yields $(A^*_{25} \cup A_{29}) \subset (A^*_{25} \cup A_{28})$ and $A^*_{26} \subset A^*_{27}$ yields $(A^*_{26} \cup A_{29}) \subset (A^*_{27} \cup A_{29})$, the only non-omitted such pair is the second circuit in Table 4.2.40. Last, although R^*_{26-} (since of type III) can also be paired with R_{28-}, $R^*_{26-} \cup R_{28-}$ includes $R^*_{25-} \cup R_{28-}$ since $A^*_{26} \subset A^*_{25}$.

□

Theorem 4.2.23. *Up to equivalence, the incomplete pairs of rectangles of Tables 4.2.16 to 4.2.20, Tables 4.2.24 to 4.2.29, Tables 4.2.32 to 4.2.38, Tables 4.2.39 to 4.2.40 and 4.2.1 comprise the complete list of circuit members of* $\mathcal{C}_2$ *.*

Proof. This directly follows from Theorems 4.2.4, 4.2.17, 4.2.19, 4.2.21 and 4.2.22. □

4.3 Global circuits

In this section we will show that members of $\mathcal{C}_{1,1}$, $\mathcal{C}_{1,2}$, $\mathcal{C}_{1,3}$, $\mathcal{C}_{1,5}$ and $\mathcal{C}_{2,4}$ are *global circuits*. Remember that the first four are circuits of the 2-row Latin rectangle independence system and they are all five circuit of the OLR2 independence system. The term global circuits means that they are also subsets of the clutter of circuits associated to any set of $|T|$ MOLRm of order n, where $|T| \leq n-1$. Clearly this holds only if that particular n, the set of $|T|$ MOLRm exists. For example an OLS of order 6 does not exist, therefore its clutter of bases and clutter of circuits is empty.

We first start with a note on the exclusion property. Let G be the ground set of $|T|$ MOLRm, where $|T| < n-1$ and IS is its independence system. If a set $C \subset G$ satisfies the exclusion property for IS, then it also satisfies the exclusion property for IS', where IS' is the independence system of a set of $|T'|$ MOLRp with $p > m$ and $|T| < |T'| \leq n-1$. This becomes apparent if we consider an incomplete 2-row rectangle that is incompletable to a 2-row Latin rectangle; it clearly cannot be completed to an to an OLS.

Something similar does not hold for the minimality property. If C satisfies the minimality property then for any $c \in C,\ \ C \backslash \{c\}$ is completable to a set of $|T|$ MOLR. However, it is not necessarily completable to a set of $|T'|$ MOLRp.

Proposition 4.3.1. *Let IS^{MOLR} be the independence system of a set of $|T|$ MOLRm of order n, where $m,\ |T| > 2$. Then $\mathcal{C}_{1,1}$, $\mathcal{C}_{1,2}$, $\mathcal{C}_{1,3}$, $\mathcal{C}_{1,5}$ and $\mathcal{C}_{2,4}$ are circuits for IS^{MOLR}.*

Proof. Every member of $\mathcal{C}_{1,1}$, $\mathcal{C}_{1,2}$, $\mathcal{C}_{1,3}$, $\mathcal{C}_{1,5}$ and $\mathcal{C}_{2,4}$ satisfies the exclusion property for IS^{MOLR}. What remains to show is that they also satisfy the minimality property. We start with some $C \in \mathcal{C}_{1,1}$, and with no loss of generality this can be the representative of Table 4.1.1. Let B^{MOLR} be the clutter of basis associated to IS^{MOLR}. We sill show that up to equivalence, $C \backslash \{c\} \subset B$, for every $B \in B^{MOLR}$. Let $\{c\} = \{(2, j_1, 1)\}$, then $C \backslash \{c\} = \{(1, j_1, 1)\}$ and we know that we can permute elements of the first rectangle comprising $B \in B^{MOLS}$, such that value 1 appears in cell $(1, j_1)$. A similar argument holds for $\mathcal{C}_{1,2}$, $\mathcal{C}_{1,3}$, $\mathcal{C}_{1,5}$ and $\mathcal{C}_{2,4}$. □

4.4 Concluding remarks

The main focus of this chapter has been to fully characterise the clutter of circuits $\mathcal{C}_2 \backslash \mathcal{C}_1$. With use of the availability matrix and a structured enumerative approach its's been shown that $\mathcal{C}_2 \backslash \mathcal{C}_1$ can be organised into five distinct classes containing a total of twenty-one non-equivalent circuits. In the set theoretical context that was established in Chapter 2, these circuits are presented as sets. They are also presented as arrays and in these illustrations it is easy to see that the exclusion and minimality properties are met.

It's been established that an incomplete pair of rectangles is completable to an OLR2 if and only if it contains no circuit. Therefore to answer the completability question, the next step is to construct a polynomial-time recognition algorithm in order to verify the existence or non-existence of a circuit member of $\mathcal{C}_2$ in any incomplete pair. Such an algorithm is not presented in this Thesis but is suggested as a topic for further research.

The next chapter continues to introduce a 3-index formulation for the OLR2 problem and formulates all circuit members of $\mathcal{C}_2 \backslash \mathcal{C}_1$ as lifted circuit inequalities.

Chapter 5

Formulations and lifted circuit inequalities for MOLR2

The results of the previous chapter address the question of whether a given pair of (incomplete) 2-row rectangles is completable to an OLR2 and to reach an answer it suffices to examine the existence of all non-equivalent circuits of $\mathcal{C}_2$. However, knowledge of $\mathcal{C}_2$ can also be interesting in terms of optimisation, i.e., to design an Integer Program (IP) for finding whether a specific pair of incomplete rectangles is completable to an OLR. The first to approach the OLS problem as an IP was Gale (in [19]), where the OLS is formulated as a 4-index planar assignment problem. Appa et al. in [4], [6] based on this formulation continue to establish the dimension of the OLS polytope and derive facet-defining and other valid inequalities for this polytope. Clearly in this formulation the number of rows can be reduced to address the OLR2 problem.

Section 5.1 presents an alternative formulation for the OLR2 case. This is a 3-index formulation shown for the first time and is attributed to R. Euler, that essentially considers two IP formulations, one for each rectangle comprising the OLR2 and links the two with what are referred to as orthogonality constraints and are derived from the representative of $\mathcal{C}_{2,4}$ as shown in Chapter 4. This ensures that the solution will produce a pair of rectangles in which no pair of values is repeated. Related theoretical results on formulating circuits of $\mathcal{C}_2 \backslash \mathcal{C}_1$ as circuit inequalities are presented in Sections 5.2 and all such inequalities are lifted and listed in 5.3.

5.1 A 3-index Integer Programming formulation for MORL2

Consider the four disjoint sets $I = \{1, ..., m\}$, $J = \{1, ..., n\}$, $K^1 = \{k_1^1, ..., k_n^1\}$ and $K^2 = \{k_1^1, ..., k_n^2\}$. Suppose that I and J are the row and column sets respectively, while K^1 and K^2 are the set of values in R_1 and R_2 respectively. Let x_{ijk^1} be a binary variable that takes value 1 if k^1 appears in cell (i, j) of R_1 and takes value 0 otherwise. Variable y_{ijk^2} is defined similarly for R_2.

For two rectangles each having m rows, a solution to the following IP formulation can have at most $2mn$

variables take value 1, in which case it produces a pair of m-row orthogonal Latin rectangles of order n. For solutions with less than $2mn$ variables, the IP provides a pair of incomplete m-row rectangles (with no explicit violation) proving that a complete structure does not exist. In a similar manner, when we start with an incomplete rectangle as an initial solution (i.e. pre-setting one variable per non-empty cell to 1), the IP will show whether it can be completed or not. In this case, if the incomplete rectangles explicitly violate the Latin rectangle and/or orthogonality definitions, then the IP will be infeasible due to the violation of the corresponding constraints.

IP formulation:

$$Max \sum \{x_{ijk^1} : i \in I, \;\; j \in J, \;\; k^1 \in K^1\} + \sum \{y_{ijk^2} : i \in I, \;\; j \in J, \;\; k^2 \in K^2\}$$

Subject to,

Latin rectangle constraints:

$$\sum\{x_{ijk^1} : i \in I\} \leq 1, \qquad \forall\, j \in J, \quad k^1 \in K^1, \tag{5.1.1}$$
$$\sum\{x_{ijk^1} : j \in J\} \leq 1, \qquad \forall\, i \in I, \quad k^1 \in K^1, \tag{5.1.2}$$
$$\sum\{x_{ijk^1} : k^1 \in K^1\} \leq 1, \qquad \forall\, i \in I, \quad j \in J, \tag{5.1.3}$$

$$\sum\{y_{ijk^2} : i \in I\} \leq 1, \qquad \forall\, j \in J, \quad k^2 \in K^2, \tag{5.1.4}$$
$$\sum\{y_{ijk^2} : j \in J\} \leq 1, \qquad \forall\, i \in I, \quad k^2 \in K^2, \tag{5.1.5}$$
$$\sum\{y_{ijk^2} : k^2 \in K^2\} \leq 1, \qquad \forall\, i \in I, \quad j \in J, \tag{5.1.6}$$

Orthogonality constraints:

$$\begin{gathered} x_{i_1 j_1 k^1} + x_{i_2 j_2 k^1} + y_{i_1 j_1 k^2} + y_{i_2 j_2 k^2} \leq 3, \\ \forall\, \{i_1, i_2 \,:\, i_2 > i_1\} \subseteq I, \quad \{j_1, j_2\} \subset J, \quad k^1 \in K^1, \quad k^2 \in K^2, \end{gathered} \tag{5.1.7}$$

$$x_{ijk^1} \in \{0,1\}, \qquad \forall\, i \in I, \quad j \in J, \quad k^1 \in K^1,$$
$$y_{ijk^2} \in \{0,1\}, \qquad \forall\, i \in I, \quad j \in J, \quad k^2 \in K^2$$

The first six sets of constraints ensure that the produced rectangles R_1 and R_2 are Latin. More specifically, constraints (5.1.1) and (5.1.4) ensure that no value is repeated in each column of R_1 and R_2 respectively, constraints (5.1.2) and (5.1.5) ensure that no value is repeated in each row of R_1 and R_2 respectively and constraints (5.1.3) and (5.1.6) ensure that every cell of the rectangles does not contain more than one value. Finally, constraints (5.1.7) ensure that R_1 and R_2 are orthogonal, i.e. forbid a pair of values to appear twice. This is accomplished by introducing a constraint for each possible combination of four variables that represent a repetition of a pair, and allowing at most three out of the four variables to take value 1. Therefore, integer vectors (x, y) that are feasible with respect to (5.1.1)-(5.1.7) are in $1 - 1$ correspondence with pairs of rectangles (possibly incomplete), each not violating the Latin rectangle and orthogonality definitions.

For $m = n$ where we have an OLS, if such a structure exists then each value k^1 must appear exactly once in each row i and column j of R_1 and similarly for k^2 in R_2. Therefore inequalities (5.1.1)-(5.1.6) can be written as equality constraints and the objective function becomes arbitrary. We now have a feasibility problem in which case a feasible solution containing $2n^2$ variables proves that an OLS of order n exists whereas an infeasible solution proves that it doesn't. Similarly for incomplete pairs, if they are completable the solution will be feasible and infeasible otherwise.

We continue to examine the OLR2 case ($m = 2$) in more detail. For the OLR2 observe that the formulation has $4n^2$ variables. The number of variables appearing in each constraint type as well as the number of constraints of each type are shown in the Table 5.1.1

Type	**Vars/Cons**	**No. of Cons**
(5.1.1)	2	n^2
(5.1.2)	n	$2n$
(5.1.3)	n	$2n$
(5.1.4)	2	n^2
(5.1.5)	n	$2n$
(5.1.6)	n	$2n$
(5.1.7)	4	$(n-1)n^3$

Table 5.1.1: Number of variables and constraints

5.2 Theoretical results for circuit inequalities

In this section, we formulate the complete list of circuits found in Chapter 4 as circuit inequalities and then maximally lift them to derive strong valid inequalities. For this purpose, assume we have two Latin rectangles, then for any circuit $C \in \mathcal{C}_2$ the corresponding circuit inequality will be violated if C appears in the rectangles and will be satisfied otherwise. Circuit inequalities (and lifted versions) associated with $\mathcal{C}_1$ are listed in [29], hence here we present all such inequalities for members of $\mathcal{C}_2 \setminus \mathcal{C}_1$.

From Definition 2.1.7 we know that every circuit satisfies the exclusion and minimality properties. Therefore for all $C \in \mathcal{C}_2 \setminus \mathcal{C}_1$ the right hand side of the corresponding circuit inequality is $|C| - 1$ where $|C|$ is the cardinality of the circuit. The general form is:

$$\sum\{x_c : c \in C \cap G^1\} + \sum\{y_c : c \in C \cap G^2\} \leq |C| - 1, \quad C \in \mathcal{C}_2 \setminus \mathcal{C}_1$$

or, if C is denoted as $R_- \cup R'_-$,

$$\sum\{x_c : c \in R_-\} + \sum\{y_c : c \in R'_-\} \leq |R \cup R'_-| - 1, \quad (R_- \cup R'_-) \in \mathcal{C}_2 \setminus \mathcal{C}_1$$

For example, (5.1.7) is the set of all inequalities arising from $\mathcal{C}_{2,4}$. Clearly, adding all circuit inequalities restricts the set of feasible integer vectors (x, y) to those corresponding to pairs of rectangles in which any two rows are completable in a way that no pair of values occurs more than once (in these two rows only); hence our interest in actually obtaining these inequalities. However, since the number of circuit inequalities is prohibitively large, it would be far more useful to employ such inequalities in a cutting-plane algorithm, i.e., generating them only if violated by the current LP-solution.

Even better, one would be interested in generating *lifted* circuit-inequalities, i.e., circuit inequalities in which further variables are included one-by-one in their left-hand side with the largest (positive) coefficient such that the augmented inequality remains valid. In our setting, an inequality is valid if it includes integer feasible vectors, i.e., vectors associated with completable pairs of rectangles. This process is known as *sequential lifting* [52]. A lifted circuit inequality has the form,

$$\begin{aligned}&\sum\{a_s x_s :\ s \in S \cap G^1\} + \sum\{a_s y_s :\ s \in S \cap G^2\}\\ &+ \sum\{x_c :\ c \in C \cap G^1\} + \sum\{y_c :\ c \in C \cap G^2\} \leq |C| - 1, \quad C \in \mathcal{C}_2 \setminus \mathcal{C}_1, \qquad (5.2.1)\end{aligned}$$

where $S \subseteq \left(G^1 \cup G^2\right) \setminus C$ and $a_s > 0,\ s \in S$.

Proposition 5.2.1. *No lifted circuit inequality can have a left-hand side coefficient greater than* 2.

Proof. Consider the circuit inequality (5.2.1) for a given circuit $C \in \mathcal{C}_2 \setminus \mathcal{C}_1$, that is augmented by introducing variable x_s with coefficient a_s where $s \in G^1 \setminus C$. Hence,

$$a_s x_s + \sum\{x_c : c \in C \setminus G^1\} + \sum\{y_c : c \in C \setminus G^2\} \leq |C| - 1 \qquad (5.2.2)$$

Let $C(s) = \{c \in C \cap G^1\ :\ |s \cap c| = 2\}$ and let us show that $|C(s)| \leq 3$. For $s = (i_s, j_s, k_s), c \in C(s)$ implies $c \in \{(i_c, j_s, k_s), (i_s, j_c, k_s), (i_s, j_s, k_c)\}$ where $i_c \in I \setminus \{i_s\}$, $\quad j_c \in J \setminus \{j_s\}$ and $k_c \in K \setminus \{k_s\}$. Clearly, $|C(s)| > 3$ only if there are two elements c, d in $C(s)$ sharing the same two indices with s, e.g., (i_c, j_s, k_s) and (i_d, j_s, k_s); but then, circuit C including both (i_c, j_s, k_s) and (i_d, j_s, k_s) implies that value k_s appears twice in column j_s, i.e. a contradiction to the fact that no $C \in \mathcal{C}_2 \setminus \mathcal{C}_1$ violates the Latin square structure.

Next, notice that constraints (5.1.1)-(5.1.3) yield that any $s \in G^1$ appears in the same constraint with some $c \in G^1 \setminus \{s\}$ if and only if c and s share two among the indices i, j, k. It follows that setting $x_s = 1$ implies $x_c = 0$ for all $c \in C(s)$, in which case (5.2.2) becomes,

$$a_s + \sum\{x_c : c \in \left(C \cap G^1\right) \setminus C(s)\} + \sum\{y_c : c \in C \cap G^2\} \leq |C| - 1$$

or, using that $|C(s)| \leq 3$,

$$\begin{aligned} a_s &\leq |C| - 1 - \left(\sum\{x_c : c \in \left(C \cap G^1\right) \setminus C(s)\} + \sum\{y_c : c \in C \cap G^2\}\right) \\ &\leq |C| - 1 - (|C| - |C(s)|) \leq |C| - 1 - (|C| - 3) = 2. \end{aligned}$$

□

For a given circuit $C \in \mathcal{C}_2 \setminus \mathcal{C}_1$ that is partially lifted, let L be the set of indices of all lifted variables. Clearly if no variables have been lifted then $L = \emptyset$.

Corollary 5.2.2. *$a_s \geq 1$ only if $|C(s)| \geq 2$ and*

a) $L(s) = \{c \in C(s),\ l \in L : \ |c \cap l| = 2\} = \emptyset$ or

b) there exists a circuit $C' \in \mathcal{C}_2 \setminus \mathcal{C}_1$ such that $C' \subseteq (C \cup L \cup \{s\}) \setminus C(s)$

Remark 5.2.3. *If case $b)$ of Corollary 5.2.2 holds, then $a_s = 1$.*

5.3 Lifted circuit inequalities

We now continue to formulate circuit inequalities and present their lifted versions. It becomes necessary at this point to revert to initial notation and denote values in cell (i, j) of R_{1-} and R_{2-} with $k^1 \in K^1$ and $k^2 \in K^2$ respectively in order to express lifted circuit inequalities in closed form. To express the values that a given cell can take, we define a $1-1$ mapping $\pi(j)$, such that for a row $i \in I$ and column j $\in J$ the function returns the value contained in that cell (i, j). The function will return value $k^1 \in K^1$ for R_1 and value $k^2 \in K^2$ for R_2.

5.3.1 Lifted circuit inequalities for $\mathcal{C}_{2,4}$

Recall that $\mathcal{C}_{2,4}$ (Table 4.2.1) has a single circuit up to equivalence, hence (5.1.7) represents the circuit inequalities in this class hence,

$$\mathcal{C}_{2,4} : \quad x_{i_1 j_1 k^1} + x_{i_2 j_2 k^1} + y_{i_1 j_1 k^2} + y_{i_2 j_2 k^2} \leq 3,$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2\} \subset J, \quad k^1 \in K^1, \quad k^2 \in K^2$

By Corollary 5.2.2a., it is easy to see that $a_s \geq 1$ only if s is one of (i_1, j_2, k^1), $(i_2, j_1, k^1) \in G^1$ or one of $(i_1, j_2, k^2), (i_2, j_1, k^2) \in G^2$. Notice however that including both $(i_1, j_2, k^1), (i_2, j_1, k^1)$ yields the inequality

$$x_{i_1 j_1 k^1} + x_{i_2 j_2 k^1} + y_{i_1 j_1 k^2} + y_{i_2 j_2 k^2} + x_{i_1 j_2 k^1} + x_{i_2 j_1 k^1} \leq 3$$

which is not valid since setting the last four variables to value 1 is not allowed (because of the left-hand side becoming 4), although the corresponding pair of rectangles (see Table 5.3.1) contains no circuit; it follows that only one of $(i_1, j_2, k^1), (i_2, j_1, k^1)$ can be included, the same applying to $(i_1, j_2, k^2), (i_2, j_1, k^2)$ of G^2 . Hence the lifted circuit inequalities arising from class $\mathcal{C}_{2,4}$ are

$$x_{i_1 j_1 k^1} + x_{i_2 j_2 k^1} + y_{i_1 j_1 k^2} + y_{i_2 j_2 k^2} + x_{i_1 j_2 k^1} + y_{i_1 j_2 k^2} \leq 3, \tag{5.3.1}$$

$$x_{i_1 j_1 k^1} + x_{i_2 j_2 k^1} + y_{i_1 j_1 k^2} + y_{i_2 j_2 k^2} + x_{i_1 j_2 k^1} + y_{i_2 j_1 k^2} \leq 3, \tag{5.3.2}$$

$$x_{i_1 j_1 k^1} + x_{i_2 j_2 k^1} + y_{i_1 j_1 k^2} + y_{i_2 j_2 k^2} + x_{i_2 j_1 k^1} + y_{i_1 j_2 k^2} \leq 3 \qquad \text{and} \tag{5.3.3}$$

$$x_{i_1 j_1 k^1} + x_{i_2 j_2 k^1} + y_{i_1 j_1 k^2} + y_{i_2 j_2 k^2} + x_{i_2 j_1 k^1} + y_{i_2 j_1 k^2} \leq 3, \tag{5.3.4}$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2\} \subset J, \quad k^1 \in K^1, \quad k^2 \in K^2$

Notice that by performing permutations $i_1 \leftrightarrow i_2$ and $j_1 \leftrightarrow j_2$ on inequalities (5.3.1) and (5.3.2), respectively inequalities (5.3.4) and (5.3.3) are obtained. Therefore we can say that up to equivalence the only maximally lifted circuit inequalities are (5.3.1) and (5.3.2). From this point onwards we will only present non-equivalent lifted inequalities.

j_1	j_2	$\cdots$	j_n
	k^1	...	
k^1		...	

R_-

j_1	j_2	$\cdots$	j_n
k^2		...	
	k^2	...	

R'_-

Table 5.3.1: A pair of incomplete rectangles containing no circuit

5.3.2 Lifted circuit inequalities for $\mathcal{C}_{2,3}$

Class 3 consists of two types of circuits, $\mathcal{C}^1_{2,3}$ and $\mathcal{C}^2_{2,3}$. The corresponding circuit inequality for $\mathcal{C}^1_{2,3}$ depicted in Table 4.2.39 is presented below and we list inequalities for $\mathcal{C}^2_{2,3}$ in Appendix B.

j_1	j_2	j_3	j_4	...	j_n
		k_3^1	k_4^1	$K^1 \backslash \{k_1^1, k_2^1, k_3^1, k_4^1\}$	
	k_1^1				

R^*_{25-}

j_1	j_2	j_3	...	j_n
k^2				
	k^2			

R_{28-}

Table 5.3.2: Representative of $\mathcal{C}^1_{2,3}$

j_1	j_2	j_3	j_4	...	j_n
		k_4^1	k_3^1	$K^1 \backslash \{k_1^1, k_2^1, k_3^1, k_4^1\}$	
	k_1^1				

R^*_{25-}

j_1	j_2	j_3	...	j_n
k^2				
	k^2			

R_{28-}

Table 5.3.3: Representative of $\mathcal{C}^1_{2,3}$

$$\mathcal{C}^1_{2,3} \ : \sum_{j \in J \backslash \{j_1, j_2\}} (x_{i_1 j \pi(j)}) + x_{i_2 j_2 k^1} + \quad y_{i_1 j_1 k^2} + y_{i_2 j_2 k^2} \leq n,$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2\} \subset J, \quad \{k_1^1, k_2^1\} \subset K^1, \quad \pi : J \setminus \{j_1, j_2\} \longrightarrow K^1 \setminus \{k_1^1, k_2^1\}, \quad k^2 \in K^2$

We will first lift coefficients for the first row of R^*_{25}. Assume that cells (i_1, j_3) and (i_1, j_4) contain values k_3^1 and k_4^1 respectively, as shown in Table 5.3.2. By Corollary 5.2.2a. it is easy to see that $x_{i_1 j_3 k_4^1}$ can be lifted with value 1, since $|C(s)| = 2$ and $L = \emptyset$. To make this more clear, notice that $|(i_1, j_3, k_4^1) \cap (i_1, j_3, k_3^1)| = 2$ and $|(i_1, j_3, k_4^1) \cap (i_1, j_4, k_4^1)| = 2$ and also notice that if $x_{i_1 j_3 k_4^1}$ takes value 1 then $y_{i_1 j_3 k_3^1}$ and $y_{i_1 j_4 k_4^1}$ are forced to take value 0.

Next, if we introduce variable $x_{i_1 j_4 k_3^1}$ with value 1 notice that now, $L = \{(i_1, j_3, k_4^1)\}, C(s) = \{(i_1, j_3, k_3^1), (i_1, j_4, k_4^1)\}$ thus $L(s) = 2 \neq \emptyset$ and Corollary 5.2.2a. does not hold. However, $x_{i_1 j_3 k_4^1}$ can now take value 1 and therefore constitutes the circuit depicted in Table 5.3.3 and can be expressed as inequality,

$$x_{i_1 j_3 k_4^1} + a_{i_1 j_4 k_3^1} \, x_{i_1 j_4 k_3^1} + \sum_{j \in J \backslash \{j_1, j_2, j_3, j_4\}} (x_{i_1 j \pi(j)}) + x_{i_2 j_2 k_1^1} + y_{i_1 j_1 k^2} + y_{i_2 j_2 k^2} \leq n,$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2, j_3, j_4\} \subset J, \quad \{k_1^1, k_2^1, k_3^1, k_4^1\} \subset K^1, \pi : J \backslash \{j_1, j_2\} \longrightarrow K^1 \backslash \{k_1^1, k_2^1, k_3^1, k_4^1\}, \quad k^2 \in K^2$, therefore from Corollary 5.2.2b. and Remark 5.2.3, $x_{i_1 j_4 k_3^1}$ can be lifted with value 1. By applying this logic, all $x_{i_1 j k^1}$ for $j \in J \setminus \{j_1, j_2\}, \quad k^1 \in K^1 \setminus \{k_1^1, k_2^1, \pi(j)\}$ can be lifted with value 1 simultaneously.

Exactly as per lifting for circuit in $\mathcal{C}_{2,4}$, by Corollary 5.2.2$a.$ it follows that either $y_{i_1 j_2 k^2}$ or $y_{i_2 j_1 k^2}$ can be lifted with value 1. No more values can be lifted with non-zero coefficient and finally the two maximally lifted circuit inequalities shown below are the only ones.

$$\sum_{\substack{j \in J \setminus \{j_1, j_2\} \\ k^1 \in K^1 \setminus \{k_1^1, k_2^1\}}} (x_{i_1 j k^1}) + x_{i_2 j_2 k_1^1} + y_{i_1 j_1 k^2} + y_{i_2 j_2 k^2} + y_{i_1 j_2 k^2} \leq n \qquad \text{and}$$

$$\sum_{\substack{j \in J \setminus \{j_1, j_2\} \\ k^1 \in K^1 \setminus \{k_1^1, k_2^1\}}} (x_{i_1 j k^1}) + x_{i_2 j_2 k_1^1} + y_{i_1 j_1 k^2} + y_{i_2 j_2 k^2} + y_{i_2 j_1 k^2} \leq n,$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2\} \subset J, \quad \{k_1^1, k_2^1\} \subset K^1, \quad k^2 \in K^2$

5.3.3 Lifted circuit inequalities for $\mathcal{C}_{2,2}$

Class 2 consists of nine types of circuits, $\mathcal{C}_{2,2}^1$ - $\mathcal{C}_{2,2}^9$. The corresponding circuit inequalities for $\mathcal{C}_{2,2}^1$ depicted in Table 4.2.32 are presented below and the remaining circuit inequalities and their lifted versions for this class are listed in Appendix B.

$$\begin{aligned} \mathcal{C}_{2,2}^1 \ : \quad & x_{i_1 j_2 k_3^1} + \sum_{j \in J \setminus \{j_1, j_2, j_3\}} (x_{i_1 j \pi_1(j)}) + x_{i_2 j_1 k_2^1} + x_{i_2 j_3 k_n^1} + \sum_{j \in J \setminus \{j_1, j_2, j_3, j_4\}} (x_{i_2 j \pi_2(j)}) \\ & + y_{i_1 j_1 k^2} + y_{i_2 j_2 k^2} \leq 2n - 3 \end{aligned}$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2, j_3, j_4\} \subset J, \quad \{k_1^1, k_2^1, k_3^1, k_n^1\} \subset K^1, \quad k^2 \in K^2, \quad \pi_1 : J \setminus \{j_1, j_2, j_3\} \longrightarrow K^1 \setminus \{k_1^1, k_2^1, k_3^1\}, \quad \pi_2 : J \setminus \{j_1, j_2, j_3, j_4\} \longrightarrow K^1 \setminus \{k_1^1, k_2^1, k_3^1, k_n^1\}$, such that $\pi_1(j) \neq \pi_2(j)$, for all $j \in J \setminus \{j_1, j_2, j_3, j_4\}$

Variables $x_{i_1jk^1}$, for all $j \in J \setminus \{j_1, j_2, j_3\}$, $k^1 \in K^1 \setminus \{k^1_1, k^1_2, k^1_3, \pi(j)\}$ and $x_{i_2jk^1}$, for all $j \in J \setminus \{j_1, j_2, j_3, j_4\}$, $k^1 \in K^1 \setminus \{k^1_1, k^1_2, k^1_3, k^1_n, \pi(j)\}$, can be lifted simultaneously with value 1. From Remark 5.2.2a. if follows that either $x_{i_1j_2k^2}$ or $x_{i_2j_1k^2}$ can be lifted with value 1. No more values can be lifted with non-zero coefficient and the two maximally lifted circuit inequalities shown below are the only ones.

$$\begin{aligned} & x_{i_1j_2k^1_3} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\} \\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_2} + x_{i_2j_3k^1_n} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3,j_4\} \\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3,k^1_n\}}} (x_{i_2jk^1}) \\ & + y_{i_1j_1k^2} + y_{i_2j_2k^2} + y_{i_1j_2k^2} \leq 2n-3, \end{aligned}$$

$$\begin{aligned} & x_{i_1j_2k^1_3} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\} \\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_2} + x_{i_2j_3k^1_n} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3,j_4\} \\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3,k^1_n\}}} (x_{i_2jk^1}) \\ & + y_{i_1j_1k^2} + y_{i_2j_2k^2} + y_{i_2j_1k^2} \leq 2n-3 \end{aligned}$$

where $\{i_1, i_2\} \subseteq I$, $\{j_1, j_2, j_3, j_4\} \subset J$, $\{k^1_1, k^1_2, k^1_3, k^1_n\} \subset K^1$, $k^2 \in K^2$

5.3.4 Lifted circuit inequalities for $\mathcal{C}_{2,1}$

Class 1 consists of six types of circuits, $\mathcal{C}^1_{2,1}$ - $\mathcal{C}^6_{2,1}$. The corresponding circuit inequalities for $\mathcal{C}^1_{2,1}$ depicted in Table 4.2.24 presented below and the remaining circuit inequalities and their lifted versions for this class are listed in Appendix B.

$$\begin{aligned} \mathcal{C}^1_{2,1} \ : \quad & x_{i_1j_2k^1_3} + \sum_{j\in J\setminus\{j_1,j_2,j_3\}} (x_{i_1j\pi_1(j)}) + x_{i_2j_1k^1_2} + x_{i_2j_3k^1_n} + \sum_{j\in J\setminus\{j_1,j_2,j_3,j_4\}} (x_{i_2j\pi_2(j)}) \\ & + \sum_{j\in J\setminus\{j_1,j_2\}} (y_{i_1j\pi_3(j)}) + y_{i_2j_2k^2_1} \leq 3(n-2) \end{aligned}$$

where $\{i_1, i_2\} \subseteq I$, $\{j_1, j_2, j_3, j_4\} \subset J$, $\{k^1_1, k^1_2, k^1_3, k^1_n\} \subset K^1$, $\{k^2_1, k^2_2\} \subset K^2$, $\pi_1 : J\setminus\{j_1, j_2, j_3\} \longrightarrow K^1 \setminus \{k^1_1, k^1_2, k^1_3\}$, $\pi_2 : J \setminus \{j_1, j_2, j_3, j_4\} \longrightarrow K^1 \setminus \{k^1_1, k^1_2, k^1_3, k^1_n\}$, such that $\pi_1(j) \neq \pi_2(j)$, for all $j \in J \setminus \{j_1, j_2, j_3, j_4\}$, $\pi_3 : J \setminus \{j_1, j_2\} \longrightarrow K^2 \setminus \{k^2_1, k^2_2\}$

We can easily derive the lifted circuit inequality by replicating procedure followed for the two circuit inequities presented above for $\mathcal{C}^1_{2,3}$ and $\mathcal{C}^1_{2,2}$. Therefore $x_{i_1jk^1}$, for all $j \in J \setminus \{j_1, j_2, j_3\}$, $k^1 \in K^1 \setminus \{k^1_1, k^1_2, k^1_3, \pi_1(j)\}$ and for $x_{i_2jk^1}$, for all $j \in J\setminus\{j_1, j_2, j_3, j_4\}$, $k^1 \in K^1\setminus\{k^1_1, k^1_2, k^1_3, k^1_n, \pi_2(j)\}$, that can be lifted with value 1. The same also holds for $x_{i_1jk^2}$, for all $j \in J\setminus\{j_1, j_2\}$, $k^2 \in K^2\setminus\{k^2_1, k^2_2, \pi_3(j)\}$. From Remark 5.2.2$a$. if follows that $x_{i_2j_2k^2_2}$ can be lifted with value 1. No more values can be lifted with non-zero coefficient and the maximally lifted circuit inequality shown below is the only one.

$$\begin{aligned} x_{i_1j_2k^1_3} + &\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_2} + x_{i_2j_3k^1_n} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3,j_4\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3,k^1_n\}}} (x_{i_2jk^1}) \\ + &\sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^1_2\in K^2\setminus\{k^2_1,k^2_2\}}} (y_{i_1jk^2}) + y_{i_1j_2k^2} \leq 3(n-2) \end{aligned}$$

where $\{i_1, i_2\} \subseteq I$, $\{j_1, j_2, j_3, j_4\} \subset J$, $\{k^1_1, k^1_2, k^1_3, k^1_n\} \subset K^1$, $\{k^2_1, k^2_2\} \subset K^2$

5.3.5 Lifted circuit inequalities for $\mathcal{C}_{2,0}$

Class 0 consists of nine types of circuits, $\mathcal{C}^1_{2,0}$ - $\mathcal{C}^5_{2,0}$. The corresponding circuit inequalities for $\mathcal{C}^1_{2,1}$ depicted in Table 4.2.16 is presented below and the remaining circuit inequalities and their lifted versions for this class are listed in Appendix B.

$$\begin{aligned} \mathcal{C}^1_{2,0}\ :\quad & x_{i_1j_2k^1_3} + \sum_{j\in J\setminus\{j_1,j_2,j_3\}} (x_{i_1j\pi_1(j)}) + x_{i_2j_1k^1_2} + x_{i_2j_3k^1_n} + \sum_{j\in J\setminus\{j_1,j_2,j_3,j_4\}} (x_{i_2j\pi_2(j)}) \\ & + y_{i_1j_2k^2_3} + \sum_{j\in J\setminus\{j_1,j_2,j_3\}} (y_{i_1j\pi_3(j)}) + y_{i_2j_1k^2_2} + y_{i_2j_3k^2_n} + \sum_{j\in J\setminus\{j_1,j_2,j_3,j_4\}} (y_{i_2j\pi_4(j)}) \leq 2(2n-5) \end{aligned}$$

where $\{i_1, i_2\} \subseteq I$, $\{j_1, j_2, j_3, j_4\} \subset J$, $\{k^1_1, k^1_2, k^1_3, k^1_n\} \subset K^1$, $\{k^2_1, k^2_2, k^2_3, k^2_n\} \subset K^2$, $\pi_1 : J \setminus \{j_1, j_2, j_3\} \longrightarrow K^1 \setminus \{k^1_1, k^1_2, k^1_3\}$, $\pi_2 : J \setminus \{j_1, j_2, j_3, j_4\} \longrightarrow K^1 \setminus \{k^1_1, k^1_2, k^1_3, k^1_n\}$, such that $\pi_1(j) \neq \pi_2(j)$, for all $j \in J\setminus\{j_1, j_2, j_3, j_4\}$, $\pi_3 : J\setminus\{j_1, j_2, j_3\} \longrightarrow K^2\setminus\{k^2_1, k^2_2, k^2_3\}$, $\pi_4 : J\setminus\{j_1, j_2, j_3, j_4\} \longrightarrow K^2 \setminus \{k^2_1, k^2_2, k^2_3, k^2_n\}$, such that $\pi_3(j) \neq \pi_4(j)$, for all $j \in J \setminus \{j_1, j_2, j_3, j_4\}$

The procedure follows as per the previous classes. The maximally lifted circuit inequality shown below is the only one.

$$
\begin{aligned}
& x_{i_1 j_2 k_3^1} + \sum_{\substack{j \in J \setminus \{j_1, j_2, j_3\} \\ k^1 \in K^1 \setminus \{k_1^1, k_2^1, k_3^1\}}} (x_{i_1 j k^1}) + x_{i_2 j_1 k_2^1} + x_{i_2 j_3 k_n^1} + \sum_{\substack{j \in J \setminus \{j_1, j_2, j_3, j_4\} \\ k^1 \in K^1 \setminus \{k_1^1, k_2^1, k_3^1, k_n^1\}}} (x_{i_2 j k^1}) \\
& + y_{i_1 j_2 k_3^2} + \sum_{\substack{j \in J \setminus \{j_1, j_2, j_3\} \\ k^2 \in K^2 \setminus \{k_1^2, k_2^2, k_3^2\}}} (y_{i_1 j k^2}) + y_{i_2 j_1 k_2^2} + y_{i_2 j_3 k_n^2} + \sum_{\substack{j \in J \setminus \{j_1, j_2, j_3, j_4\} \\ k^2 \in K^2 \setminus \{k_1^2, k_2^2, k_3^2, k_n^2\}}} (y_{i_2 j k^2}) \leq 2(2n - 5)
\end{aligned}
$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2, j_3, j_4\} \subset J, \quad \{k_1^1, k_2^1, k_3^1, k_n^1\} \subset K^1, \quad \{k_1^2, k_2^2, k_3^2, k_n^2\} \subset K^2$

5.4 Concluding remarks

This chapter presented a new 3-index IP formulation for finding an OLR2. With this model, one can also test whether a pair of incomplete rectangles is completable. If a solution exists then a completed OLR2 is given otherwise the problem is infeasible. All circuits found in Chapter 4 were formulated initially as valid inequalities and then strengthened with the process of lifting. It was proven that no lifted circuit inequality can have a coefficient larger than 2. It was also shown that most circuits gave rise to more than one maximally lifted inequalities and their structure vary depending on the lifting sequence.

As topics for further research, it would be interesting to show whether these maximally lifted inequalities are facet defining and to assess their performance as added cuts for the LP relaxation problem. For the latter, a polynomial-time separation algorithm will have to be constructed in order to add the right cuts at each iteration of solving the LP relaxation problem.

The next chapter extends the 3-index IP formulation for the more general case of 3MOLRm and presents an additional 5-index formulation for solving the same problem. Lastly CP models are introduced and results on some initial computational results are mentioned.

Chapter 6

On the clutter of circuits $\mathcal{C}_3$ and formulations for sets for 3 MOLRm

This chapter extends the previous work on OLR2 to examining sets of 3 MOLR2. This sets the scene for further research in this direction; that is to obtain circuits for sets of $|T|$ MOLR2 and for more than two rows, to potentially address the bigger problem of characterising circuits for MOLS in general. The chapter is organised as follows.

Section 6.1 looks at circuits for the 3 MOLR2 problem. Seven new circuits in $\mathcal{C}_3 \setminus \mathcal{C}_2$ are found and it is explained in detail how these differ from circuits in $\mathcal{C}_2$. Briefly, these are circuits whose corresponding incomplete rectangles are individually completable to Latin rectangles and also any two of them can be completed to an orthogonal pair. However, when all three are completed simultaneously they are no longer all pairwise orthogonal.

For the general problem of finding or completing a set of 3 MOLRm where $m \leq n$, Sections 6.2 and 6.3 present two very different Integer Programming (IP) formulations. The first, in Section 6.2, extends the 3-index formulation discussed in Section 5.1 to now address the problem of finding a set of 3 MOLRm (consisting $3mn^2$ binary variables). Section 6.3 extends the 4-index planar assignment formulation for finding an OLS, to a 5-index formulation (consisting of mn^4 binary variables), for finding a set of 3 MOLRm. The former is attributed to D. Gale (in [19]) and one of the most studied Integer Programming (IP) formulations for finding OLS particularly by Appa et al. in [2], [3], [4], [5], [6].

Finally, Section 6.4 lists Constraint Programming (CP) formulations for finding a single Latin rectangle, an OLRm or a set of 3 MOLRm and then shows how these formulations can be extended to address the general problem of finding a set of $|T|$ MOLRm where $m \leq n$. The formulations are based on `alldiff` constraints and the principal idea here is that values appearing in each row (column) of a rectangle must be all different to each other and similarly for the collection of all pairs of value for any two rectangles. Three redundant models (in the sense that they all cover the same solution space) are introduced and linked to each other using channelling and inverse constraints as defined in [39] and [59]. This is done to improve constraint propagation and consequently computational time, as in suggested in [16], [10]. Finally we present a new CP formulation that is unique for the OLS case. Given that all n^2 possible pairs of values are found in any OLS, this formulation assigns a cell to each of these pairs. Some initial computational times for the CP models are mentioned.

6.1 An introduction to $\mathcal{C}_3$

Corollary 3.2.3 shows that $\mathcal{C}_2 \subset \mathcal{C}_3$. All members of $\mathcal{C}_2$ were characterised in the previous section and in this section we list some members of $\mathcal{C}_3 \setminus \mathcal{C}_2$, i.e. circuits in $\mathcal{C}_3$ that do not appear in $\mathcal{C}_2$. Essentially, these are minimal dependent subsets of $S^{\mathcal{B}_3}$, that are not members of $\mathcal{C}_2$ and consequently not members of $\mathcal{C}_1$.

To examine how these different circuits affect completablilty, let us consider three incomplete rectangles R_{1-}, R_{2-} and R_{3-}. The following four scenarios are possible:

Scenario 1**:** All three rectangles comprising the triple are completable to set of 3 MORL2, indicating that $R_{1-} \cup R_{2-} \cup R_{3-}$ does not contain a circuit member of $\mathcal{C}_3$.

Scenario 2**:** Any of the rectangles comprising the triple is not individually completable to a Latin rectangle indicating that at least one of R_{1-}, R_{2-}, R_{3-} contains at least one member of $\mathcal{C}_1$ and obviously the three rectangles cannot be completed to form a set of 3 MORL2.

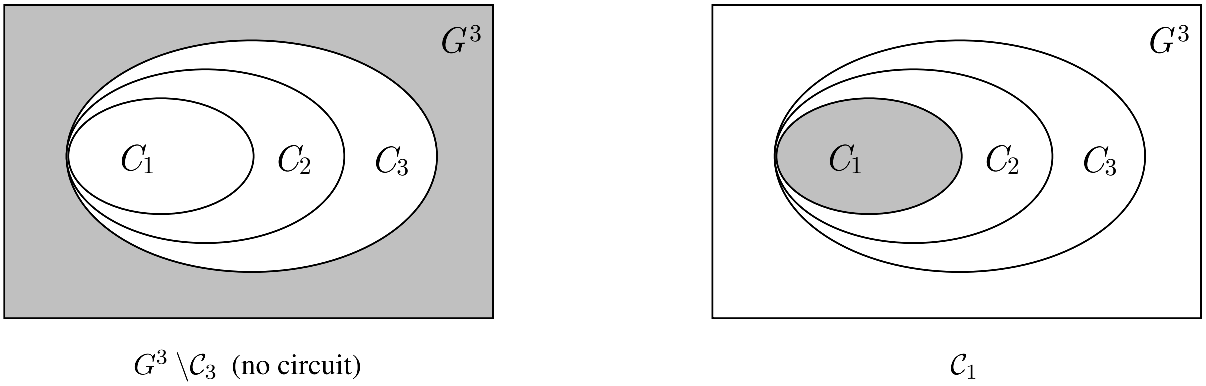
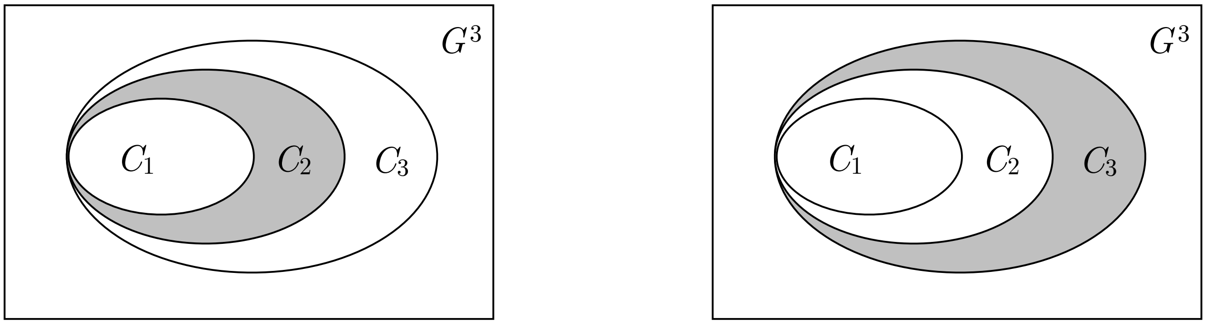


Figure 6.1.1: Venn diagram illustrating Scenario 1 (left), Scenario 2 (right)

Scenario 3**:** All rectangles are individually completable to a Latin rectangle however, an attempt to complete two rectangles leads to a pair of values being repeated, indicating that there exists a circuit of $\mathcal{C}_2 \setminus \mathcal{C}_1$.

Scenario 3**:** Finally, all pairs of rectangles are completable to orthogonal pairs however, any attempt to complete all three leads to a pair of values being repeated, indicating that there exits a circuit of $\mathcal{C}_3 \setminus \mathcal{C}_2$.



$\mathcal{C}_2 \backslash \mathcal{C}_1$ $\mathcal{C}_3 \backslash \mathcal{C}_2$

Figure 6.1.2: Venn diagram illustrating Scenario 3 (left), Scenario 4 (right)

Therefore to clearly demonstrate the properties of the new circuits, we comment that every circuit $C \in \mathcal{C}_3$ is a member of $\mathcal{C}_3 \backslash \mathcal{C}_2$ if:

- Every incomplete rectangle R_{t-} in C, where $t \in \{1, 2, 3\}$ is completable to a Latin rectangle
- Every pair of incomplete rectangles $R_{t-} \cup R'_{t-}$ in C, where $t, t' \in \{1, 2, 3\}$ is completable to an OLR2
- The triple $R_{1-} \cup R_{2-} \cup R_{3-}$ cannot be completed to a set of 3 MOLR2

In Table 6.1.1 we give a new representatives of $\mathcal{C}_3$. To simplify notation we assume that $I = \{1, 2\}$ and $K^1 = K^2 = K^3 = \{1, \ldots, n\}$.

j_1	j_2	j_3	
		1	
	1		

R_{1-}

j_1	j_2	j_3	
1			
	1		

R_{2-}

j_1	j_2	j_3	
			$K^3\backslash\{1,2,3\}$
			$K^3\backslash\{1,2,3\}$

R_{3-}

Table 6.1.1: Circuit in $\mathcal{C}_3 \backslash \mathcal{C}_2$

Notice that this set of 3 rectangles does not contain a circuit of $\mathcal{C}_1$ since R_{1-}, R_{2-}, R_{3-} are completable to Latin rectangles. It also does not contain a circuit of $\mathcal{C}_2 \backslash \mathcal{C}_1$; it is easy to see that $R_{1-} \cup R_{2-}$ is completable to an OLR2 and the same holds for $R_{1-} \cup R_3$ and $R_{2-} \cup R_{3-}$ as shown in Tables 6.1.2 and 6.1.3.

j_1	j_2	j_3	
k_1	k_2	1	...
k_3	1	k_4	...

R_1

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
3	1	2	$K^3\backslash\{1,2,3\}$

R_3

Table 6.1.2: An OLR2

j_1	j_2	j_3	
1	k_1	k_2	...
k_3	1	k_4	...

R_2

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
2	3	1	$K^3\backslash\{1,2,3\}$

R_3

Table 6.1.3: An OLR2

It remains to establish that Table 6.1.1 satisfies both the exclusion and minimality properties for a set of 3 MOLR2. To prove the former, we attempt to complete Table 6.1.1 as shown in Table 6.1.4. Cells in the first row of R_{3-}, with no loss of generality, can be filled in natural order. Since value 2 appears in cell $(1, j_2)$, then according to the Latin rectangle definition, cell $(2, j_2)$ can only be filled with values 1 or 3. However, notice that selecting value 1 will result in a repetition of pair $(1, 1)$ in $R_{2-} \cup R_{3-}$ (see Table 6.1.5) and selecting value 3 will result in a repetition of pair $(1, 3)$ in $R_{1-} \cup R_{3-}$ (see Table 6.1.6); this establishes that $R_{1-} \cup R_{2-} \cup R_{3-}$ contains a circuit of $\mathcal{C}_3 \backslash \mathcal{C}_2$.

j_1	j_2	j_3	
		1	
	1		

R_{1-}

j_1	j_2	j_3	
1			
	1		

R_{2-}

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
	1 or 3		$K^3\backslash\{1,2,3\}$

R_{3-}

Table 6.1.4: Completing rectangles of Table 6.1.1

j_1	j_2	j_3	
		1	
	1		

R_{1-}

j_1	j_2	j_3	
1			
	1		

R_{2-}

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
	1		$K^3\backslash\{1,2,3\}$

R_{3-}

Table 6.1.5: Completing rectangles of Table 6.1.1

j_1	j_2	j_3	
		1	
	1		

R_{1-}

j_1	j_2	j_3	
1			
	1		

R_{2-}

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
	3		$K^3\backslash\{1,2,3\}$

R_{3-}

Table 6.1.6: Completing rectangles of Table 6.1.1

Finally, to prove that the minimality property is also satisfied, notice that if any cell of Table 6.1.1 is emptied, then the set is completable hence $R_{1-} \cup R_{2-} \cup R_{3-}$ is a circuit. Table 6.1.7 presents a set of 3 MOLR2 that can be obtained if cell $(1, j_4)$ of R_{3-} in Table 6.1.1 is emptied and notice that set of 3 MOLR2 can be obtained if any cell of Table 6.1.1 is emptied.

j_1	j_2	j_3	
k_1	k_2	1	...
k_3	1	k_4	...

R_1

j_1	j_2	j_3	
1	k_5	k_6	...
k_7	1	k_8	...

R_2

j_1	j_2	j_3	j_4	
1	2	k_9	3	$K^3\backslash\{1,2,3,k_9\}$
2	3	1	$K^3\backslash\{1,2,3\}$	

R_{3-}

Table 6.1.7: A set of 3 MOLR2

Tables 6.1.8 to 6.1.13 present six additional non-equivalent circuits in $\mathcal{C}_3 \backslash\mathcal{C}_2$. Relevant proofs for exclusion and minimality are presented in Appendix C.

j_1	j_2	j_3	
		1	
	1		

R_{1-}

j_1	j_2	j_3	
			$K^2\backslash\{1,2,3\}$
			$K^2\backslash\{1,2,3\}$

R_{2-}

j_1	j_2	j_3	
			$K^3\backslash\{1,2,3\}$
			$K^3\backslash\{1,2,3\}$

R_{3-}

Table 6.1.8: Circuit in $\mathcal{C}_3 \backslash\mathcal{C}_2$

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3	
			$K^2\backslash\{1,2,3\}$
			$K^2\backslash\{1,2,3\}$

R_{2-}

j_1	j_2	j_3		j_n
		$K^3\backslash\{1,2,3\}$		
			$K^3\backslash\{1,2,3\}$	

R_{3-}

Table 6.1.9: Circuit in $\mathcal{C}_3 \backslash\mathcal{C}_2$

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3		j_n
		$K^2\backslash\{1,2,3\}$		
			$K^2\backslash\{1,2,3\}$	

R_{2-}

j_1	j_2	j_3		j_n
		$K^3\backslash\{1,2,3\}$		
			$K^3\backslash\{1,2,3\}$	

R_{3-}

Table 6.1.10: Circuit in $\mathcal{C}_3 \backslash\mathcal{C}_2$

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3	
1			
	1		

R_{2-}

j_1	j_2	j_3	
			$K^3\backslash\{1,2,3\}$
			$K^3\backslash\{1,2,3\}$

R_{3-}

Table 6.1.11: Circuit in $\mathcal{C}_3 \backslash\mathcal{C}_2$

j_1	j_2	j_3	
		1	
	1		

R_{1-}

j_1	j_2	j_3	
1			
	1		

R_{2-}

j_1	j_2	j_3	
			$K^3\backslash\{1,2,3\}$
	1		

R_{3-}

Table 6.1.12: Circuit in $\mathcal{C}_3 \backslash\mathcal{C}_2$

j_1	j_2	j_3	j_4	
				$K^1\backslash\{1,2,3,4\}$
				$K^1\backslash\{1,2,3,4\}$

R_{1-}

j_1	j_2	j_3	
			$K^2\backslash\{1,2,3\}$
			$K^2\backslash\{1,2,3\}$

R_{2-}

j_1	j_2	j_3	
			$K^3\backslash\{1,2,3\}$
			$K^3\backslash\{1,2,3\}$

R_{3-}

Table 6.1.13: Circuit in $\mathcal{C}_3 \backslash \mathcal{C}_2$

6.2 A 3-index Integer Programming formulation for a set of 3 MOLRm

The IP resented below is a generalisation of the formulation presented in Section 5.1, only here, the objective is to produce 3 mutually orthogonal Latin rectangles of order n.

Consider the disjoint sets $T = \{1, ..., 3\}$, $I = \{1, ..., m\}$ and $J = \{1, ..., n\}$ and the 3 disjoint sets $K^t = \{k^t_1, ..., k^t_n\}$, where $t \in T$, and K^t is the set of values in R_t. Let x_{ijk^t} be a binary variable that takes value 1 if k^t appears in cell (i, j) of R_t and takes value 0 otherwise.

For three rectangles each having m rows, the solution can have at most $3mn$ variables take value 1, in which case it produces a set of 3 m-row orthogonal Latin rectangles of order n. For solutions with less than $3mn$ variables, the IP provides an incomplete set.

$$Max \sum \{x_{ijk^t} : \;\; i \in I, \;\; j \in J, \;\; k^t \in K^t, \;\; t \in T\}$$

Subject to,

Latin rectangle constraints:

$$\sum \{x_{ijk^t} : i \in I\} \leq 1, \qquad \forall\, j \in J, \quad k^t \in K^t, \quad t \in T, \tag{6.2.1}$$

$$\sum \{x_{ijk^t} : j \in J\} \leq 1, \qquad \forall\, i \in I, \quad k^t \in K^t, \quad t \in T, \tag{6.2.2}$$

$$\sum \{x_{ijk^t} : k^t \in K^t\} \leq 1, \quad \forall\, i \in I, \quad j \in J, \quad t \in T, \tag{6.2.3}$$

Orthogonality constraints:

$$x_{i_1 j_1 k^{t_1}} + x_{i_1 j_1 k^{t_2}} + x_{i_2 j_2 k^{t_1}} + x_{i_2 j_2 k^{t_2}} \leq 3, \tag{6.2.4}$$

$$\forall \quad \{i_1, i_2 \,:\, i_2 > i_1\} \subseteq I, \quad \{j_1, j_2\} \subset J, \quad k^{t_1} \in K^{t_1}, \quad k^{t_2} \in K^{t_2}, \quad \{t_1, t_2\} \subset T,$$

$$x_{ijk^t} \in \{0, 1\}, \quad \forall\, i \in I, \quad j \in J, \quad k^t \in K^t, \quad t \in T \tag{6.2.5}$$

Constraints (6.2.1) ensure that no value is repeated in each column of R_t, constraints (6.2.2) ensure that no value is repeated in each row of R_t and constraints (6.2.3) ensure that every cell of the R_t contains exactly one value.

Finally, constraints (6.2.4) ensure that all Latin rectangles comprising the set of $|T|$ MOLRm are pairwise orthogonal i.e. forbid a pair of values to appear twice in each pair of rectangles. Therefore, for each pair of rectangles, we introduce a constraint for each possible combination of four variables that represent a repetition of a pair, and allowing at most three out of the four variables to take value 1. For clarity, consider the example presented in Table 6.2.1 in which orthogonality is violated in all pairs of rectangles i.e. in $R_{1-} \cup R_{2-}$ the values (k^1, k^2) are repeated in cells $(1, j_1)$ and $(2, j_2)$; in $R_{1-} \cup R_{3-}$ the values (k^1, k^3) are repeated in cells $(1, j_1)$ and $(2, j_2)$ and in $R_{2-} \cup R_{3-}$ the values (k^2, k^3) are repeated in cells $(1, j_1)$ and $(2, j_2)$.

j_1	j_2	j_3	
k^1			
	k^1		

R_{1-}

j_1	j_2	j_3	
k^2			
	k^2		

R_{2-}

j_1	j_2	j_3	
k^3			
	k^3		

R_{3-}

Table 6.2.1: A violation of orthogonality

The IP would prevent this violation from occurring with the following orthogonality constraints of (6.2.4) :

$$\begin{aligned} x_{i_1j_1k^1} + x_{i_2j_2k^1} + x_{i_1j_1k^2} + x_{i_2j_2k^2} &\leq 3 \\ x_{i_1j_1k^1} + x_{i_2j_2k^1} + x_{i_1j_1k^3} + x_{i_2j_2k^3} &\leq 3, \\ x_{i_1j_1k^2} + x_{i_2j_2k^2} + x_{i_1j_1k^3} + x_{i_2j_2k^3} &\leq 3 \end{aligned}$$

The formulation is defined on $3mn^2$ binary variables and a total of $3n^2(12 + m(m-1)n(n-1))/4$ constraints. The solution allows for at most $3mn$ variables to equal 1.

For the case where $m = n$ we are interested in finding a set of 3 MOLS therefore the inequalities (6.2.1) to (6.2.3) can be written as equality constraints enforcing each value to appear exactly once in each row and column of each rectangle. Clearly in this case the objective function becomes arbitrary and if a set of 3 MOLS exists, then we have a feasible solution with exactly $3n^2$ variables taking value 1 and if not, then the problem is infeasible.

The formulation can be instantly generalised to a set of $|T|$ MOLRm, where $|T| \leq (n-1)$ is the number of mutually orthogonal Latin rectangles, if set $T = \{1, ..., 3\}$ is replaced by $T = \{1, ..., |T|\}$ In this generalised case, if a set of $|T|$ MOLRm exist, then the optimal solution consists of $|T|$mn variables taking value 1. In general this model contains a total of $|T|mn^2$ variables and $|T|n^2(12 + m(m-1)n(n-1))/4$ constraints.

6.3 A 5-index Integer Programming formulation for a set of 3 MOLRm

A number of mathematical programming formulations have been given for the OLS problem in [7]. The one most studied, is a binary 4-index planar assignment formulation attributed to D. Gale (in [19]) who is quoted as the person who first suggested the application of integer programming to the OLS problem. Due to its simplicity and symmetry it has formed the base for work on MOLS by Appa et al. in [2], [3], [4], [5], [6]. Here, we first extend Gale's formulation for sets of 3 MOLRm and then extend it for the general MOLS case.

Similarly to the 3-index formulation, the 5-index formulation also produces a set of 3 mutually orthogonal Latin rectangles of order n if such a structure exists, or provides a set of 3 incomplete rectangles in which no violation explicitly occurs and the maximum number of cells are filled. We consider the same disjoint sets $T = \{1, ..., 3\}$, $I = \{1, ..., m\}$ and $J = \{1, ..., n\}$ and the 3 disjoint sets $K^t = \{k_1^t, ..., k_n^t\}$, where K^t is the set of values in R_t, $t \in T$. Let variable $x_{ijk^1k^2k^3}$ be 1 if values k^1, k^2 and k^3 appear in cell (i, j) of R_1, R_2 and R_3 respectively.

We now explain one of the constraints in detail to show how the formulation is constructed. For example, rectangles R_2 and R_3 must be orthogonal hence each of the ordered pairs of values (k^2, k^3) must appear at most once, irrespective of value k^1. This property can be ensured by constraint:

$$\sum\{x_{ijk^1k^2k^3} : i \in I,\ j \in J,\ k^3 \in K^3\} \leq 1, \qquad \forall\ k^1 \in K^1,\ k^2 \in K^2$$

In a similar manner an additional 9 set of constraints can be created as shown in the formulation below, to ensure that the solution complies with the Latin rectangle definition and that the orthogonality condition is imposed.

$$Max \sum \{x_{ijkl} : i \in I,\ j \in J,\ k^1 \in K^1,\ k^2 \in K^2,\ k^3 \in K^3\}$$

Subject to

$$\sum\{x_{ijk^1k^2k^3} : i \in I,\ j \in J,\ k^1 \in K^1\} \leq 1, \qquad \forall\ k^2 \in K^2,\ k^3 \in K^3, \qquad (6.3.1)$$

$$\sum\{x_{ijk^1k^2k^3} : i \in I,\ j \in J,\ k^2 \in K^2\} \leq 1, \qquad \forall\ k^1 \in K^1,\ k^3 \in K^3, \qquad (6.3.2)$$

$$\sum\{x_{ijk^1k^2k^3} : i \in I,\ j \in J,\ k^3 \in K^3\} \leq 1, \qquad \forall\ k^1 \in K^1,\ k^2 \in K^2, \qquad (6.3.3)$$

$$\sum\{x_{ijk^1k^2k^3} : i \in I,\ k^1 \in K^1,\ k^2 \in K^2\} \leq 1, \qquad \forall\ j \in J,\ k^3 \in K^3, \qquad (6.3.4)$$

$$\sum\{x_{ijk^1k^2k^3} : i \in I,\ k^1 \in K^1,\ k^3 \in K^3\} \leq 1, \qquad \forall\ j \in J,\ k^2 \in K^2, \qquad (6.3.5)$$

$$\sum\{x_{ijk^1k^2k^3} : i \in I,\ k^2 \in K^2,\ k^3 \in K^3\} \leq 1, \qquad \forall\ j \in J,\ k^1 \in K^1, \qquad (6.3.6)$$

$$\sum\{x_{ijk^1k^2k^3} : j \in J,\ k^1 \in K^1,\ k^2 \in K^2\} \leq 1, \qquad \forall\ i \in I,\ k^3 \in K^3, \tag{6.3.7}$$

$$\sum\{x_{ijk^1k^2k^3} : j \in J,\ k^1 \in K^1,\ k^3 \in K^3\} \leq 1, \qquad \forall\ i \in I,\ k^2 \in K^2, \tag{6.3.8}$$

$$\sum\{x_{ijk^1k^2k^3} : j \in J,\ k^2 \in K^2,\ k^3 \in K^3\} \leq 1, \qquad \forall\ i \in I,\ k^1 \in K^1, \tag{6.3.9}$$

$$\sum\{x_{ijk^1k^2k^3} : k^1 \in K^1\ k^2 \in K^2,\ k^3 \in K^3\} \leq 1, \qquad \forall\ i \in I,\ j \in J, \tag{6.3.10}$$

$$x_{ijk^1k^2k^3} \in \{0,1\} \qquad \forall\ i \in I,\ j \in J,\ k^1 \in K^1\ k^2 \in K^2,\ k^3 \in K^3$$

The formulation is defined by n^5 binary variables, out of which n^3 have a coefficient of 1 in each of the $10n^2$ constraints. The solution allows for at most mn variables to equal 1.

In this formulation, when starting with incomplete rectangles as an initial solution, a filled cell cannot always be expressed by pre-setting particular variables to take value 1. In fact, it cannot, if that cell is empty in at least one of the other rectangles. More specifically, consider a set of 3 incomplete MOLRm and take cell $(1,1)$ in R_{1-} to contain value 1. In both R_{2-} and R_{3-} take cell $(1,1)$ to be empty. Then the filled cell of R_{1-} can be expressed as constraint,

$$\sum\{x_{111k^2k^3} : k^2 \in K^2,\ k^3 \in K^3\} \geq 1 \tag{6.3.11}$$

For sets of $|T|$ MOLRm of order n, one can extend this formulation to a $(|T|+2)$-index assignment formulation, defined by $n^{|T|+2}$ binary variables out of which exactly $n^{|T|}$ have a coefficient of 1 in each of the $\binom{|T|+2}{2}$ constraints. The solution will restrict at mn variables to equal 1.

For the case where $m = n$ we have sets of $|T|$ MOLS of order n. The problem is now translated into a feasibility problem that identifies an MOLS pair of order n or proves that no such exists. Every feasible solution to this problem is also an optimal one. The solution now restricts n^2 variables to equal 1, which is the exact number of entries in each square of the MOLS. All constraints can now become equalities and we can now have a $(|T|+2)$-index planar assignment problem.

6.4 A Constraint Programming formulation for a set of 3 MOLRm

In this section we present a series of Constraint Programming (CP) formulations for finding a single Latin rectangle, an OLR or a set of 3 MOLRm and then show how these formulations can be extended to address the general problem of finding a set of $|T|$ MOLRm. What differentiates them from other formulations presented in [7], [5], is that here we introduce new techniques from the literature for the purpose of reducing computational time.

There are a number of references in the literature, that support the addition of redundant constraints improve computational times by increasing constraint propagation [59], [39], however there is no systematic approach as to how these constraints should be generated. Inspired by cases where these constraints have proven computationally beneficial results [16], [10], we present two models that leverage the advantages of *channelling* constraints. These are used to connect two or more independent formulations for the same problem; and by independent we mean models that provide the complete description of the problem's solution space.

We start by introducing three models for the Latin rectangle problem. Clearly for $m = n$ the models generate a Latin square. Let $I = \{1, ..., m\}$, $m \leq n$ and J, $K = \{1, ..., n\}$ be three disjoint sets denoting the rows, columns an values respectively, of a Latin rectangle.

We first describe a model based on *value constraints*, these are `alldiff` expressions imposed on the values appearing in the rectangle. Let x_{ij} be the value in row i and column j of the rectangle R,

$$\texttt{alldiff}(x_{ij} \,:\, j \in J), \quad \forall\, i \in I$$

$$\texttt{alldiff}(x_{ij} \,:\, i \in I), \quad \forall\, j \in J$$

The first constraint forces all values appearing in each row to be different, whereas the second imposes the same condition for the columns. This feasibility model generates a Latin rectangle (square). To answer the completability question of an incomplete rectangle (square) we first fix the variables corresponding to filled cells, to take value 1 and then the model with show the problem is feasible if the rectangle is completable and or infeasible otherwise.

A similar model can be obtained if instead of focusing on values we introduce *row constraints*. Let r_{jk} be the row containing value k in column j of R,

$$\texttt{alldiff}(r_{jk} \,:\, k \in K), \quad \forall\, j \in J$$

$$\texttt{alldiff}(r_{jk} \,:\, j \in J)\ \ , \quad \forall\, k \in K$$

The first constraint forces all rows containing a value in a particular column, to be different. More specifically, it forces rows in R containing values k in a column j to be different. The second forces the rows containing a particular value, to be different across all columns.

Now, an alternative third model can be obtained if instead of focusing on rows or values we introduce *column constraints*. Let c_{ik} be the column containing value k in row i of R,

$$\texttt{alldiff}(c_{ik} \,:\, k \in K), \quad \forall\, i \in I$$

$$\texttt{alldiff}(c_{ik} \,:\, i \in I), \quad \forall\, k \in K$$

Similar logic is applied here as per the row constraints. The first constraint forces all columns containing a value in a particular row, to be different, i.e. it forces columns in R containing values k in a row i to be different. The second constraint forces the columns containing a particular value, to be different across all rows.

All three models independently, describe the solution space of the Latin rectangle problem, hence only one of the three is necessary to produce a solution and the remaining two are redundant. However they can all be linked to each other to improve propagation. In [39], J. Hooker states the combination of redundant constraints along with *channelling* constraints improve constraint propagation and as a result provide much improved computational times. B. Smith in [59] also states, that channelling constraints can be replaced by *inverse* constraints as they offer shorter running times even though constraint propagation is the same. For the Latin squares problem we discovered this is true for orders higher than 9. Channelling constraints can be formulated as follows,

$$\begin{aligned}
r_{jx_{ij}} &= i, \quad \forall\, i \in I,\ j \in J, & x_{r_{jk}j} &= k, \quad \forall\, j \in J,\ k \in K,\\
c_{ix_{ij}} &= j, \quad \forall\, i \in I,\ j \in J, & x_{ic_{ik}} &= k, \quad \forall\, i \in i,\ k \in K,\\
r_{c_{ik}k} &= i, \quad \forall\, i \in I,\ k \in K, & c_{r_{jk}k} &= j, \quad \forall\, j \in J,\ k \in K
\end{aligned}$$

The first set of constraints links the value constraints with the row constraints, the second set links the value constraints with the column constraints and finally the third set links the row with column constraints.

Alternatively, the three models can be connected the use of inverse constraints, which force two sets of variables to be in strict correspondence with each other. Assume the variables are represented by two arrays, then an inverse constraint makes sure that the i^{th} item of one array is the inverse of the i^{th} item of the other array, and vice versa. Hence the channelling constraints can be recreated as,

$$\begin{aligned}
&\texttt{inv}(x_{ij} \,:\, i \in I,\ r_{jk} \,:\, k \in K), && \forall\, j \in J,\\
&\texttt{inv}(x_{ij} \,:\, i \in I,\ c_{ik} \,:\, k \in K), && \forall\, i \in I,\\
&\texttt{inv}(r_{jk} \,:\, j \in J,\ c_{ik} \,:\, i \in I), && \forall\, k \in K
\end{aligned}$$

Initial tests using IBM ILOG CPLEX Optimization Studio V12.5, for $m = n$ and $2 \leq n \leq 100$ showed that computational times significantly and consistently improve with the addition of redundant models to the first model. In fact, an average improvement of at least 25% and 88% is reached with use of channelling and inverse constraint respectively. These initial tests showed that inverse constraints for this case perform better than channelling constraints.

The formulation above can easily be extended to address the problem of finding a set of 3 MOLRm and consequently for more sets (including the OLR case). We start by constructing the first model based on the *value constraints* for each rectangle R_1, R_2 and R_3 comprising the set of 3 MOLRm. Let x^1_{ij}, x^2_{ij} and x^3_{ij} be the value in row i and column j of R_1, R_2 and R_3 respectively and constraints can be formulated as follows:

$$\begin{array}{llll}
\texttt{alldiff}(x^1_{ij} : j \in J), & \texttt{alldiff}(x^2_{ij} : j \in J), & \texttt{alldiff}(x^3_{ij} : j \in J), & \forall\, i \in I \\
\texttt{alldiff}(x^1_{ij} : i \in I), & \texttt{alldiff}(x^2_{ij} : i \in I), & \texttt{alldiff}(x^3_{ij} : i \in I), & \forall\, j \in J \\
\texttt{alldiff}(\{x^1_{ij}, x^2_{ij}\} : i \in I,\ j \in J), & & \texttt{alldiff}(\{x^1_{ij}, x^3_{ij}\} : i \in I,\ j \in J) &
\end{array}$$

The first two sets of constraints force all values appearing in each row and column to be different, whereas the last set force all pairs of values occurring from superimposing R_1, R_2 and R_3 to be different (orthogonality constraint).

In similar manner, as for the Latin rectangle problem, we can easily introduce new redundant models (with column and row constraints) as well as channelling and inverse constraints to link these models with each other.

We now continue to introduce a new formulation for the problem of finding a pair of OLS. This is a feasibility problem and the goal is to assign a value to every cell of the Latin rectangle such that the Latin square and orthogonality conditions are satisfied. A primary CP formulation was described and analysed [5] and was proven to achieve better running times than any IP relaxation technique that concludes infeasibility for the pairs of OLS problem of order 6. Aiming to improve previous results and also for the purpose of to introducing new formulations for infeasible problems of higher orders (i.e. sets of 3 MOLS or order 10), we constructed the following model based on linking two redundant formulations.

We introduce the first model based on `alldiff` *value* constraints. Let x^1_{ij} and x^2_{ij} be the value in row i and column j of the first square (S_1) and the second square (S_2) respectively,

$$\begin{array}{lll}
\texttt{alldiff}(x^1_{ij}, : j \in J), & \texttt{alldiff}(x^2_{ij}, : j \in J), & \forall\, i \in I \\
\texttt{alldiff}(x^1_{ij}, : i \in I), & \texttt{alldiff}(x^2_{ij}, : i \in I), & \forall\, j \in J \\
\multicolumn{3}{c}{\texttt{alldiff}(\{x^1_{ij}, x^2_{ij}\}, : i \in I,\ j \in J),}
\end{array}$$

The first two sets of constraints force all values appearing in each row and column to be different, whereas the last set force all pairs of values occurring from superimposing S_1 and S_2 to be different (orthogonality constraint).

A second model can be constructed if we look at the problem from a different angle. Instead of thinking that we have an $n \times n$ square whose cells we need to fill with distinct pairs of values without violating the orthogonality condition, we can reverse this logic to say we have a set of distinct n^2 pairs of values all of which must be placed in a $n \times n$ square such that no cell contains more than one pair. Clearly the Latin square conditions must be respected in both models (i.e. each value to appear once in each row and column). Hence, let $r_{k^1k^2}$ and $c_{k^1k^2}$ be the row and column containing the pair of values (k^1,k^2), where $k^1 \in K^1$ are the values appearing S_1 and $k^2 \in K^2$ are the values appearing in S_2.

$$\texttt{alldiff}(r_{k^1k^2} : k^1 \in K^1), \quad \texttt{alldiff}(c_{k^1k^2} : k^1 \in K^1), \qquad \forall\, k^2 \in K^2$$
$$\texttt{alldiff}(r_{k^1k^2} : k^2 \in K^2), \quad \texttt{alldiff}(c_{k^1k^2} : k^2 \in K^2), \qquad \forall\, k^1 \in K^1$$
$$\texttt{alldiff}(\{r_{k^1k^2}, c_{k^1k^2}\} : k^1 \in K^1,\ k^2 \in K^2)$$

The first two sets of constrains ensure that each row and column of S_1 and S_2 contain distinct values and the last constraints ensures that each combination of row and column , i.e. each cell (after superimposing S_1 and S_2 , contains no more than one pair of values. Finally, to link the two independent models we introduce the channelling constraints,

$$r_{x^1_{ij}x^2_{ij}} = i, \qquad c_{x^1_{ij}x^2_{ij}} = j, \qquad \forall\, i \in I,\ j \in J$$
$$x^1_{r_{k^1k^2}c_{k^1k^2}} = k^1, \quad x^2_{r_{k^1k^2}c_{k^1k^2}} = k^2, \qquad \forall\, k^1 \in K^1,\ k^2 \in K^2$$

Initial tests using IBM ILOG CPLEX Optimization Studio V12.5, for $2 \leq n \leq 9$ showed that computational times significantly and consistently improve with the addition of redundant models to the first model. An average improvement of at least 30% and 70% is reached with use of channelling and inverse constraint respectively. Once again, these initial tests showed that inverse constraints for this case perform better than channelling constraints.

6.5 Concluding remarks

This section first presented 7 new circuit members of $\mathcal{C}_3$ and demonstrated how these circuits have different properties than members of $\mathcal{C}_2$.

Moreover, two different IP formulations were presented for solving the same problem: finding a set of 3MOLRm or completing such an incomplete triple. The first, a 3-index formulation, is new and consists of three models that are linked with newly introduced constraints called orthogonality constraints. The second, is a highly symmetrical 5-index assignment formulation that in some recent research has been the base for analysing the OLS problem.

It is well known that IP for large number of n does not produce results in a reasonable amount of time. However, a natural next step could be to compare the performance of the two formulations for small values of n and relate results to their structural differences. This can also be considered in combination with the maximally lifted circuit inequalities presented in Chapter 5.

In addition to the IP formulations, this chapter presented CP formulations for finding Latin rectangles, sets of 3 MOLRm but also OLS. For all formulations, we applied the idea of combining independent models and linking them together into one model, to improve constraint propagation and consequently computational times. Initial computational experiments showed that this method is affecting, and in fact, inverse constraints compared to channelling constraints should be preferred for linking the independent models. As next steps, further work on exploring the power of CP can be done, in order to achieve the best computational results.

With this chapter, the work the on MOLR comes to an end. The next chapter considers variation of the Latin rectangles problem known as the multi-index bottleneck assignment problem.

Chapter 7

The multi-level bottleneck assignment problem (MBA)

Latin squares and OLS are well known for their applications in scheduling and timetabling [20], [23], [48]. As an example, consider a truck scheduling problem for a period of n days, each day having m different jobs to be carried out by m drivers. The goal is to derive a schedule where for each day, drivers are assigned to a job in such a way that no driver has more than one job in a day and does not repeat a job throughout the period of $n$days. This is a feasibility problem whose solution can easily be given by an m-row Latin rectangle of order n. There are many variations of this problem; one of which is to relax the last requirement to allow for the assignment of a driver to the same job more than once. The problem becomes an optimisation problem when a cost is assigned to each job. This cost can be interpreted as hours of travel or the size of load the truck is carrying. A natural objective would be to minimise the heaviest load. This variation of the simple scheduling problem is known as the multi-level bottleneck assignment problem (MBA) and is the focus of this chapter.

In Section 7.1 the MBA problem is formally defined taking a graph theoretical approach. Depending on whether a driver can be assigned to any job in a given day or to only a subset of jobs, the MBA problem is divided into three types: complete, mixed and arbitrary. If the problem is complete it means that the driver can be assigned to any job in a day, i.e. there are no restrictions; if arbitrary then there are restrictions for each day and mixed if for some days the problem is complete and for other days arbitrary. Furthermore, it is stated by Burkard et al. in [13], that the complexity status of the complete-MBA problem is open and the focus of Section 7.2 is to prove that it is $\mathcal{NP}$-hard.

Section 7.3 formulates the MBA problem both as an IP and a CP. For large instances of the problem both methods fail to provide a solution in reasonable time and therefore Section 7.4 focuses on deriving heuristics with a good approximation that solve the problem in polynomial time. More specifically, for MBA with $n = 3$ days (MBA3), three new heuristics are presented. For complete-MBA3 a $\frac{7}{6}$-approximation algorithm is given for a special case where job costs are high. For mixed-MBA3 a $\frac{3}{2}$-approximation algorithm is presented and it is the first time a heuristic is shown for the case of one arbitrary edge-set and one complete edge-set of the MBA problem. Lastly, a simple 2-approximation algorithm is shown for arbitrary-MBA3 and it is proven that no polynomial time algorithm can improve upon the factor of 2, unless $\mathcal{P}=\mathcal{NP}$.

7.1 Problem description

We describe the m-dimensional bottleneck assignment problem (MBA). Consider the two sets $I = 1, \ldots, m$ and $J = 1, \ldots, n$ where $i \in I$ and $j \in J$. Let $V_1, \ldots, V_m$ be m pairwise disjoint sets, each with cardinality n, and let $V = \bigcup_{i \in I} V_i$. Set V is the node-set of an m-partite graph that has a given set of arcs E of the following form: $E = \{(v, v') : \ v \in V_i, \ v' \in V_{i+1}, \ i \in I \backslash \{m\}\}$. One can notice that members of E only connect nodes of consecutive sets, i.e. connect a node from V_i with a node from V_{i+1}, where $i \in I \backslash \{m\}$. Moreover, every $v \in V$ has a weight denoted by $w(v) \in \mathbb{N}$. Note, that if there exists a node-set V_i with cardinality $d = |V_i|$ where $d < m$ and $m = \max_{i \in I} |V_i|$, then we can add $m - d$ dummy nodes with $w(v) = 0$ to obtain sets with equal number of nodes.

A feasible m-tuple, which we call a *duty*, is an ordered list of nodes $D_j(v_1, \ldots, v_m)$ such that $v_i \in V_i$ where $i \in I$, $j \in J$ and $(v_i, v_{i+1}) \in E$ for $i \in I \backslash \{m\}$. For a given $j \in J$, the cost of a duty equals $c(D_j) = \sum_{i \in I} w(v_i)$ where $v_i \in D_j, \ i \in I$. The goal is to find a partition of V into n duties $D_1, D_2, \ldots, D_n$ such that $\max_{j \in J} c(D_j)$ is minimum. Therefore a solution s to the MBA problem is a collection of n m-tuples and the cost of the solution equals $c(s) = \max_{j \in J} \{c(D_j) : \ s = \{D_1, D_2, \ldots, D_n\}\}$.

This problem was introduced by Carraresi and Gallo [15], motivated by an application in bus driver scheduling. The goal here is to design a balanced allocation of shifts to each driver for a certain period of time. In the context of this application, a set V_i corresponds to the shifts that need to be carried on day $i \in I$ and an edge $(v_i, v_{i+1}) \in E$ where $i \in I \backslash \{m\}$ indicates that it is possible to perform shift v_{i+1} directly after shift v_i. The weight of a shift $w(v)$ represents a measure of cost for example the length of the shift or its lateness. A duty D_j is a set of shifts, one from each day, to be carried out by a driver and clearly each shift can be assigned to only one driver. The cost of a duty is simply the load of a driver, and in order to achieve a balanced allocation of shifts we want to minimise the load of the driver that is worst off i.e. to minimise the maximum load.

Let E_i where $i \in I \backslash \{m\}$ denote the edge-set between node-sets V_i and V_{i+1}. If there exists an edge between every node of V_i and V_{i+1}, then E_i is called *complete* otherwise *arbitrary*. Hence, the following three cases arise,

1. *complete-MBA*: for all $i \in I \backslash \{m\}$, E_i is complete,
2. *mixed-MBA*: $\exists \, i \in I \backslash \{m\}$ such that E_i is complete and $\exists \, i' \in I \backslash \{m\}$ such that $E_{i'}$ is arbitrary,
3. *arbitrary-MBA*: for all $i \in I \backslash \{m\}$, E_i is arbitrary

An example of a complete-MBA as well as its optimal solution are presented below.

Example 7.1.1

Figure 7.1.1 illustrates an MBA problem for $m = 3$ days and $n = 4$ shifts therefore $I = \{1, \ldots, 3\}$ and $J = \{1, \ldots, 4\}$. The node-sets are $V_1 = \{v_{11}, v_{12}, v_{13}, v_{14}\}$, $V_2 = \{v_{21}, v_{22}, v_{23}, v_{24}\}$ and $V_3 = \{v_{31}, v_{32}, v_{33}, v_{34}\}$. Clearly $V = V_1 \cup V_2 \cup V_3$. For the shift weights of these days we have, $\{w(v_{1j}) : v_{1j} \in V_1, \text{ for all } j \in J\} = \{1, 2, 5, 3\}$, $\{w(v_{2j}) : v_{2j} \in V_2, \text{ for all } j \in J\} = \{10, 8, 2, 0\}$ and $\{w(v_{3j}) : v_{3j} \in V_3, \text{ for all } j \in J\} = \{2, 1, 6, 7\}$. All lines represent the edges of E and notice that all nodes of consecutive node-sets are connected, therefore E is complete.

An optimal solution to this example is a partition of V into 4 duties. Hence, $S = \{D_1, D_2, D_3, D_4\}$ where $D_1 = \{v_{11}, v_{21}, v_{32}\}$, $D_2 = \{v_{12}, v_{22}, v_{31}\}$, $D_3 = \{v_{13}, v_{24}, v_{33}\}$ and $D_4 = \{v_{14}, v_{23}, v_{34}\}$. Consequently, $c(D_1) = 1 + 10 + 1 = 12$, $c(D_2) = 2 + 8 + 2 = 12$, $c(D_3) = 5 + 0 + 6 = 11$ and $c(D_4) = 3 + 2 + 7 = 12$. The cost of the solution is $max\{12, 12, 11, 12\} = 12$.

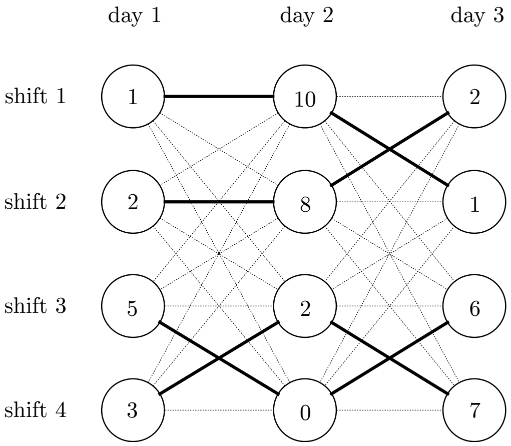


Figure 7.1.1: MBA example for $m = 3$, $n = 4$

7.2 Complexity results

In [15] the problem is described using a weight $w(v)$ on each edge leaving node v, when $v \in \bigcup_{i=1}^{m-2} V_i$, and an edge with weight $w(v) + w(v')$ for each edge $(v, v') \in V_{m-1} \times V_m$. They show that the problem is $\mathcal{NP}$-hard when m is part of the input by a reduction from Even-Odd Partition, and they leave as an open problem the complexity for a fixed m. MBA is also described in the recent book of Burkard et al. [13], where it is stated that the complexity of this problem is unresolved for each fixed $m \geq 3$ (pages 188-189); we will now settle this question.

Finding a feasible solution to the MBA is not a difficult problem. As observed in [15], there exists a feasible solution if edge-set E contains a perfect matching between each pair of sets (V_i, V_{i+1}), where $i \in I \backslash \{m\}$. However, finding an optimal solution is a far more difficult problem, even for $m = 3$. We will refer to the MBA problem for which $m = 3$ as MBA3.

Let S be the set of all feasible solutions (schedules). We denote with $c(s, D_j) = \sum_{i=1}^{3} w(v_i)$ the cost of solution s for some duty $D_j \in D$, where $j \in J$. The mini-max optimisation problem corresponding to MBA3 consists of finding a solution $s \in S$ with the best worst cost across all duties $D_j \in D$. This can be stated as,

$$min_{s \in S}\ max_{j \in J}\ c(s, D_j) \tag{7.2.1}$$

We will now show that MBA3 is $\mathcal{NP}$-hard. For this to be true we need to show that the decision version of our problem, as stated below, is $\mathcal{NP}$-complete.

Instance: A non-negative integer a and a 3-partite graph $G(V, E)$ with $V = V_1 \cup V_2 \cup V_3$ and $|V_1| = |V_2| = |V_3| = n$. We assume the edge-set is complete therefore $E := \{(v, v')|\ (v, v') \in (V_1 \times V_2) \cup (V_2 \times V_3)\}$ and for each $1 \leq i \leq 3$, $v_j \in V_i$ has a non-negative integer weight $w(v_j)$.

Question: Does there exist a solution $s \in S$ consisting of n duties such that the maximum load of a duty is no more than a?

The decision version of MBA3 is obviously in $\mathcal{NP}$, since given a solution we can easily verify its correctness by checking whether the sum of weights in each duty does not exceed a. To complete the proof, we will show that the Numerical 3-Dimensional Matching problem which is proven to be $\mathcal{NP}$-complete in [33] , reduces to the decision version of complete-MBA3.

The Numerical 3-Dimensional Matching problem (N3DM): has as input 3 sets of positive integers $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_n\}$, and $Z = \{z_1, \ldots, z_n\}$, and a positive integer bound a such that $x_i,\ y_i,\ z_i \leq a,\ \forall i = \{1, ..., n\}$ and also $\sum_{i=1}^{n}(x_i + y_i + z_i) = na$. The question is whether there exist n disjoint triples, each containing one element from each of the three sets, such that for each triple (x_i, y_j, z_k) we have $x_i + y_j + z_k = a$, where $i, j, k \in \{1, ..., n\}$.

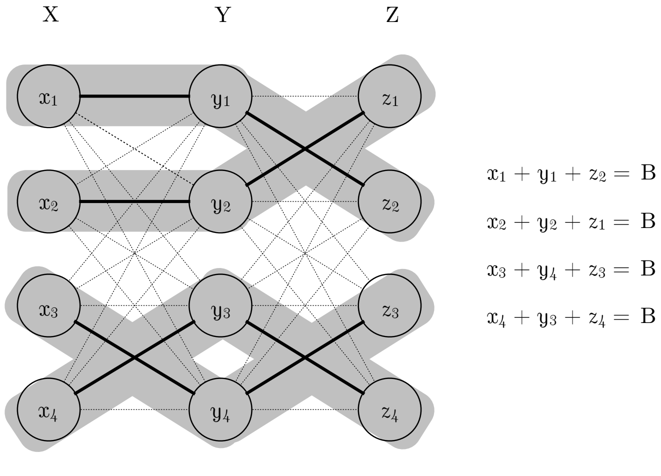


Figure 7.2.1: N3DM example for $m = 3$, $n = 4$

To construct, from a given instance of N3DM , an instance of MBA3, let the number of shifts equal n, and the number of days $m = 3$. We assume that the edge-set E is complete and the weight of a node $v_i \in V_1$ (or, in terms of [15], the weight of an edge leaving a node in V_1), equal x_i, thus $w(v_i) = x_i$ for each $v_i \in V_1$. Similarly, we have $w(v_i) := y_i$ for each $v_i \in V_2$, and $w(v_i) := z_i$ for each $v_i \in V_3$ (or, when phrased in terms of [15], the weight of an edge leaving node $v_i \in V_2$, going to node $v_{i'} \in V_3$ equals $y_i + z_{i'}$, where $1 \leq i$ and $i' \leq n$.

There exists a solution consisting of n duties such that the maximum load of a duty is no more than a, if there exists an instance of N3DM. From the example illustrated in Figure 7.2.1 it is easy to see that a "Yes" instance of N3DM corresponds to a solution of our problem with cost of each duty equal to a. Clearly a "No" instance of N3DM corresponds to a solution to MBA3 with highest duty cost greater than a or corresponds to an infeasible instance of MBA3. We have now shown that MBA3 is at least as hard as N3DM and we have settled the question in [13] by proving that complete-MBA3 is $\mathcal{NP}$-hard. Therefore we conclude the following,

Theorem 7.2.1. *Complete-MBA for a fixed value of m is $\mathcal{NP}$-hard.*

7.3 Formulations for the MBA problem

Many exact and heuristic approaches are applied to several generalisations of MBA mainly motivated by the various applications such as crew rostering and manpower scheduling [9], [14], [15]. Although there exist many approaches to solve large instances of more general problems, MBA in its fundamental form is not computationally well understood. In this section we explore the structure of the problem and present IP and CP formulations.

7.3.1 An Integer Programming formulation

In the context of the bus driver scheduling application, let the disjoint sets I, J and K denote the set of days, shifts and duties respectively, where $I = \{1, ..., m\}, \quad J, K = \{1, ..., n\}$. Let the binary variable x_{ijk} take value 1 if on day i shift j is assigned to duty k and value 0 otherwise. Parameter w_{ij} denotes the weight of shift j on day i and cD_k denotes the cost of duty k. Complete-MBA can be formulated as follows,

$$Min\ Max_{k\in K} \qquad cD_k \tag{7.3.1}$$

$$\text{s.t.} \qquad \sum_{j\in J} x_{ijk} = 1, \qquad \forall\ i \in I,\ \ k \in K, \tag{7.3.2}$$

$$\sum_{k\in K} x_{ijk} = 1, \qquad \forall\ i \in I,\ \ j \in J, \tag{7.3.3}$$

$$\sum_{i\in I,\, j\in J} w_{ij}\, x_{ijk} - cD_k = 0, \qquad \forall\ k \in K, \tag{7.3.4}$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall\ i \in I,\ \ j \in J,\ \ k \in K \tag{7.3.5}$$

Constraint (7.3.3) ensures that in a day, each duty contains only one shift. Constraint (7.3.4) ensures that in a day, each shift is assigned to only one duty. Finally constraint (7.3.5) defines the cost of a duty.
A linearised IP formulation is obtained if the objective function (7.3.8) is replaced by (7.3.6) and constraint (7.3.7) is added,

$$Min \qquad cD_1 \tag{7.3.6}$$

$$cD_k \leq cD_1, \qquad \forall\ k \in K \tag{7.3.7}$$

7.3.2 A Constraint Programming formulation

CP models for the MBA problem have natural formulations using `alldiff` constraints. In this case, these constraints force a particular set of integer variables to take values within a given set. The strength of the `alldiff` constraint is an efficient propagation algorithm that is able to reduce the domain of each variable by exploiting matching theory properties.

As for the IP model the disjoint sets I, J and K denote the set of days, shifts and duties respectively, where $I = \{1, ..., m\},\ \ J, K = \{1, ..., n\}$. Let D_{ij} be a finite domain of integer array variables that gives the duty containing shift j on day i and similarly let S_{ik} give the shift allocated to duty k on day i. The domain of each D_{ij} is set K and the domain of each S_{ik} the set J. Additionally, let cD_k denote the cost of duty k and let w_{ji} denote the weight of shift $j \in J$ on day $i \in I$.

$$Min \qquad cD_1 \tag{7.3.8}$$

$$\text{s.t.} \quad \texttt{alldiff}(\{S_{ik}\} : k \in K), \qquad \forall\, i \in I \tag{7.3.9}$$

$$\sum_{i\in I} w_{S_{ik}i} - cD_k = 0, \qquad \forall\, k \in K \tag{7.3.10}$$

$$cD_k \leq cD_1, \qquad \forall\, k \in K \tag{7.3.11}$$

Constraint (7.3.9) ensures that in a day, each duty contains a different shift and (7.3.10) calculates the cost of the k^{th} duty by adding the weights of the shifts that are assigned to it over the period of m days. This is a simple CP model that will minimise the cost of the heaviest duty and produce a schedule of shifts assigned duties.

A widely used technique aiming to improve the efficiency of a CP model, is to add redundant constraints. For this case, these are constraints that are logically implied by others in the model and often improve constraint propagation, thus making the CP model computationally faster. We list three different sets of redundant constraints for the MBA problem.

$$\texttt{alldiff}(\{D_{ij}\} \,:\, j \in J), \qquad \forall\, i \in I \tag{7.3.12}$$

Constraint (7.3.12) ensures that in a day, each shift is assigned to a different duty. This is implied by constraint (7.3.9), fact which makes (7.3.12) redundant.
Remember that inverse constraints force two groups of decision variables to be in strict correspondence with each other. Constraint `inv`therefore ensures that in a day, the duty containing shift j, is duty k and the shift assigned to that duty k, is shift j.

$$\texttt{inv}(\{D_{ij}, S_{ik}\} \,:\, j \in J,\ k \in K), \qquad \forall\, i \in I \tag{7.3.13}$$

Also remember that channelling constraints relate variables of two mutually redundant models. Constraint (7.3.14) states that in a day, the shift assigned to the duty containing shift j, is shift j and constraint (7.3.15) states that in a day, the duty containing the shift assigned to duty k, is duty k.

$$S_{D_{IJ}i} = j, \qquad \forall\, i \in I,\ \ j \in J \tag{7.3.14}$$

$$D_{S_{IK}i} = k, \qquad \forall\, i \in I,\ \ k \in K \tag{7.3.15}$$

7.4 Approximation algorithms for the MBA3 problem

Ideally, we would like to have a polynomial time algorithm that finds an optimal solution to the MBA problem for all instances. It has been shown that this is not possible, as is the case for many discrete optimisation problems. Among others, a common approach is to relax the optimality requirement and construct algorithms that run in polynomial time and provide solutions that approximate the optimal one in terms of value. In this section, we focus on MBA3.

First, it is worth elaborating on related work for the complete case. For complete-MBA3 Hsu gives in [41] a $\frac{3}{2}$-approximation algorithm that runs in $O(n\log n)$, and a $\frac{4}{3}$-approximation algorithm that runs in $O(n^3\log n)$.

In summary, for a given $n \times 3$ weight matrix the first algorithm sorts, for each column, values in ascending order and then recursively places the largest weight of the matrix in the row with the smallest sum. The second algorithm, although similar to the first, applies the restriction that in every row there can appear at most 2 of the $2n$ largest weights of the matrix. Moreover, Coffman and Yannakakis in [17] give a $(\frac{3}{2} - \frac{1}{2n})$-approximation algorithm for complete-MBA that runs in $O(n^2m)$. They describe an algorithm called Row Sum which in summary, takes the $n \times m$ matrix of all shift weights and after placing the m largest weights of the matrix in separate rows, arranges all other elements in each column in ascending order. Next, it performs exchanges between weights (in the same column) appearing in the lowest and highest row sum in order to reduce the latter.

In this section, we first present a simple 2-approximation algorithm called Sequential Bottleneck (SB) for arbitrary-MBA3 and show that this approximation factor is the best possible. We in fact show, that no polynomial time algorithm can improve upon the factor of 2, unless P=NP. For mixed-MBA3, we present a $\frac{3}{2}$-approximation algorithm called Assign and Bottleneck (AB) and to our knowledge this is the first time a heuristic is shown for the case of one arbitrary edge-set and one complete edge-set for MBA3. Finally, for a special case of complete-MBA3 we describe a $\frac{7}{6}$-approximation algorithm called Heavy Weight (HW). For all three approximation algorithms we give examples to demonstrate their bounds are tight.

7.4.1 Sequential Bottleneck heuristic

For arbitrary-MBA3, we present the Sequential Bottleneck (SB) heuristic, which is a 2-approximation polynomial algorithm that runs in two stages. In summary, SB first computes a bottleneck matching M between node-sets V_1 and V_2 and then computes a bottleneck matching between M and V_3. As a reminder, E_1 (the edge-set between V_1 and V_2) and E_2 (the edge-set between V_2 and V_3) are arbitrary, therefore not every shift (node) of V_1 can be succeeded by every shift (node) in V_2 and the same holds for shifts in V_2 and V_3. More formally,

Stage 1: Let the binary variable $x_{v,v'}$ take value 1 if shift $v \in V_1$ is succeeded by shift $v' \in V_2$ and 0 otherwise. Parameter $w(v)$ denotes the weight of shift $v \in V_1$ and $w(v')$ denotes the weight of shift $v' \in V_2$. In this stage, SB solves the following integer program:

$$\begin{aligned}
min \qquad & max_{v \in V_1,\, v' \in V_2} \, (w(v) + w(v'))\, x_{v,v'} \\
\text{s.t.} \qquad & \sum_{v' :\, \{v,v'\} \in E_1} x_{v,v'} = 1 \qquad \forall \;\; v \;\in V_1 \\
& \sum_{v \,:\, \{v,v'\} \in E_1} x_{v,v'} = 1 \qquad \forall \;\; v' \in V_2 \\
& \qquad x_{v,v'} \in \{0,1\} \qquad \forall \;\; \{v,v'\} \;\in E_1.
\end{aligned}$$

Stage 2: Let us denote the resulting matching from Stage 1 as M, i.e., $M = \{(v,v')|\; x^*_{v,v'} = 1\}$, where $x^*_{v,v'}$ is the solution to the IP presented above and let $w(M)$ denote the the highest weight of this matching.

Now let $w(v') := w(v) + w(v'),\ \forall\,(v, v') \in M$ where $v \in V_1$ and let $w(v'')$ denote the weight of shift $v'' \in V_3$. Lastly, let the binary variable $x_{v',v''}$ take value 1 if shift $v' \in V_2$ is succeeded by shift $v'' \in V_3$ and 0 otherwise.

$$\begin{array}{lll} min & max_{v' \in V_2,\, v'' \in V_3}\, (w(v') + w(v''))\, x_{v',v''} & \\ \text{s.t.} & \sum_{v'':\, \{v',v''\} \in E_2} x_{v',v''} = 1 & \forall\ v'\ \in V_2 \\ & \sum_{v'\,:\, \{v',v''\} \in E_2} x_{v',v''} = 1 & \forall\ v'' \in V_2 \\ & x_{v',v''} \in \{0, 1\} & \forall\ \{v', v''\}\ \in E_2. \end{array}$$

The final maximum duty cost is given by the objective function of the second IP, and for a given instance I is denoted by *SB(I)*.

Theorem 7.4.1. *SB is a polynomial-time,* 2*-approximation algorithm for MBA*3*.*

Proof. Obviously, SB is a polynomial-time algorithm, since it amounts to solving two bottleneck assignment problems. Now consider the solution obtained by SB; let its cost be determined by the triple $(v, v', v'') \in V_1 \times V_2 \times V_3$. Then:

$$\text{SB(I)} = w(v) + w(v') + w(v'') \leq w(M) + \max_{v'' \in V_3} w(v'')$$

Further, it is easily seen that OPT $\geq w(M)$, and that OPT $\geq \max_{v'' \in V_3} w(v'')$, where OPT refers to the cost of an optimal solution. It follows that SB(I) $\leq$ 2OPT.
A worst-case example for arbitrary-MBA3 that demonstrates how the upper bound can be achieved, is shown in Figure 7.4.1; where SB(I)=2OPT.

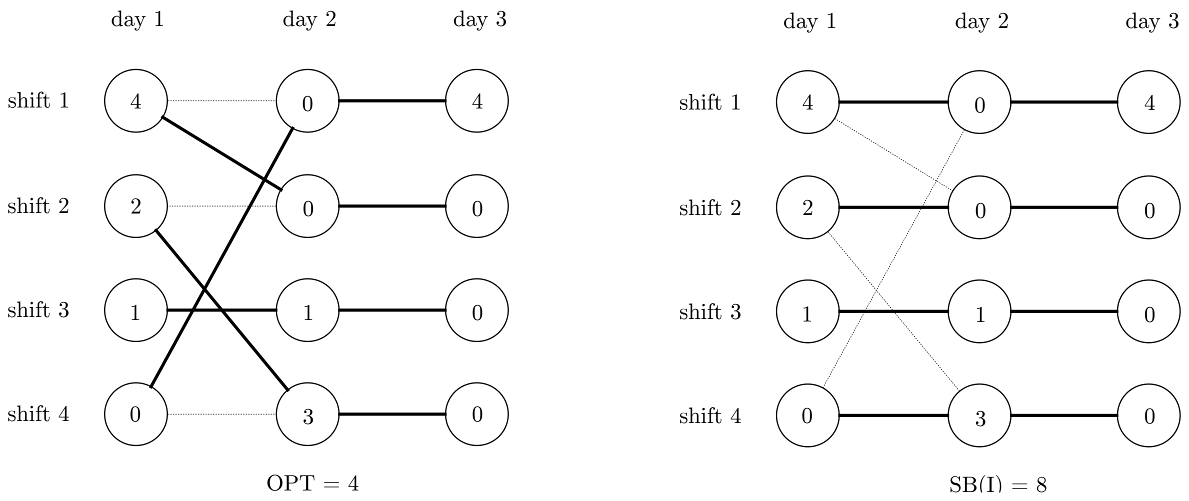


Figure 7.4.1: Example for worst-case performance of SB

□

Finally, notice that SB detects whether an instance has a feasible solution. Also, notice that in order to obtain a ratio of 2, any assignment in the second stage suffices. And although even more elaborate algorithms than SB can certainly be conceived, no polynomial time algorithm can improve upon the factor of 2 (unless P=NP), as we show next.

For the case of complete-MBA3, one could derive the optimal solutions for the two bottleneck problems without solving the suggested IP models but with use of a simple algorithm that involves sorting and summing of node-sets. More specifically, we refer to a $(2-\frac{1}{n})$-approximation algorithm that runs in $O(mn\log n)$ and was introduced by Hsu in [41]. It works as follows: Weights of nodes in V_1 are first sorted in ascending order and added to the weights of nodes in V_2 which are sorted in descending order. Weight sums of the resulting matching are then sorted in ascending order and added to weights of nodes in V_3 which are sorted in descending order. For $m > 3$ the process is repeated until all node-sets are processed.

We now continue to show that arbitrary-MBA3 cannot be approximated within a factor of 2 unless P=NP. This is done with use of a traditional technique: we will show that a “Yes” instance of 3-Dimensional Matching (3DM) corresponds to an instance of arbitrary -MBA3 with cost 1, whereas a “No” instance corresponds to an instance of our problem with cost 2 (or one that has no feasible solution). Then, a polynomial time approximation algorithm with a worst-case ratio strictly less than 2 would be able to distinguish the “Yes” instances of 3DM from the “No” instances and this would imply P=NP. We first describe 3DM.

The 3-Dimensional Matching problem (3DM): has as input 3 sets of positive integers $X = \{x_1, ..., x_q\}$, $Y = \{y_1, ..., y_q\}$, and $Z = \{z_1, ..., z_q\}$ and a subset $T \subseteq X \times Y \times Z$. The question is whether there exists a subset T' of T such that each element of $X \cup Y \cup Z$ is in exactly one triple of T'.

Let the number of triples be denoted by $|T| = p$. Further, let the number of triples in which element y_j occurs, be denoted by $\#occ(y_j)$, $j = \{1, ..., q\}$. Starting from an arbitrary instance of 3DM, we now build a corresponding instance of arbitrary-MBA3 by specifying V_i where $i = \{1, 2, 3\}$, E and the weights w as follows:

- For each triple in T, there is a node in V_2. We refer to these nodes as *triple* nodes.
- For each $x_i \in X$ where $i = \{1, ..., q\}$, there is a node in V_1. In addition, for each $y_j \in Y$, there are $\#occ(y_j) - 1$ nodes in V_1, where $j = \{1, ..., q\}$; for such a node in V_1 we say that this node *corresponds* to element y_j. These latter nodes will be referred to as the *dummy* nodes of V_1.
- For each $z_k \in Z$ where $k = \{1, ..., q\}$, there is a node in V_3. Further, we have $p - q$ additional nodes in V_3 which will be referred to as the *dummy* nodes of V_3.

Notice that this construction ensures that $|V_1| = |V_2| = |V_3| = p$. Let the nodes of V_1, V_2, and V_3 be denoted by $\{x'_1, ..., x'_p\}$, $\{t'_1, ..., t'_p\}$ and $\{z'_1, ..., z'_p\}$ respectively. Thus, $\{x'_1, ..., x'_q\}$ are the non-dummy nodes of V_1 and $\{x'_{q+1}, ..., x'_p\}$ are the dummy nodes of V_1; notice that each dummy node of V_1 corresponds to some element $y_i \in Y$. Further, triple nodes $\{t'_1, ..., t'_p\}$ simply correspond to the triples in T, while $\{z'_1, ..., z'_q\}$ are the non-dummy nodes of V_3, and $\{z'_{q+1}, ..., z'_p\}$ are the dummy nodes of V_3. The edge set E is defined as follows:

- There is an edge (x'_i, t'_j) if x_i is in the j-th triple in T, for $i = \{1, ..., q\}$ and $j = \{1, ..., p\}$.
- There is an edge (t'_j, z'_k) if z_k is in the j-th triple in T, for $k = \{1, ..., q\}$ and $j = \{1, ..., p\}$.
- There is an edge (t'_j, z'_k), for $j = \{1, ..., p\}$ and $k = \{q + 1, ..., p\}$.
- There is an edge (x'_i, t'_k) if element $y_j \in Y$, to which dummy node x'_i corresponds, is contained in the k-th triple of T.

To complete the description of our instance of arbitrary-MBA3, we assign a 0 weight to all nodes of V_1, V_2 and V_3 with the exception of the dummy nodes in V_1 the non-dummy nodes in V_3 which are assigned a weight of 1.

Lemma 7.4.2. *If the instance of 3DM is a YES-instance, the corresponding instance of MBA3 has cost 1. If the instance of 3DM is a NO-instance, the corresponding instance of MBA3 has cost 2, or is infeasible.*

Proof. Suppose that the instance of 3DM is a YES-instance. Then we construct a solution to the corresponding MBA3 instance as follows. First, we copy each of the q triples in T' to duties in our solution of MBA3 by selecting the corresponding triple node t'_j in V_2, together with the associated nondummy node x'_i from V_1 and nondummy node z'_k from V_3 (notice that the corresponding edges are in E). The resulting q duties contain all nondummy nodes in V_1 as well as all nondummy nodes in V_3. Further, we build duties containing the dummy nodes in V_1 by assigning each such node to the triple node in V_2 that contains element y_j corresponding to the dummy node in V_1. This is always possible, since the instance of 3DM is a YES-instance, and hence, for each $y_j \in Y$, exactly $\#occ(y_j) - 1$ nodes in V_2 remain. Since the edge set E contains any edge between a dummy node in V_3 and a node in V_2, we can extend these pairs to duties by assigning the dummy nodes in V_3 to these pairs. Observe that each resulting duty has cost 1.

A "No" instance for 3DM is such, either because some element of $X \cup Y \cup Z$ does not appear in T; or because even though all elements of $X \cup Y \cup Z$ appear in T (implying that T has at least q triples) in every subset T' of T where $|T'| = q$, at least one of the members of $X \cup Y \cup Z$ is repeated. For the first case, it is obvious that if an element of X or Z does not appear in T then the corresponding arbitrary-MBA3 problem is infeasible since an edge connecting the corresponding node of V_1 or V_3 with a triple node of V_2 does not exist. For the second case, we will assume that it is possible to construct a corresponding solution to arbitrary-MBA3 where duty cost is not 2 and prove that this is not possible. Let us consider any subset V_2' of the triple nodes such that $|V_2'| = q$ and let the repeated element in the corresponding triples belong to X. There exist q edges (t', z) where $t' \in V_2'$ and $z \in V_3$ connecting nodes in V_2' with q non-dummy nodes in V_3. Up to this point, the cost of these "partially" constructed duties is 1. Now each element in V_2' must be connected with a non-dummy node of V_1 which is not possible, since two nodes in V_2' are connected to the same node in V_1. Thus there will always exist a duty in the solution containing a dummy node of V_1 and a non-dummy node of V_3 and subsequently its duty cost will be 2. A similar argument holds if the repeated element belongs to Z. □

For clarity we present Example 7.4.1 showing a "Yes" 3DM instance that corresponds to a highest duty cost of 1 for arbitrary-3MBA. We also present Example 7.4.2 showing a "No" 3DM instance that corresponds to a highest duty cost of 2 for arbitrary-3MBA.

Example 7.4.1

Consider the 3DM instance $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2, y_3\}$, $Z = \{z_1, z_2, z_3\}$ and $T = \{(x_1, y_1, z_1),$ $(x_1, y_1, z_2), (x_2, y_2, z_1), (x_3, y_3, z_3)\}$. There exists a solution $T' = \{(x_1, y_1, z_2), (x_2, y_2, z_1),$ $(x_3, y_3, z_3)\}$. Notice that $T' \subset T$ and that all elements of $X \cup Y \cup Z$ appear in T'. Figure 7.4.2 presents the corresponding solution for arbitrary-MBA3. In this construction (and also for Example 7.4.2), shaded nodes have weight of 1 and all other weight of 0; double circled nodes are the dummy nodes of V_1 and V_3. Finally, the duties are $\{(x'_1, t'_2, z'_2), (x'_2, t'_3, z'_1), (x'_3, t'_4, z'_3), (x'_4, t'_1, z'_4)\}$ and notice that every duty has cost 1, as it contains exactly one shaded node.

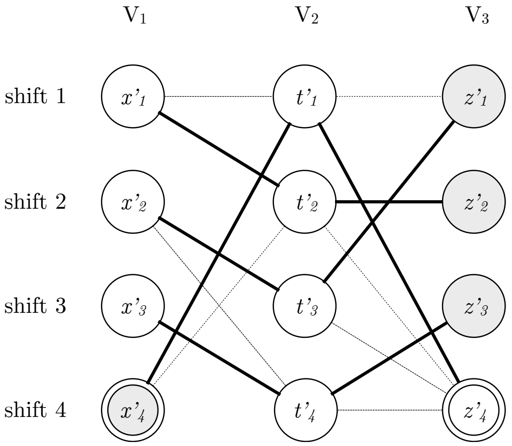


Figure 7.4.2: Solution with duty cost 1, corresponding to "Yes" instance of 3DM

Example 7.4.2
Consider the 3DM instance $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2, y_3\}$, $Z = \{z_1, z_2, z_3\}$ and $T = \{(x_1, y_1, z_1), (x_1, y_2, z_2), (x_2, y_1, z_3), (x_3, y_3, z_3)\}$. There does not exists a solution to this 3DM instance. Figure 7.4.3 presents a corresponding solution to this instanced for arbitrary-MBA3. Notice that in every attempt to construct 4 duties, the highest duty cost is always 2, as there is always a duty containing two shaded nodes. One such set if duties is $\{(x'_1, t'_2, z'_2), (x'_2, t'_3, z'_3), (x'_3, t'_4, z'_4), (x'_4, t'_1, z'_1)\}$.

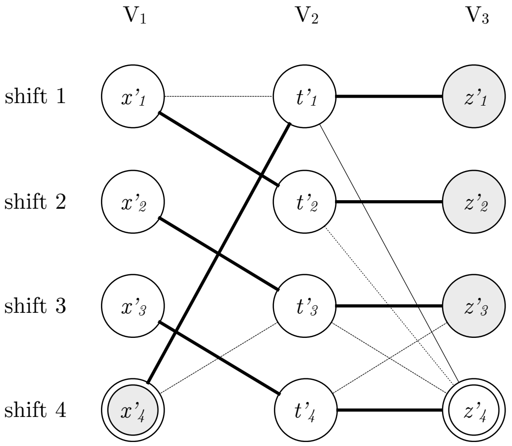


Figure 7.4.3: Solution with duty cost 2, corresponding to "No" instance of 3DM

The instances we have constructed have weights in $\{0, 1\}$, however the arguments hold when the degree of each node is bounded by some constant. Based on Lemma 7.4.2 we can now state:

Theorem 7.4.3. *There is no polynomial time algorithm for arbitrary-MBA*3 *that achieves an approximation guarantee of* $2 - \epsilon$, *for any* $\epsilon > 0$, *unless P=NP.*

7.4.2 Assign and Bottleneck heuristic

For mixed-MBA3, we present the Assign and Bottleneck (AB) heuristic, which is a $\frac{3}{2}$-approximation polynomial algorithm. We assume the edge set E_2 (i.e., the edge set between V_2 and V_3) is arbitrary and the edge set E_1 (i.e., the edge set between V_1 and V_2) is complete. Notice that, for complete-MBA3, both [41] and [17] present $\frac{3}{2}$-approximation algorithms. These heuristics, however, do not seem to be generalisable to mixed-MBA3 while preserving the approximation factor. The main part of AB consists of the procedure called $Core$-AB, which takes a guess of the optimal cost from the interval $[\left\lceil \frac{\sum_{v\in V} w(v)}{3n} \right\rceil, W]$, where W is the largest weight occurring in the input, and finds a feasible solution with the cost at most $\frac{3}{2}$ times the guess, if there exists one. AB outputs the feasible solution corresponding to the smallest guess. Let $V = V_1 \cup V_2 \cup V_3$ and let $w(v)$ denote the weight of node $v \in V$.

Algorithm 1 Heuristic-AB

```
{Input: MBA3 Instance}
Call Sequential Bottleneck(SB)
if no feasible solution found then
    STOP
    Output: No feasible solution
end if
$upper = W$
$lower = \left\lceil \frac{\sum_{v\in V} w(v)}{3n} \right\rceil$
repeat
    $OPT = \frac{upper+lower}{2}$
    Call $Core\text{-}AB(OPT)$
    if no feasible solution exists then
        $lower = OPT$
    else
        $upper = OPT$
    end if
until $upper = lower$
Call $Core\text{-}AB(upper)$
Output: Solution
```

Description of Core-AB: We assume that the value of the optimal solution, called OPT, is known. This is not a problem as $OPT \in [\left\lceil \frac{\sum_{v\in V} w(v)}{3n} \right\rceil, W]$. Without loss of generality, we further assume that the nodes in V_1, V_2, and V_3 are ordered in non-increasing order of their weights, with $V_1 = \{v_1, v_2, ..., v_n\}$ and $V_2 = \{v'_1, v'_2, ..., v'_n\}$. We say that a node $v \in V$ is *heavy* if $w(v) > \frac{OPT}{2}$, and we call v *non-heavy* otherwise. Let k_i be the number of heavy nodes in V_i, where $i = \{1, 2, 3\}$; notice that $k_1 + k_2 + k_3 \leq n$.

$Core$-AB has two stages. In the first stage, we solve an instance of the maximum weight perfect matching problem on a bipartite graph $G' = (V_2, V_3, E'_2)$. The edge set E'_2 is defined as follows: there is an edge $(v', v'') \in E'_2$ (with $v' \in V_2$ and $v'' \in V_3$), if (i) $(v', v'') \in E_2$, and (ii) $w(v') + w(v'') \leq OPT$. Further, we define the weight w' of an edge $(v', v'') \in E_2'$ as follows:

$$\begin{aligned} w'(v', v'') &= 1 \text{ if } w(v') + w(v'') \leq OPT/2 \\ &= 0 \text{ otherwise} \end{aligned}$$

In the second stage, we compute a bottleneck matching between the nodes from V_1 and the n pairs found in Stage 1; this gives us the solution of AB.

Theorem 7.4.4. *Heuristic AB is a polynomial-time $\frac{3}{2}$-approximation algorithm for mixed-MBA*3.

Proof. We begin by showing that AB is a polynomial-time algorithm. The first stage of $Core$-AB amounts to solving a maximum weight bipartite perfect matching problem, which can be done in polynomial time and $Core$-AB is run at most log$W$ times.

Let us next argue that heuristic AB finds a feasible solution whenever one exists. Indeed, assuming an optimal solution exists, any pair of nodes (v', v'') (with $v' \in V_2$ and $v'' \in V_3$) that are together in a duty in an optimal solution, are connected in G'. This is true, since these (v', v'') apparently satisfy $(v', v'') \in E_2$, and $w(v') + w(v'') \leq OPT$. Thus, a perfect matching exists in G', and since we solve an assignment problem in the first stage of AB, we find a perfect matching in the first stage. Then it follows easily that, since E_1 is complete by assumption, a feasible solution is found by AB.

We now prove the approximation guarantee. Again due to the existence of an optimal solution there exists a perfect matching in G' such that there are at least k_1 edges that are vertex disjoint whose weight in the original graph is bounded by $OPT/2$. Therefore, a maximum weight matching in G' will have weight at least k_1. Clearly in the second stage the k_1 heavy elements from V_1 can be bottleneck matched with these pairs. The total weight of each of these triples will be bounded by $\frac{3}{2}OPT$. Any other triple will consist of three non-heavy nodes and hence its weight is also bounded by $\frac{3}{2}OPT$. This proves the approximation factor.

A worst-case example for mixed-MBA3 that demonstrates how the upper bound can be achieved, is shown in Figure 7.4.4, where AB(I)=$\frac{3}{2}$OPT.

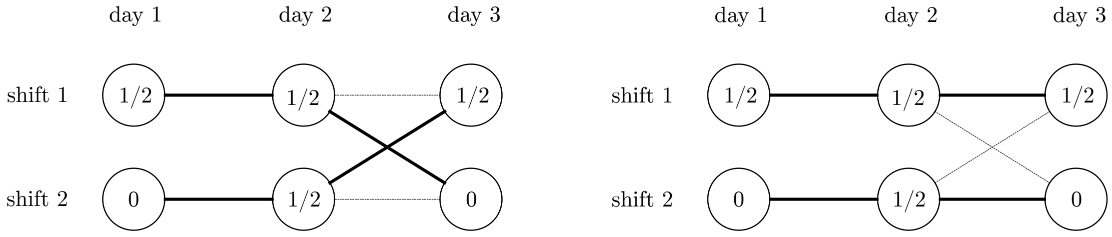


Figure 7.4.4: Example for worst-case performance of AB

□

7.4.3 Heavy Weight heuristic

In this section, we assume OPT, the optimal solution to an MBA3 problem is known and we scale the weights of all nodes $v \in V$ by dividing them by OPT. Moreover, let $V(v)$ denote the node-set V_j containing node $v \in V$, where $j \in \{1, 2, 3\}$ and note that $V_1 \cup V_2 \cup V_3 = V$.

Remark 7.4.5. *It holds that,*

(i) $\forall v \in V, \quad w(v) \leq 1$

(ii) There exist no more than $2n$ *nodes* $v \in V$ *with weight* $w(v) \geq \frac{1}{2}$

(iii) There exist at least n *nodes* $v \in V$ *with weight* $w(v) \leq \frac{1}{3}$

Proof. Statements (i) and (ii) are obvious. For (iii), let us assume that there exist only $n - 1$ nodes in $v \in V$ with weight $w(v) \leq \frac{1}{3}$. Then, in the optimal solution there exists a triple (p, q, r) where $w(p)$, $w(q)$, $w(r) > \frac{1}{3}$ and therefore $w(p) + w(q) + w(r) > 1$. This cannot be true. □

Theorem 7.4.6. *Let* V *be the node-set of a complete-MBA*3 *problem and let* A*,* B *be two disjoint* n*-subsets of* V*. Then the bipartite graph* $G(A, B, E)$ *has a perfect matching, where the edge-set* E *is defined as follows: there is an edge* $(a, b) \in E$ *(with* $a \in A$ *and* $b \in B$*) if* $V(a) \neq V(b)$*.*

Proof. According to Hall's theorem [37] on the sufficient condition for the existence of a perfect matching in bipartite graphs (see Theorem 1.4.1), we must show that for every subset S of A, the number of distinct nodes adjacent to some member of S is at least $|S|$. Let $V_j^A = A \cap V_j$, $\forall j \in \{1, 2, 3\}$ and let $V_j^C = C \cap V_j$, $\forall j \in \{1, 2, 3\}$. By definition of the edge-set E, node $a \in V_j^A$ is not adjacent to any of the members of V_j^C and is adjacent to all other $n - |V_j^C|$ nodes in C. Since $|V_j^C| \leq n - |V_j^A|$ we conclude that,

Statement 1*:* each node $a \in V_j^A$ is adjacent to at least $|V_j^A|$ nodes in C.

If $S = A$ or S contains at least one member of each V_j^A, where $j \in \{1, 2, 3\}$ then from Statement 1 it follows that S is adjacent to exactly n nodes $c \in C$. If S is a subset of V_j^A, where $j \in \{1, 2, 3\}$, then from

Statement 1 we know that S is adjacent to exactly $|V_j^A|$ members of C and $|S| \leq |V_j^A|$. Finally, if S is a subset of $V_1^A \cup V_2^A$, containing at least one node from each of the two sets, then each member of V_1^A is adjacent to $|V_2^C| + |V_3^C|$ members of C and each member of of V_2^A is adjacent to $|V_1^C| + |V_3^C|$ members of C. This gives us a total of n distinct nodes and it follows that S is adjacent to exactly n nodes and $|S| \leq n$ (note that $|V_1^C| + |V_2^C| + |V_3^C| = |C| = n$). A similar argument holds if S is a subset of $V_2^A \cup V_3^A$ or a subset of $V_1^A \cup V_3^A$. □

We now continue to describe the Heavy Weight (HW) heuristic.

Let nodes in V be arranged in decreasing order. The Heavy Weight (HW) heuristic, is a $\frac{7}{6}$-approximation polynomial algorithm for a special case: the $2n^{th}$ node has weight greater than $\frac{1}{3}$. Since the value of OPT is actually not known, steps 1 to 3 will be repeated as demonstrated for the heuristics of the previous section, until a perfect matching in Step 2 is found.

Step 1**:** Let nodes in V be arranged in decreasing order such that $w(v_i) \geq w(v_{i+1})$, $\forall v \in V$ where $i = \{1, ..., 3n-1\}$. Let k be the number of nodes $v \in V$ such that $w(v) > \frac{1}{2}$ and let A contain these first k nodes of V in addition to nodes $n+k+1$ to $2n$. Lastly, let B be a set containing the n lightest nodes of V.

Step 2**:** Find a perfect matching for the bipartite graph $G = (A, B, E)$ where E is defined as follows: there exists an edge $(a, b) \in E$ (with $a \in A$ and $b \in B$), if $w(a) + w(b) \leq \frac{2}{3}$.

Step 3**:** Extend the pair (a, b) to a triple (a, b, c) by adding a node $c \in V_j$ such that $V(a) \neq V(b) \neq V_j$.

Theorem 7.4.7. *HW is a polynomial-time $\frac{7}{6}$-approximation algorithm for complete-MBA3 for the case where the $2n^{th}$ node of V has weight greater than $\frac{1}{3}$.*

Proof. We fist list some observations:

(i) Since the $2n^{th}$ node of V has weight greater than $\frac{1}{3}$, then each triple in the optimal solution consists of two nodes with weight greater than $\frac{1}{3}$ and one node with weight less than $\frac{1}{3}$.

(ii) From (i) it follows that $\forall v \in V$, $w(v) < \frac{2}{3}$.

(iii) Assume (v_1, v_2, v_3) is a triple in the optimal solution of MBA3 with $w(v_1), w(v_2) > \frac{1}{3}$ and $w(v_3) < \frac{1}{3}$. Then it holds that $w(v_1) + w(v_3) \leq \frac{2}{3}$ and $w(v_2) + w(v_3) \leq \frac{2}{3}$.

We will first show that the bipartite graph in Step 2 always has a perfect matching. It is equivalent to show that the maximum weight perfect matching problem on a graph G' described below, always has a solution with objective function value equal to n. Therefore, consider an n-regular graph $G'(A, B, E')$ where A and B are as defined in Step 1 and edge set E' consists of n^2 edges (a, b), with $a \in A$ and $b \in B$. Further we define a weight w' of an edge $(a, b) \in E'$ as follows:

$$\begin{aligned} w'(a,b) &= 1 \text{ if } w(a) + w(b) \leq \frac{2}{3} \\ &= 0 \text{ otherwise} \end{aligned}$$

From Theorem 7.4.6 it follows that this problem always has a solution. It remains to show that the objective function value is always equal to n. First note that from observation (ii) we know that all nodes have weight less than $\frac{1}{3}$. Now assume the objective function value is equal to $n-1$. Then in the solution there exists a pair of nodes (a_1, b_1) for which $w'(a_1, b_1) > \frac{2}{3}$. If $w(a_1) > \frac{1}{2}$ we have a contradiction, since in the optimal solution of MBA3 no two of the heaviest k nodes of V with weight greater than $\frac{1}{2}$ can appear in the same triple. Therefore from observation (i) and (iii) it follows that for each of these nodes there exists a matching with a member of B and edge-weight equal to 1. If $w(a_1) \leq \frac{1}{2}$, then in the optimal solution of MBA3, node b_1 appears in a triple with node $v_1 \in V \backslash A \cup B$ and from observation (iii) we know that $w(v_1) + w(b_1) < \frac{2}{3}$. However, by definition of set A in Step 1, $w(v_1) \geq w(a_1)$ which results in a contradiction.

We have now shown the bipartite graph in Step 2 always has a perfect matching and it remains to prove the approximation guarantee. By definition of sets A and B in Step 1, it follows that $\forall v \in V \backslash A \cup B$, $w(v) \leq \frac{1}{2}$. Therefore for the resulting triples (a, b, c) after the completion of Step 3, it holds that $w(a) + w(b) \leq \frac{2}{3}$ and $w(c) < \frac{1}{2}$ and therefore $w(a) + w(b) + w(c) \leq \frac{7}{6}$.

A worst-case example for complete-MBA3 that demonstrates how the upper bound can be achieved, is shown in Figure 7.4.5, where $\epsilon > 0$ and HW(I)=$\frac{7}{6}$.

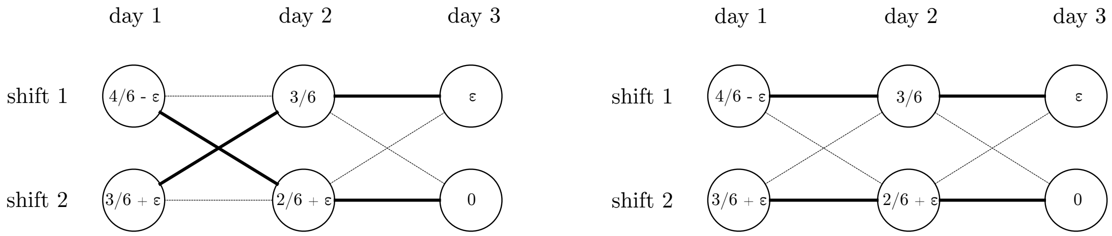


Figure 7.4.5: Example for worst-case performance of HW

□

7.5 Concluding remarks

This chapter introduced the MBA problem within the context of a driver's scheduling problem. It was shown that finding an MBA3 is $\mathcal{NP}$-hard, by reducing the problem to the N3DM problem. An IP formulation was presented, but since for a large number of shifts this does not give a solution in a reasonable amount of time, heuristic algorithms were introduced. Specifically for complete-MBA3, mixed-MBA3 and arbitrary-MBA3 a $\frac{7}{6}$, $\frac{3}{2}$ and 2-approximation algorithms respectively were given.

As a first step to find fast ways of reaching optimal solutions, a CP formulation was shown. However, computational experiments were not included and it would be interesting to test and improve how CP performs for this type of problem.

In further research, it would also be interesting to consider a similar problem to the MBA, where the goal is assigning duties as fairly as possible, in the sense that the cost of duties across all drivers must be as close as possible. We can call this the Fairness m-dimensional bottleneck assignment problem and refer to it as FMBA. One way to approach this problem, is to minimise the largest difference between two duty costs, amongst all pairs of duties. Reasonably, the greatest difference will appear between the heaviest (denoted by cD_{max}) and lightest (denoted by cD_{min}) duty cost. Therefore, the aim now is to find a partition of V into n duties $D_1, D_2, \ldots, D_n$ such that $cD_{max} - cD_{min}$ is minimum. This essentially means that we would like to simultaneously maximise the lightest duty cost and minimise the highest one. Further research can focus on showing complexity of the FMBA problem. It is clear from the example below that an optimal solution to the MBA problem is not optimal for FMBA. However, this raises the question as to whether the opposite holds.

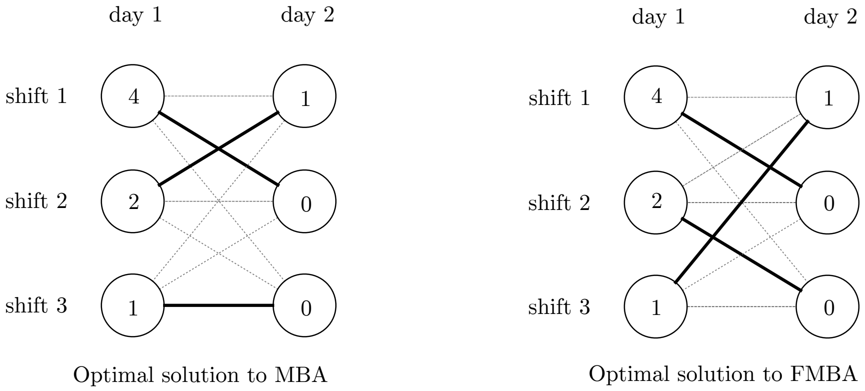


Figure 7.5.1: Optimal solution to MBA is not optimal solution to FMBA.

Appendix A: Completion of pink rectangles

<table>
<tr><th>j_1</th><th>j_2</th><th>j_3</th><th>... j_n</th></tr>
<tr><td>$\mathbf{1}$</td><td>2</td><td colspan="2">$K^1\backslash\{1,2\}$</td></tr>
<tr><td>K^1</td><td>$\mathbf{1}$</td><td>$\mathbf{2}$</td><td>$K^1\backslash\{1,2,k_1\}$</td></tr>
</table>

R^*_{1-}

<table>
<tr><th>j_1</th><th>j_2</th><th>... j_n</th></tr>
<tr><td>$\mathbf{1}$</td><td>k_2</td><td>$K^1\backslash\{1,k_2\}$</td></tr>
<tr><td>$\mathbf{2}$</td><td>$\mathbf{1}$</td><td>$K^1\backslash\{1,2\}$</td></tr>
</table>

R_{2-}

<table>
<tr><th>j_1</th><th>j_2</th><th>j_3</th><th>j_4</th><th>... j_n</th></tr>
<tr><td>$\mathbf{1}$</td><td>3</td><td>$\mathbf{2}$</td><td colspan="2">$K^1\backslash\{1,2,3\}$</td></tr>
<tr><td>2</td><td>$\mathbf{1}$</td><td>n</td><td>$\mathbf{3}$</td><td>$K^1\backslash\{1,2,3,n\}$</td></tr>
</table>

R^*_{6-}

<table>
<tr><th>j_1</th><th>j_2</th><th>j_3</th><th>... j_n</th></tr>
<tr><td>$\mathbf{1}$</td><td>$\mathbf{2}$</td><td colspan="2">$K^1\backslash\{1,2\}$</td></tr>
<tr><td>n</td><td>$\mathbf{1}$</td><td>$\mathbf{2}$</td><td>$K^1\backslash\{1,2,n\}$</td></tr>
</table>

R_{10-}

,

<table>
<tr><th>j_1</th><th>j_2</th><th>j_3</th><th>... j_n</th></tr>
<tr><td>$\mathbf{2}$</td><td>$\mathbf{1}$</td><td colspan="2">$K^1\backslash\{1,2\}$</td></tr>
<tr><td>n</td><td>$\mathbf{2}$</td><td>$\mathbf{1}$</td><td>$K^1\backslash\{1,2,n\}$</td></tr>
</table>

R_{10-}

<table>
<tr><th>j_1</th><th>j_2</th><th>... j_n</th></tr>
<tr><td>$\mathbf{1}$</td><td>$\mathbf{2}$</td><td>$K^1\backslash\{1,2\}$</td></tr>
<tr><td>$\mathbf{k_1}$</td><td>1</td><td>...</td></tr>
</table>

R^*_{25-}

<table>
<tr><th>j_1</th><th>j_2</th><th>... j_n</th></tr>
<tr><td>1</td><td>$\mathbf{k_1}$</td><td>...</td></tr>
<tr><td>$\mathbf{k_2}$</td><td>1</td><td>...</td></tr>
</table>

R_{28-}

Table A.1: Completion of type I pink rectangles

j_1	j_2	j_3	j_n
1	2	3	$K^1\backslash\{1,2,3\}$
3	**1**	**2**	$K^1\backslash\{1,2,3\}$

R^*_{4-}

,

j_1	j_2	j_3	j_n
1	2	3	$K^1\backslash\{1,2,3\}$
2	**3**	**1**	$K^1\backslash\{1,2,3\}$

R^*_{4-}

j_1	j_2	j_3	··· j_n
1	**2**	$K^1\backslash\{1,2\}$	
2	**1**	**3**	$K^1\backslash\{1,2,3\}$

R^*_{7-}

,

j_1	j_2	j_3	··· j_n
1	**2**	$K^1\backslash\{1,2\}$	
2	**3**	**1**	$K^1\backslash\{1,2,3\}$

R^*_{7-}

j_1	j_2	··· j_n
1	**2**	$K^1\backslash\{1,2\}$
3	**1**	$K^1\backslash\{1,3\}$

R_{8-}

,

j_1	j_2	··· j_n
2	**1**	$K^1\backslash\{1,2\}$
1	**3**	$K^1\backslash\{1,3\}$

R_{8-}

j_1	j_2	j_3	··· j_n
1	**2**	k_1	$K^1\backslash\{1,2,k_1\}$
2	**1**	**3**	$K^1\backslash\{1,2,3\}$

R_{15-}

,

j_1	j_2	j_3	··· j_n
1	**2**	k_1	$K^1\backslash\{1,2,k_1\}$
3	**1**	**2**	$K^1\backslash\{1,2,3\}$

R_{15-}

j_1	j_2	j_3	··· j_n
2	**1**	k_1	$K^1\backslash\{1,2,k_1\}$
1	**2**	**3**	$K^1\backslash\{1,2,3\}$

R_{15-}

,

j_1	j_2	j_3	··· j_n
2	**1**	k_1	$K^1\backslash\{1,2,k_1\}$
3	**2**	**1**	$K^1\backslash\{1,2,3\}$

R_{15-}

j_1	j_2	j_3	··· j_n
1	**2**	3	$K^1\backslash\{1,2,3\}$
3	**1**	**2**	$K^1\backslash\{1,2,3\}$

R^*_{15-}

,

j_1	j_2	j_3	··· j_n
1	**2**	3	$K^1\backslash\{1,2,3\}$
2	**3**	**1**	$K^1\backslash\{1,2,3\}$

R^*_{15-}

j_1	j_2	j_3	··· j_n
2	**1**	3	$K^1\backslash\{1,2,3\}$
1	**3**	**2**	$K^1\backslash\{1,2,3\}$

R^*_{15-}

,

j_1	j_2	j_3	··· j_n
2	**1**	3	$K^1\backslash\{1,2,3\}$
3	**2**	**1**	$K^1\backslash\{1,2,3\}$

R^*_{15-}

j_1	j_2	j_3	j_4	··· j_n
1	**2**	**3**	$K^1\backslash\{1,2,3\}$	
2	**1**	n	**3**	$K^1\backslash\{1,2,3,n\}$

R_{23-}

,

j_1	j_2	j_3	j_4	··· j_n
1	**2**	**3**	$K^1\backslash\{1,2,3\}$	
2	**3**	n	**1**	$K^1\backslash\{1,2,3,n\}$

R_{23-}

j_1	j_2	j_3	j_4	··· j_n
1	**2**	**3**	$K^1\backslash\{1,2,3\}$	
3	**1**	n	**2**	$K^1\backslash\{1,2,3,n\}$

R_{23-}

,

Similar completion for R_{23-} if values
$1, 2, 3$ are placed in different cells in first row.

j_1	j_2	j_3	··· j_n
1	**2**	**3**	$K^1\backslash\{1,2,3\}$
2	**3**	**1**	$K^1\backslash\{1,2,3\}$

R_{24-}

,

j_1	j_2	j_3	··· j_n
1	**2**	**3**	$K^1\backslash\{1,2,3\}$
3	**1**	**2**	$K^1\backslash\{1,2,3\}$

R_{24-}

Similar completion for R_{24-} if values $1, 2, 3$ are placed in different cells in first row.

Table A.2: Completion of type II pink rectangles (Continued in Table 7.5)

j_1	j_2	j_3	... j_n
1	**2**	**3**	$K^1\backslash\{1,2,3\}$
2	1	...	

R^*_{27-}

,

j_1	j_2	j_3	... j_n
1	**3**	**2**	$K^1\backslash\{1,2,3\}$
2	1	...	

R^*_{27-}

j_1	j_2	j_3	... j_n
3	**2**	**1**	$K^1\backslash\{1,2,3\}$
2	1	...	

R^*_{27-}

Table A.3: Completion of type II pink rectangles

j_1	j_2	... j_n
1	2	$K^1\backslash\{1,2\}$
2	**1**	$K^1\backslash\{1,2\}$

R^*_{2-}

j_1	j_2	... j_n
1	**2**	$K^1\backslash\{1,2\}$
2	**1**	$K^1\backslash\{1,2\}$

R_{11-}

,

j_1	j_2	... j_n
2	**1**	$K^1\backslash\{1,2\}$
1	**2**	$K^1\backslash\{1,2\}$

R_{11-}

j_1	j_2	... j_n
1	**2**	$K^1\backslash\{1,2\}$
2	1	...

R^*_{26-}

j_1	j_2	... j_n
1	2	...
2	1	...

R_{29-}

Table A.4: Completion of type III pink rectangles

Appendix B: Lifted circuit inequalities

Lifted circuit inequalities of Class 3

Inequalities for $\mathcal{C}^1_{2,3}$ are presented in Chapter 5. Here we formulate and lift circuit inequalities for $\mathcal{C}^2_{2,3}$.

$$\mathcal{C}^2_{2,3} \; : \sum_{j\in J\setminus\{j_1,j_2\}} (x_{i_1 j\pi(j)}) + x_{i_2 j_1 k^1_2} + x_{i_2 j_2 k^1} + \quad y_{i_1 j_1 k^2_1} + y_{i_1 j_2 k^2_2} + y_{i_2 j_1 k^2_2} + y_{i_2 j_2 k^2_1} \leq n+2,$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2\} \subset J, \quad \{k^1_1, k^1_2, k^1_3\} \subset K^1, \quad \pi : J \setminus \{j_1, j_2\} \longrightarrow K^1 \setminus \{k^1_1, k^1_2, k^1_3\}$, $\{k^2_1, k^2_2\} \subset K^2$

Lifted circuit inequalities:

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1 j k^1}) + x_{i_2 j_1 k^1_2} + x_{i_2 j_2 k^1_1} + y_{i_1 j_1 k^2_1} + y_{i_1 j_2 k^2_2} + y_{i_2 j_1 k^2_2} + y_{i_2 j_2 k^2_1}$$
$$+ x_{i_2 j_1 k^1_1} + 2y_{i_1 j_2 k^2_1} + y_{i_1 j_1 k^2_2} \leq n+2,$$

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1 j k^1}) + x_{i_2 j_1 k^1_2} + x_{i_2 j_2 k^1_1} + y_{i_1 j_1 k^2_1} + y_{i_1 j_2 k^2_2} + y_{i_2 j_1 k^2_2} + y_{i_2 j_2 k^2_1}$$
$$+ x_{i_2 j_1 k^1_1} + 2y_{i_1 j_2 k^2_1} + y_{i_2 j_1 k^2_1} \leq n+2,$$

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1 j k^1}) + x_{i_2 j_1 k^1_2} + x_{i_2 j_2 k^1_1} + y_{i_1 j_1 k^2_1} + y_{i_1 j_2 k^2_2} + y_{i_2 j_1 k^2_2} + y_{i_2 j_2 k^2_1}$$
$$+ x_{i_2 j_1 k^1_1} + 2y_{i_1 j_2 k^2_1} + y_{i_2 j_2 k^2_2} \leq n+2,$$

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_2} + x_{i_2j_2k^1_1} + y_{i_1j_1k^2_1} + y_{i_1j_2k^2_2} + y_{i_2j_1k^2_2} + y_{i_2j_2k^2_1}$$

$$+ x_{i_2j_1k^1_1} + 2y_{i_1j_1k^2_2} + y_{i_1j_2k^2_1} \leq n+2,$$

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_2} + x_{i_2j_2k^1_1} + y_{i_1j_1k^2_1} + y_{i_1j_2k^2_2} + y_{i_2j_1k^2_2} + y_{i_2j_2k^2_1}$$

$$+ x_{i_2j_1k^1_1} + 2y_{i_1j_1k^2_2} + y_{i_2j_1k^2_1} \leq n+2,$$

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_2} + x_{i_2j_2k^1_1} + y_{i_1j_1k^2_1} + y_{i_1j_2k^2_2} + y_{i_2j_1k^2_2} + y_{i_2j_2k^2_1}$$

$$+ x_{i_2j_1k^1_1} + 2y_{i_1j_1k^2_2} + y_{i_2j_2k^2_2} \leq n+2,$$

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_2} + x_{i_2j_2k^1_1} + y_{i_1j_1k^2_1} + y_{i_1j_2k^2_2} + y_{i_2j_1k^2_2} + y_{i_2j_2k^2_1}$$

$$+ x_{i_2j_1k^1_1} + 2y_{i_2j_1k^2_1} + y_{i_1j_1k^2_2} \leq n+2,$$

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_2} + x_{i_2j_2k^1_1} + y_{i_1j_1k^2_1} + y_{i_1j_2k^2_2} + y_{i_2j_1k^2_2} + y_{i_2j_2k^2_1}$$

$$+ x_{i_2j_1k^1_1} + 2y_{i_2j_1k^2_1} + y_{i_1j_2k^2_1} \leq n+2,$$

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_2} + x_{i_2j_2k^1_1} + y_{i_1j_1k^2_1} + y_{i_1j_2k^2_2} + y_{i_2j_1k^2_2} + y_{i_2j_2k^2_1}$$

$$+ x_{i_2j_1k^1_1} + 2y_{i_2j_1k^2_1} + y_{i_2j_2k^2_2} \leq n+2,$$

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_2} + x_{i_2j_2k^1_1} + y_{i_1j_1k^2_1} + y_{i_1j_2k^2_2} + y_{i_2j_1k^2_2} + y_{i_2j_2k^2_1}$$
$$+\, x_{i_2j_1k^1_1} + 2y_{i_2j_2k^2_2} + y_{i_2j_1k^2_1} \leq n+2,$$

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_2} + x_{i_2j_2k^1_1} + y_{i_1j_1k^2_1} + y_{i_1j_2k^2_2} + y_{i_2j_1k^2_2} + y_{i_2j_2k^2_1}$$
$$+\, x_{i_2j_1k^1_1} + 2y_{i_2j_2k^2_2} + y_{i_1j_1k^2_2} \leq n+2,$$

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_2} + x_{i_2j_2k^1_1} + y_{i_1j_1k^2_1} + y_{i_1j_2k^2_2} + y_{i_2j_1k^2_2} + y_{i_2j_2k^2_1}$$
$$+\, x_{i_2j_1k^1_1} + 2y_{i_2j_2k^2_2} + y_{i_1j_2k^2_1} \leq n+2,$$

where $\{i_1,i_2\} \subseteq I, \quad \{j_1,j_2,j_3\} \subset J, \quad \{k^1_1,k^1_2,k^1_3\} \subset K^1, \quad \{k^2_1,k^2_2\} \subset K^2$

Lifted circuit inequalities of Class 2

Class 2 consists of nine types of circuits, $\mathcal{C}^1_{2,2}$, $\mathcal{C}^2_{2,2}$, $\mathcal{C}^3_{2,2}$, $\mathcal{C}^4_{2,2}$, $\mathcal{C}^5_{2,2}$, $\mathcal{C}^6_{2,2}$, $\mathcal{C}^7_{2,2}$, $\mathcal{C}^8_{2,2}$ and $\mathcal{C}^9_{2,2}$. The corresponding circuit inequalities for $\mathcal{C}^1_{2,2}$ are presented in Chapter 5. Here we formulate and lift circuit inequalities for the remaining six subclasses.

The corresponding circuit inequalities for $\mathcal{C}^2_{2,2}$ depicted in Table 4.2.33 are of the form:

$$\mathcal{C}^2_{2,2}\ : \quad \sum_{j\in J\setminus\{j_1,j_2\}} (x_{i_1j\pi_1(j)}) + x_{i_2j_1k^1_n} + \sum_{j\in J\setminus\{j_1,j_2,j_3\}} (x_{i_2j\pi_2(j)}) + y_{i_1j_1k^2} + y_{i_2j_2k^2} \leq 2n-3$$

where $\{i_1,i_2\} \subseteq I$, $\{j_1,j_2,j_3\} \subset J$, $\{k^1_1,k^1_2,k^1_n\} \subset K^1$, $k^2 \in K^2$, $\pi_1 : J \setminus \{j_1,j_2\} \longrightarrow K^1 \setminus \{k^1_1,k^1_2\}$, $\pi_2 : J \setminus \{j_1,j_2,j_3\} \longrightarrow K^1 \setminus \{k^1_1,k^1_2,k^1_n\}$, such that $\pi_1(j) \neq \pi_2(j)$ where $j \in J \setminus \{j_1,j_2,j_3\}$

Lifted circuit inequalities:

$$\sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_n} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_n\}}} (x_{i_2jk^1}) + y_{i_1j_1k^2} + y_{i_2j_2k^2} + y_{i_1j_2k^2} \leq 2n-3$$

and

$$\sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_n} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_n\}}} (x_{i_2jk^1}) + y_{i_1j_1k^2} + y_{i_2j_2k^2} + y_{i_2j_1k^2} \leq 2n-3$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2, j_3\} \subset J, \quad \{k^1_1, k^1_2, k^1_n\} \subset K^1, \quad k^2 \in K^2$

The corresponding circuit inequalities for $\mathcal{C}^3_{2,2}$ depicted in Table 4.2.34 are of the form:

$$\mathcal{C}^3_{2,2} \ : \quad \sum_{j\in J\setminus\{j_1,j_2\}} (x_{i_1j\pi_1(j)} + x_{i_2j\pi_2(j)}) + y_{i_1j_1k^2} + y_{i_2j_2k^2} \leq 2n-3$$

where $\{i_1, i_2\} \subseteq I$, $\{j_1, j_2\} \subset J$, $\{k^1_1, k^1_2\} \subset K^1$, $k^2 \in K^2$, $\pi_1, \pi_2\colon\ J \setminus \{j_1, j_2\} \longrightarrow K^1 \setminus \{k^1_1, k^1_2\}$, such that $\pi_1(j) \neq \pi_2(j)$, where $j \in J \setminus \{j_1, j_2\}$

Lifted circuit inequalities:

$$\sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2\}}} (x_{i_1jk^1} + x_{i_2jk^1}) + y_{i_1j_1k^2} + y_{i_2j_2k^2} + y_{i_1j_2k^2} \leq 2n-3$$

and

$$\sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2\}}} (x_{i_1jk^1} + x_{i_2jk^1}) + y_{i_1j_1k^2} + y_{i_2j_2k^2} + y_{i_2j_2k^2} \leq 2n-3$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2\} \subset J, \quad \{k^1_1, k^1_2\} \subset K^1, \quad k^2 \in K^2$

The corresponding circuit inequalities for $\mathcal{C}^4_{2,2}$ depicted in Table 4.2.35 are of the form:

$$\mathcal{C}^4_{2,2} \ : \qquad \sum_{j \in J \setminus \{j_1, j_2, j_3\}} (x_{i_1 j \pi_1(j)} + x_{i_2 j \pi_2(j)}) + y_{i_1 j_1 k_1^2} + y_{i_1 j_2 k_2^2} + y_{i_2 j_1 k_2^2} + y_{i_2 j_2 k_1^2} \leq 2n - 3$$

where $\{i_1, i_2\} \subseteq I$, $\{j_1, j_2, j_3\} \subset J$, $\{k_1^1, k_2^1, k_3^1\} \subset K^1$, $\{k_1^2, k_2^2\} \subset K^2$, π_1, π_2: $J \setminus \{j_1, j_2, j_3\} \longrightarrow K^1 \setminus \{k_1^1, k_2^1, k_3^1\}$, such that $\pi_1(j) \neq \pi_2(j)$, where $j \in J \setminus \{j_1, j_2, j_3\}$

Lifted circuit inequalities:

$$\begin{aligned} \sum_{\substack{j \in J \setminus \{j_1, j_2, j_3\} \\ k^1 \in K^1 \setminus \{k_1^1, k_2^1, k_3^1\}}} & (x_{i_1 j k^1} + x_{i_2 j k^1}) + y_{i_1 j_1 k_1^2} + y_{i_1 j_2 k_2^2} + y_{i_2 j_1 k_2^2} + y_{i_2 j_2 k_1^2} + 2y_{i_1 j_1 k_2^2} \\ & + y_{i_1 j_2 k_1^2} \leq 2n - 3 \end{aligned}$$

and

$$\begin{aligned} \sum_{\substack{j \in J \setminus \{j_1, j_2, j_3\} \\ k^1 \in K^1 \setminus \{k_1^1, k_2^1, k_3^1\}}} & (x_{i_1 j k^1} + x_{i_2 j k^1}) + y_{i_1 j_1 k_1^2} + y_{i_1 j_2 k_2^2} + y_{i_2 j_1 k_2^2} + y_{i_2 j_2 k_1^2} + 2y_{i_1 j_1 k_2^2} \\ & + y_{i_2 j_1 k_1^2} \leq 2n - 3 \end{aligned}$$

where $\{i_1, i_2\} \subseteq I$, $\{j_1, j_2, j_3\} \subset J$, $\{k_1^1, k_2^1, k_3^1\} \subset K^1$, $\{k_1^2, k_2^2\} \subset K^2$

The corresponding circuit inequalities for $\mathcal{C}^5_{2,2}$ depicted in Table 4.2.36 are of the form:

$$\begin{aligned} \mathcal{C}^5_{2,2} \ : \quad & \sum_{j \in J \setminus \{j_1, j_2, j_3\}} (x_{i_1 j \pi_1(j)}) + x_{i_2 j k_n^1} + \sum_{j \in J \setminus \{j_1, j_2, j_3, j_4\}} (x_{i_2 j \pi_2(j)}) + y_{i_1 j_1 k_1^2} + y_{i_1 j_2 k_2^2} + y_{i_2 j_1 k_2^2} \\ & + y_{i_2 j_2 k_1^2} \leq 2n - 3 \end{aligned}$$

where $\{i_1, i_2\} \subseteq I$, $\{j_1, j_2, j_3, j_4\} \subset J$, $\{k_1^1, k_2^1, k_3^1, k_n^1\} \subset K^1$, $\{k_1^2, k_2^2\} \subset K^2$, $\pi_1 : J \setminus \{j_1, j_2, j_3\} \longrightarrow K^1 \setminus \{k_1^1, k_2^1, k_3^1\}$, $\pi_2 : J \setminus \{j_1, j_2, j_3, j_4\} \longrightarrow K^1 \setminus \{k_1^1, k_2^1, k_3^1, k_n^1\}$, such that $\pi_1(j) \neq \pi_2(j)$, where $j \in J \setminus \{j_1, j_2, j_3, j_4\}$

Lifted circuit inequalities:

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1 j\pi_1(j)}) + x_{i_2 j k^1_n} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3,j_4\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3,k^1_n\}}} (x_{i_2 j\pi_2(j)}) + y_{i_1 j_1 k^2_1} + y_{i_1 j_2 k^2_2} + y_{i_2 j_1 k^2_2}$$
$$+ y_{i_2 j_2 k^2_1} + 2y_{i_1 j_1 k^2_2} + y_{i_1 j_2 k^2_1} \leq 2n-3$$

and

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1 j\pi_1(j)}) + x_{i_2 j k^1_n} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3,j_4\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3,k^1_n\}}} (x_{i_2 j\pi_2(j)}) + y_{i_1 j_1 k^2_1} + y_{i_1 j_2 k^2_2} + y_{i_2 j_1 k^2_2}$$
$$+ y_{i_2 j_2 k^2_1} + 2y_{i_1 j_1 k^2_2} + y_{i_2 j_1 k^2_1} \leq 2n-3$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2, j_3, j_4\} \subset J, \quad \{k^1_1, k^1_2, k^1_3, k^1_n\} \subset K^1, \quad \{k^2_1, k^2_2\} \subset K^2$

The corresponding circuit inequalities for $\mathcal{C}^6_{2,2}$ depicted in Table 4.2.37 are of the form:

$$\mathcal{C}^6_{2,2} \; : \quad \sum_{j\in J\setminus\{j_1,j_2\}} (x_{i_1 j\pi_1(j)}) + x_{i_2 j_2 k^1_1} + \sum_{j\in J\setminus\{j_1,j_2\}} (y_{i_1 j\pi_2(j)}) + y_{i_2 j_2 k^2_1} \leq 2n-3$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2\} \subset J, \quad \{k^1_1, k^1_2\} \subset K^1, \quad \{k^2_1, k^2_2\} \subset K^2 \quad \pi_1 : J \setminus \{j_1, j_2\} \longrightarrow K^1 \setminus \{k^1_1, k^1_2\}, \quad \pi_2 : J \setminus \{j_1, j_2\} \longrightarrow K^2 \setminus \{k^2_1, k^2_2\}$

Lifted circuit inequalities:

$$\sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2\}}} (x_{i_1 j k^1}) + x_{i_2 j_2 k^1_1} + \sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^2\in K^2\setminus\{k^2_1,k^2_2\}}} (y_{i_1 j\pi_2(j)}) + y_{i_2 j_2 k^2_1} \leq 2n-3$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2\} \subset J, \quad \{k^1_1, k^1_2\} \subset K^1, \quad \{k^2_1, k^2_2\} \subset K^2$

The corresponding circuit inequalities for $\mathcal{C}^7_{2,2}$ depicted in Table 4.2.38 are of the form:

$$\mathcal{C}^7_{2,2} \; : \quad \sum_{j\in J\setminus\{j_1,j_2\}} (x_{i_1 j\pi_1(j)}) + x_{i_2 j_1 k^1_2} + x_{i_2 j_2 k^1_1} + \sum_{j\in J\setminus\{j_1,j_2,j_3\}} (y_{i_1 j\pi_2(j)}) + y_{i_2 j_2 k^2_2} + y_{i_2 j_2 k^2_1} \leq 2(n-1)$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2, j_3\} \subset J, \quad \{k_1^1, k_2^1\} \subset K^1, \quad \{k_1^2, k_2^2, k_3^2\} \subset K^2 \quad \pi_1 : J \setminus \{j_1, j_2\} \longrightarrow K^1 \setminus \{k_1^1, k_1^1\}, \quad \pi_2 : J \setminus \{j_1, j_2, j_3\} \longrightarrow K^2 \setminus \{k_1^2, k_2^2, k_3^2\}$

Lifted circuit inequalities:

$$\sum_{\substack{j \in J \setminus \{j_1, j_2\} \\ k^1 \in K^1 \setminus \{k_1^1, k_2^1\}}} (x_{i_1 j k^1}) + x_{i_2 j_1 k_2^1} + x_{i_2 j_2 k_1^1} + x_{i_2 j_1 k_1^1} + \sum_{\substack{j \in J \setminus \{j_1, j_2, j_3\} \\ k^2 \in K^2 \setminus \{k_1^2, k_2^2, k_3^2\}}} (y_{i_1 j \pi_2(j)}) + y_{i_2 j_2 k_2^2} + y_{i_2 j_2 k_1^2}$$
$$+ y_{i_2 j_1 k_1^2} \leq 2(n-1)$$

and

$$\sum_{\substack{j \in J \setminus \{j_1, j_2\} \\ k^1 \in K^1 \setminus \{k_1^1, k_2^1\}}} (x_{i_1 j k^1}) + x_{i_2 j_1 k_2^1} + x_{i_2 j_2 k_1^1} + x_{i_2 j_1 k_1^1} + \sum_{\substack{j \in J \setminus \{j_1, j_2, j_3\} \\ k^2 \in K^2 \setminus \{k_1^2, k_2^2, k_3^2\}}} (y_{i_1 j \pi_2(j)}) + y_{i_2 j_2 k_2^2} + y_{i_2 j_2 k_1^2}$$
$$+ y_{i_2 j_2 k_2^2} \leq 2(n-1)$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2, j_3\} \subset J, \quad \{k_1^1, k_2^1\} \subset K^1, \quad \{k_1^2, k_2^2, k_3^2\} \subset K^2$

Lifted circuit inequalities of Class 1

Class 1 consists of six types of circuits, $\mathcal{C}_{2,1}^1$, $\mathcal{C}_{2,1}^2$, $\mathcal{C}_{2,1}^3$, $\mathcal{C}_{2,1}^4$, $\mathcal{C}_{2,1}^5$ and $\mathcal{C}_{2,1}^6$. The corresponding circuit inequalities for $\mathcal{C}_{2,1}^1$ are presented in Chapter 5. Here we formulate and lift circuit inequalities for the remaining subclasses.

The corresponding circuit inequalities for $\mathcal{C}_{2,1}^2$ depicted in Table 4.2.25 are of the form:

$$\mathcal{C}_{2,1}^2 \; : \quad \sum_{j \in J \setminus \{j_1, j_2\}} (x_{i_1 j \pi_1(j)}) + x_{i_2 j_1 k_n^1} + \sum_{j \in J \setminus \{j_1, j_2, j_3\}} (x_{i_2 j \pi_2(j)}) + \sum_{j \in J \setminus \{j_1, j_2\}} (y_{i_1 j \pi_3(j)})$$
$$+ y_{i_2 j_2 k_1^2} \leq 3(n-2)$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2, j_3\} \subset J, \quad \{k_1^1, k_2^1, k_n^1\} \subset K^1, \quad k^2 \in K^2, \quad \pi_1 : J \setminus \{j_1, j_2\} \longrightarrow K^1 \setminus \{k_1^1, k_2^1\}, \quad \pi_2 : J \setminus \{j_1, j_2, j_3\} \longrightarrow K^1 \setminus \{k_1^1, k_2^1, k_n^1\},$ such that $\pi_1(j) \neq \pi_2(j)$ where $j \in J \setminus \{j_1, j_2, j_3\}, \quad \pi_3 : J \setminus \{j_1, j_2\} \longrightarrow K^2 \setminus \{k_1^2, k_2^2\}$

Lifted circuit inequalities:

$$\sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2\}}} (x_{i_1jk^1}) + x_{i_2j_1k^1_n} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_n\}}} (x_{i_2jk^1}) + \sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^1_2\in K^2\setminus\{k^2_1,k^2_2\}}} (y_{i_1jk^2}) + y_{i_2j_2k^2_1}$$
$$+ y_{i_2j_2k^2_2} \leq 3(n-2)$$

where $\{i_1,i_2\} \subseteq I, \quad \{j_1,j_2,j_3\} \subset J, \quad \{k^1_1,k^1_2,k^1_n\} \subset K^1, \quad \{k^2_1,k^2_2\} \subset K^2$

The corresponding circuit inequalities for $\mathcal{C}^3_{2,1}$ depicted in Table 4.2.26 are of the form:

$$\mathcal{C}^3_{2,1} : \quad \sum_{j\in J\setminus\{j_1,j_2,j_3\}} (x_{i_1j\pi_1(j)}) + x_{i_2j_3k^1_n} + \sum_{j\in J\setminus\{j_1,j_2,j_3,j_4\}} (x_{i_2j\pi_2(j)}) + \sum_{j\in J\setminus\{j_1,j_2\}} (y_{i_1j\pi_3(j)}) + y_{i_2j_2k^2_1}$$
$$+ y_{i_2j_1k^2_2} \leq 3n-7$$

where $\{i_1,i_2\} \subseteq I$, $\{j_1,j_2,j_3,j_4\} \subset J$, $\{k^1_1,k^1_2,k^1_3,k^1_n\} \subset K^1$, $\{k^2_1,k^2_2\} \subset K^2$, $\pi_1 : J \setminus \{j_1,j_2,j_3\} \longrightarrow K^1 \setminus \{k^1_1,k^1_2,k^1_3\}$, $\pi_2 : J \setminus \{j_1,j_2,j_3,j_4\} \longrightarrow K^1 \setminus \{k^1_1,k^1_2,k^1_3,k^1_n\}$, such that $\pi_1(j) \neq \pi_2(j)$, where $j \in J \setminus \{j_1,j_2,j_3,j_4\}$, $\pi_3 : J \setminus \{j_1,j_2\} \longrightarrow K^2 \setminus \{k^2_1,k^2_2\}$

Lifted circuit inequalities:

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3\}}} (x_{i_1jk^1}) + x_{i_2j_3k^1_n} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3,j_4\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_3,k^1_n\}}} (x_{i_2jk^1}) + \sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^1_2\in K^2\setminus\{k^2_1,k^2_2\}}} (y_{i_1jk^2}) + y_{i_2j_1k^2_2}$$
$$+ y_{i_2j_2k^2_1} + y_{i_2j_1k^2_1} \leq 3n-7$$

where $\{i_1,i_2\} \subseteq I, \quad \{j_1,j_2,j_3,j_4\} \subset J, \quad \{k^1_1,k^1_2,k^1_3,k^1_n\} \subset K^1, \quad \{k^2_1,k^2_2\} \subset K^2$

The corresponding circuit inequalities for $\mathcal{C}^4_{2,1}$ depicted in Table 4.2.27 are of the form:

$$\mathcal{C}^4_{2,1} : \quad \sum_{j\in J\setminus\{j_1,j_2,j_3\}} (x_{i_1j\pi_1(j)} + x_{i_2j\pi_2(j)}) + \sum_{j\in J\setminus\{j_1,j_2\}} (y_{i_1j\pi_3(j)}) + y_{i_2j_2k^2_1} + y_{i_2j_1k^2_2} \leq 3n-7$$

where $\{i_1,i_2\} \subseteq I$, $\{j_1,j_2,j_3\} \subset J$, $\{k^1_1,k^1_2,k^1_3\} \subset K^1$, $\{k^2_1,k^2_2\} \subset K^2$, $\pi_1,\pi_2 : J \setminus \{j_1,j_2,j_3\} \longrightarrow K^1 \setminus \{k^1_1,k^1_2,k^1_3\}$, such that $\pi_1(j) \neq \pi_2(j)$, $\pi_3 : J \setminus \{j_1,j_2\} \longrightarrow K^2 \setminus \{k^2_1,k^2_2\}$

Lifted circuit inequalities:

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k_1^1,k_2^1,k_3^1\}}} (x_{i_1jk^1}+x_{i_2jk^1}) + \sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k_2^1\in K^2\setminus\{k_1^2,k_2^2\}}} (y_{i_1jk^2}) + y_{i_2j_1k_2^2} + y_{i_2j_2k_1^2} + y_{i_2j_1k_1^2} \leq 3n-7$$

where $\{i_1,i_2\}\subseteq I, \quad \{j_1,j_2,j_3\}\subset J, \quad \{k_1^1,k_2^1,k_3^1\}\subset K^1, \quad \{k_1^2,k_2^2\}\subset K^2$

The corresponding circuit inequalities for $\mathcal{C}_{2,1}^5$ depicted in Table 4.2.28 are of the form:

$$\mathcal{C}_{2,1}^5 \ : \quad \sum_{j\in J\setminus\{j_1,j_2\}} (x_{i_1j\pi_1(j)}+x_{i_2j\pi_2(j)}) + \sum_{j\in J\setminus\{j_1,j_2,j_3\}} (y_{i_1j\pi_3(j)}) + y_{i_2j_2k_1^2} + y_{i_2j_1k_2^2} \leq 3(n-2)$$

where $\{i_1,i_2\}\subseteq I$, $\{j_1,j_2,j_3\}\subset J$, $\{k_1^1,k_2^1\}\subset K^1$, $\{k_1^2,k_2^2,k_3^2\}\subset K^2$, $\pi_1,\pi_2 : J\setminus\{j_1,j_2\}\longrightarrow K^1\setminus\{k_1^1,k_2^1\}$, such that $\pi_1(j)\neq\pi_2(j)$, $\pi_3 : J\setminus\{j_1,j_2,j_3\}\longrightarrow K^2\setminus\{k_1^2,k_2^2,k_3^2\}$

Lifted circuit inequalities:

$$\sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^1\in K^1\setminus\{k_1^1,k_2^1\}}} (x_{i_1jk^1}+x_{i_2jk^1}) + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k_2^1\in K^2\setminus\{k_1^2,k_2^2,k_3^2\}}} (y_{i_1jk^2}) + y_{i_2j_1k_2^2} + y_{i_2j_2k_1^2} + y_{i_2j_1k_1^2} \leq 3(n-2)$$

where $\{i_1,i_2\}\subseteq I, \quad \{j_1,j_2,j_3\}\subset J, \quad \{k_1^1,k_2^1\}\subset K^1, \quad \{k_1^2,k_2^2,k_3^2\}\subset K^2$

The corresponding circuit inequalities for $\mathcal{C}_{2,1}^6$ depicted in Table 4.2.29 are of the form:

$$\mathcal{C}_{2,1}^6 \ : \quad \sum_{j\in J\setminus\{j_1,j_2\}} (x_{i_1j\pi_1(j)}+x_{i_2j\pi_2(j)}) + \sum_{j\in J\setminus\{j_1,j_2\}} (y_{i_1j\pi_3(j)}) + y_{i_2j_2k_1^2} + \leq 3(n-2)$$

where $\{i_1,i_2\}\subseteq I$, $\{j_1,j_2\}\subset J$, $\{k_1^1,k_2^1\}\subset K^1$, $\{k_1^2,k_2^2\}\subset K^2$, $\pi_1,\pi_2 : J\setminus\{j_1,j_2\}\longrightarrow K^1\setminus\{k_1^1,k_2^1\}$, such that $\pi_1(j)\neq\pi_2(j)$, $\pi_3 : J\setminus\{j_1,j_2,j_3\}\longrightarrow K^2\setminus\{k_1^2,k_2^2\}$

Lifted circuit inequalities:

$$\sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^1\in K^1\setminus\{k_1^1,k_2^1\}}} (x_{i_1jk^1}+x_{i_2jk^1}) + \sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k_2^1\in K^2\setminus\{k_1^2,k_2^2\}}} (y_{i_1jk^2}) + y_{i_2j_2k_1^2} \leq 3(n-2)$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2\} \subset J, \quad \{k_1^1, k_2^1\} \subset K^1, \quad \{k_1^2, k_2^2\} \subset K^2$

Lifted circuit inequalities of Class 0

Class 0 consists of five types of circuits, $\mathcal{C}_{2,0}^1$, $\mathcal{C}_{2,0}^2$, $\mathcal{C}_{2,0}^3$, $\mathcal{C}_{2,0}^4$ and $\mathcal{C}_{2,0}^5$. The corresponding circuit inequalities for $\mathcal{C}_{2,0}^1$ are presented in Chapter 5. Here we formulate and lift circuit inequalities for the remaining subclasses.

The corresponding circuit inequalities for $\mathcal{C}_{2,1}^2$ depicted in Table 4.2.17 are of the form:

$$\begin{aligned} \mathcal{C}_{2,0}^2 \ : \quad & x_{i_1 j_2 k_3^1} + \sum_{j \in J \setminus \{j_1, j_2, j_3\}} (x_{i_1 j \pi_1(j)}) + x_{i_2 j_1 k_2^1} + x_{i_2 j_3 k_n^1} + \sum_{j \in J \setminus \{j_1, j_2, j_3, j_4\}} (x_{i_2 j \pi_2(j)}) \\ & + \sum_{j \in J \setminus \{j_1, j_2\}} (y_{i_1 j \pi_3(j)}) + y_{i_2 j_1 k_n^2} + \sum_{j \in J \setminus \{j_1, j_2, j_3\}} (y_{i_2 j \pi_4(j)}) \leq 4n - 9 \end{aligned}$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2, j_3, j_4\} \subset J, \quad \{k_1^1, k_2^1, k_3^1, k_n^1\} \subset K^1, \quad \{k_1^2, k_2^2, k_n^2\} \subset K^2, \quad \pi_1 : J \setminus \{j_1, j_2, j_3\} \longrightarrow K^1 \setminus \{k_1^1, k_2^1, k_3^1\}, \quad \pi_2 : J \setminus \{j_1, j_2, j_3, j_4\} \longrightarrow K^1 \setminus \{k_1^1, k_2^1, k_3^1, k_n^1\},$ such that $\pi_1(j) \neq \pi_2(j),$ where $j \in J \setminus \{j_1, j_2, j_3, j_4\}, \quad \pi_3 : J \setminus \{j_1, j_2\} \longrightarrow K^2 \setminus \{k_1^2, k_2^2\}, \quad \pi_4 : J \setminus \{j_1, j_2, j_3\} \longrightarrow K^2 \setminus \{k_1^2, k_2^2, k_n^2\},$ such that $\pi_3(j) \neq \pi_4(j),$ where $j \in J \setminus \{j_1, j_2, j_3\}$

The maximally lifted circuit inequality shown below is the only one.

$$\begin{aligned} & x_{i_1 j_2 k_3^1} + \sum_{\substack{j \in J \setminus \{j_1, j_2, j_3\} \\ k^1 \in K^1 \setminus \{k_1^1, k_2^1, k_3^1\}}} (x_{i_1 j k^1}) + x_{i_2 j_1 k_2^1} + x_{i_2 j_3 k_n^1} + \sum_{\substack{j \in J \setminus \{j_1, j_2, j_3, j_4\} \\ k^1 \in K^1 \setminus \{k_1^1, k_2^1, k_3^1, k_n^1\}}} (x_{i_2 j k^1}) \\ & + \sum_{\substack{j \in J \setminus \{j_1, j_2\} \\ k^2 \in K^2 \setminus \{k_1^2, k_2^2\}}} (y_{i_1 j k^2}) + y_{i_2 j_1 k_n^2} + \sum_{\substack{j \in J \setminus \{j_1, j_2, j_3\} \\ k^2 \in K^2 \setminus \{k_1^2, k_2^2, k_n^2\}}} (y_{i_2 j k^2}) \leq 4n - 9 \end{aligned}$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2, j_3\} \subset J, \quad \{k_1^1, k_2^1, k_3^1, k_n^1\} \subset K^1, \quad \{k_1^2, k_2^2, k_n^2\} \subset K^2$

The corresponding circuit inequalities for $\mathcal{C}^3_{2,1}$ depicted in Table 4.2.18 are of the form:

$$\begin{aligned}\mathcal{C}^3_{2,0}\ :\quad & \sum_{j\in J\setminus\{j_1,j_2\}} (x_{i_1 j\pi_1(j)}) + x_{i_2 j_1 k^1_n} + \sum_{j\in J\setminus\{j_1,j_2,j_3\}} (x_{i_2 j\pi_2(j)}) \\ & + \sum_{j\in J\setminus\{j_1,j_2\}} (y_{i_1 j\pi_3(j)}) + y_{i_2 j_1 k^2_n} + \sum_{j\in J\setminus\{j_1,j_2,j_3\}} (y_{i_2 j\pi_4(j)}) \leq 4n-1\end{aligned}$$

where $\{i_1,i_2\} \subseteq I$, $\{j_1,j_2,j_3\} \subset J$, $\{k^1_1,k^1_2,k^1_n\} \subset K^1$, $\{k^2_1,k^2_2,k^2_n\} \subset K^2$, $\pi_1 : J \setminus \{j_1,j_2\} \longrightarrow K^1 \setminus \{k^1_1,k^1_2\}$, $\pi_2 : J \setminus \{j_1,j_2,j_3\} \longrightarrow K^1 \setminus \{k^1_1,k^1_2,k^1_n\}$, such that $\pi_1(j) \neq \pi_2(j)$, where $j \in J \setminus \{j_1,j_2,j_3\}$, $\pi_3 : J \setminus \{j_1,j_2\} \longrightarrow K^2 \setminus \{k^2_1,k^2_2\}$, $\pi_4 : J \setminus \{j_1,j_2,j_3\} \longrightarrow K^2 \setminus \{k^2_1,k^2_2,k^2_n\}$, such that $\pi_3(j) \neq \pi_4(j)$, where $j \in J \setminus \{j_1,j_2,j_3\}$

The maximally lifted circuit inequality shown below is the only one.

$$\begin{aligned}& \sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2\}}} (x_{i_1 jk^1}) + x_{i_2 j_1 k^1_n} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k^1_1,k^1_2,k^1_n\}}} (x_{i_2 jk^1}) \\ & + \sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k^2\in K^2\setminus\{k^2_1,k^2_2\}}} (y_{i_1 jk^2}) + y_{i_2 j_1 k^2_n} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^2\in K^2\setminus\{k^2_1,k^2_2,k^2_n\}}} (y_{i_2 jk^2}) \leq 4n-1\end{aligned}$$

where $\{i_1,i_2\} \subseteq I$, $\{j_1,j_2,j_3\} \subset J$, $\{k^1_1,k^1_2,k^1_n\} \subset K^1$, $\{k^2_1,k^2_2,k^2_n\} \subset K^2$

The corresponding circuit inequalities for $\mathcal{C}^4_{2,0}$ depicted in Table 4.2.19 are of the form:

$$\begin{aligned}\mathcal{C}^4_{2,0}\ :\quad & \sum_{j\in J\setminus\{j_1,j_2,j_3\}} (x_{i_1 j\pi_1(j)}) + x_{i_2 j_3 k^1_n} + \sum_{j\in J\setminus\{j_1,j_2,j_3,j_4\}} (x_{i_2 j\pi_2(j)}) \\ & + \sum_{j\in J\setminus\{j_1,j_2\}} (y_{i_1 j\pi_3(j)} + y_{i_2 j\pi_4(j)}) \leq 4n-11\end{aligned}$$

where $\{i_1,i_2\} \subseteq I$, $\{j_1,j_2,j_3,j_4\} \subset J$, $\{k^1_1,k^1_2,k^1_3,k^1_n\} \subset K^1$, $\{k^2_1,k^2_2\} \subset K^2$, $\pi_1 : J \setminus \{j_1,j_2,j_3\} \longrightarrow K^1 \setminus \{k^1_1,k^1_2,k^1_3\}$, $\pi_2 : J \setminus \{j_1,j_2,j_3,j_4\} \longrightarrow K^1 \setminus \{k^1_1,k^1_2,k^1_3,k^1_n\}$, such that $\pi_1(j) \neq \pi_2(j)$, where $j \in J \setminus \{j_1,j_2,j_3,j_4\}$, $\pi_3,\pi_4 : J \setminus \{j_1,j_2\} \longrightarrow K^2 \setminus \{k^2_1,k^2_2\}$, such that $\pi_3(j) \neq \pi_4(j)$, where $j \in J \setminus \{j_1,j_2\}$

The maximally lifted circuit inequality shown below is the only one.

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k_1^1,k_2^1,k_3^1\}}} (x_{i_1jk^1}) + x_{i_2j_3k_n^1} + \sum_{\substack{j\in J\setminus\{j_1,j_2,j_3,j_4\}\\ k^1\in K^1\setminus\{k_1^1,k_2^1,k_3^1,k_n^1\}}} (x_{i_2jk^1})$$
$$+ \sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k_2^1\in K^2\setminus\{k_1^2,k_2^2\}}} (y_{i_1jk^2} + y_{i_2jk^2}) \leq 4n - 11$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2, j_3, j_4\} \subset J, \quad \{k_1^1, k_2^1, k_3^1, k_n^1\} \subset K^1, \quad \{k_1^2, k_2^2\} \subset K^2$

The corresponding circuit inequalities for $\mathcal{C}_{2,0}^5$ depicted in Table 4.2.20 are of the form:

$$\mathcal{C}_{2,0}^5 \ : \quad \sum_{j\in J\setminus\{j_1,j_2,j_3\}} (x_{i_1j\pi_1(j)} + x_{i_2j\pi_2(j)}) + \sum_{j\in J\setminus\{j_1,j_2\}} (y_{i_1j\pi_3(j)} + y_{i_2j\pi_4(j)}) \leq 4(n-3)$$

where $\{i_1, i_2\} \subseteq I$, $\{j_1, j_2, j_3\} \subset J$, $\{k_1^1, k_2^1, k_3^1\} \subset K^1$, $\{k_1^2, k_2^2\} \subset K^2$, $\pi_1, \pi_2 : J \setminus \{j_1, j_2, j_3\} \longrightarrow K^1 \setminus \{k_1^1, k_2^1, k_3^1\}$, such that $\pi_1(j) \neq \pi_2(j)$ $\pi_3, \pi_4 : J \setminus \{j_1, j_2\} \longrightarrow K^2 \setminus \{k_1^2, k_2^2\}$, such that $\pi_3(j) \neq \pi_4(j)$, where $j \in J \setminus \{j_1, j_2\}$

The maximally lifted circuit inequality shown below are the only one.

$$\sum_{\substack{j\in J\setminus\{j_1,j_2,j_3\}\\ k^1\in K^1\setminus\{k_1^1,k_2^1,k_3^1\}}} (x_{i_1jk^1} + x_{i_2jk^1}) + \sum_{\substack{j\in J\setminus\{j_1,j_2\}\\ k_2^1\in K^2\setminus\{k_1^2,k_2^2\}}} (y_{i_1jk^2} + y_{i_2jk^2}) \leq 4(n-3)$$

where $\{i_1, i_2\} \subseteq I, \quad \{j_1, j_2, j_3\} \subset J, \quad \{k_1^1, k_2^1, k_3^1\} \subset K^1, \quad \{k_1^2, k_2^2\} \subset K^2$

Appendix C: Circuits in $\mathcal{C}_3 \backslash \mathcal{C}_2$

In Section 6.1 we present seven new circuits, members of $\mathcal{C}_3 \backslash \mathcal{C}_2$ and provide a minimality and exclusion proof for the circuit depicted in Table 6.1.1. Here we provide such proofs for circuit of Tables 6.1.8 to 6.1.13.

Circuit 2 in $\mathcal{C}_3 \backslash \mathcal{C}_2$

j_1	j_2	j_3	
		1	
	1		

R_{1-}

j_1	j_2	j_3	
			$K^2\backslash\{1,2,3\}$
			$K^2\backslash\{1,2,3\}$

R_{2-}

j_1	j_2	j_3	
			$K^3\backslash\{1,2,3\}$
			$K^3\backslash\{1,2,3\}$

R_{3-}

Table C.1: Circuit in $\mathcal{C}_3 \backslash \mathcal{C}_2$

We first show that this set of 3 rectangles does not contain a member of $\mathcal{C}_1$. This is easy to see since all three rectangles are individually completable to Latin rectangles. We next need to show that they do not contain a member of $\mathcal{C}_2 \backslash \mathcal{C}_1$. For this, notice in Tables C.2 and C.3 that the combinations of completed Latin rectangles form orthogonal pairs.

j_1	j_2	j_3	
k_1	k_2	1	...
k_3	1	k_4	...

R_1

j_1	j_2	j_3	
1	2	3	$K^2\backslash\{1,2,3\}$
3	1	2	$K^2\backslash\{1,2,3\}$

R_2 or R_3

Table C.2: An OLR2

j_1	j_2	j_3	
1	2	3	$K^2\backslash\{1,2,3\}$
3	1	2	$K^2\backslash\{1,2,3\}$

R_2

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
2	3	1	$K^3\backslash\{1,2,3\}$

R_3

Table C.3: An OLR2

We now show that $R_{1-} \cup R_{2-} \cup R_{3-}$ contains a member of $\mathcal{C}_3 \backslash \mathcal{C}_2$ and therefore satisfies the exclusion property. For this, notice that up to equivalence, R_{2-} and R_{3-} can be filled in two ways as shown in Table C.4.

j_1	j_2	j_3	
		1	
	1		

R_{1-}

j_1	j_2	j_3	
1	2	3	$K^2\setminus\{1,2,3\}$
	1 or 3		$K^2\setminus\{1,2,3\}$

R_{2-}

j_1	j_2	j_3	
1	2	3	$K^3\setminus\{1,2,3\}$
	1 or 3		$K^3\setminus\{1,2,3\}$

R_{3-}

Table C.4: Completing rectangles of Table C.1

It is shown in Tables C.5 to C.7 that any selection of value in cell $(2, j_2)$ of R_{2-} and R_{3-} will lead to a repetition of a pair of values in $R_{1-} \cup R_{2-} \cup R_{3-}$.

j_1	j_2	j_3	
		1	
	1		

R_{1-}

j_1	j_2	j_3	
1	2	3	$K^2\setminus\{1,2,3\}$
	1		$K^2\setminus\{1,2,3\}$

R_{2-}

j_1	j_2	j_3	
1	2	3	$K^3\setminus\{1,2,3\}$
	1		$K^3\setminus\{1,2,3\}$

R_{3-}

Table C.5: Completing rectangles of Table C.1

j_1	j_2	j_3	
		1	
	1		

R_{1-}

j_1	j_2	j_3	
1	2	3	$K^2\setminus\{1,2,3\}$
	1		$K^2\setminus\{1,2,3\}$

R_{2-}

j_1	j_2	j_3	
1	2	3	$K^3\setminus\{1,2,3\}$
	3		$K^3\setminus\{1,2,3\}$

R_{3-}

Table C.6: Completing rectangles of Table C.1

j_1	j_2	j_3	
		1	
	1		

R_{1-}

j_1	j_2	j_3	
1	2	3	$K^2\setminus\{1,2,3\}$
	3		$K^2\setminus\{1,2,3\}$

R_{2-}

j_1	j_2	j_3	
1	2	3	$K^3\setminus\{1,2,3\}$
	3		$K^3\setminus\{1,2,3\}$

R_{3-}

Table C.7: Completing rectangles of Table C.1

It now remains to establish that $R_{1-} \cup R_{2-} \cup R_{3-}$ is minimal. For this it suffices to show that if any cell is emptied then the triple can be completed to a set of 3 MOLR2 i.e. it becomes a basis of the associated IS. We give an example in Table C.8 by emptying cell $(1, j_3)$ and a similar example can be shown if any other cell of Table C.7 is emptied.

j_1	j_2	j_3	
k_1	k_2	k_3	...
k_4	1	k_5	...

R_1

j_1	j_2	j_3	
1	2	3	$K^2\setminus\{1,2,3\}$
3	1	2	$K^2\setminus\{1,2,3\}$

R_2

j_1	j_2	j_3	
1	2	3	$K^3\setminus\{1,2,3\}$
2	3	1	$K^3\setminus\{1,2,3\}$

R_3

Table C.8: A set of 3 MOLR2

Circuit 3 in $\mathcal{C}_3 \setminus \mathcal{C}_2$

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3	
			$K^2 \setminus \{1,2,3\}$
			$K^2 \setminus \{1,2,3\}$

R_{2-}

j_1	j_2	j_3		j_n
		$K^3 \setminus \{1,2,3\}$		
			$K^3 \setminus \{1,2,3\}$	

R_{3-}

Table C.9: Circuit in $\mathcal{C}_3 \setminus \mathcal{C}_2$

We first show that this set of 3 rectangles does not contain a member of $\mathcal{C}_1$. This is easy to see since all three rectangles are individually completable to Latin rectangles. We next need to show that they do not contain a member of $\mathcal{C}_2 \setminus \mathcal{C}_1$. For this, notice in Tables C.10 to C.12 that the combinations of completed Latin rectangles form orthogonal pairs.

j_1	j_2	j_3	
k_1	1	k_2	...
1	k_3	k_4	...

R_{1-}

j_1	j_2	j_3	
1	2	3	$K^2 \setminus \{1,2,3\}$
3	1	2	$K^2 \setminus \{1,2,3\}$

R_2

Table C.10: An OLR2

j_1	j_2	j_3	
k_1	1	k_2	...
1	k_3	k_4	...

R_{1-}

j_1	j_2	j_3		j_n
1	2	$K^3 \setminus \{1,2,3\}$		3
3	1	2	$K^3 \setminus \{1,2,3\}$	

R_{3-}

Table C.11: An OLR2

j_1	j_2	j_3	
1	2	3	$K^2 \setminus \{1,2,3\}$
2	3	1	$K^2 \setminus \{1,2,3\}$

R_2

j_1	j_2	j_3		j_n
1	2	$K^3 \setminus \{1,2,3\}$		3
3	1	2	$K^3 \setminus \{1,2,3\}$	

R_{3-}

Table C.12: An OLR2

We now show that $R_{1-} \cup R_{2-} \cup R_{3-}$ contains a member of $\mathcal{C}_3 \setminus \mathcal{C}_2$ and therefore satisfies the exclusion property. For this, notice that up to equivalence, R_{2-} can be filled in two ways as shown in Table C.13.

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3	
1	2	3	$K^2 \setminus \{1,2,3\}$
	1 or 3		$K^2 \setminus \{1,2,3\}$

R_{2-}

j_1	j_2	j_3		j_n
		$K^3 \setminus \{1,2,3\}$		
			$K^3 \setminus \{1,2,3\}$	

R_{3-}

Table C.13: Completing rectangles of Table C.9

It is shown in Tables C.14 to C.16 that any selection of value in cell $(2, j_2)$ of R_{2-} will lead to a repetition of a pair of values in $R_{1-} \cup R_{2-} \cup R_{3-}$.

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3	
1	2	3	$K^2\setminus\{1,2,3\}$
2	3	1	$K^2\setminus\{1,2,3\}$

R_2

j_1	j_2	j_3		j_n
1	2	$K^3\setminus\{1,2,3\}$		3
2	3	1	$K^3\setminus\{1,2,3\}$	

R_3

Table C.14: Completing rectangles of Table C.9

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3	
1	2	3	$K^2\setminus\{1,2,3\}$
3	1	2	$K^2\setminus\{1,2,3\}$

R_{2-}

j_1	j_2	j_3		j_n
1	2	$K^3\setminus\{1,2,3\}$		3
2	3	1	$K^3\setminus\{1,2,3\}$	

R_3

Table C.15: Completing rectangles of Table C.9

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3	
1	2	3	$K^2\setminus\{1,2,3\}$
3	1	2	$K^2\setminus\{1,2,3\}$

R_{2-}

j_1	j_2	j_3		j_n
1	2	$K^3\setminus\{1,2,3\}$		3
3	1	2	$K^3\setminus\{1,2,3\}$	

R_{3-}

Table C.16: Completing rectangles of Table C.9

It now remains to establish that $R_{1-} \cup R_{2-} \cup R_{3-}$ is minimal. For this it suffices to show that if any cell is emptied then the triple can be completed to a set of 3 MOLR2 i.e. it becomes a basis of the associated IS. We give an example in Table C.17 by emptying cell $(1, j_2)$ of R_{1-} and a similar example can be shown if any other cell of Table C.9 is emptied.

j_1	j_2	j_3	
k_1	k_2	k_3	...
1	k_4	k_5	...

R_1

j_1	j_2	j_3	
1	2	3	$K^2\setminus\{1,2,3\}$
3	1	2	$K^2\setminus\{1,2,3\}$

R_2

j_1	j_2	j_3		j_n
1	2	$K^3\setminus\{1,2,3\}$		3
2	3	1	$K^3\setminus\{1,2,3\}$	

R_3

Table C.17: A set of 3 MOLR2

Circuit 4 in $\mathcal{C}_3 \setminus \mathcal{C}_2$

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3		j_n
		$K^2\setminus\{1,2,3\}$		
			$K^2\setminus\{1,2,3\}$	

R_{2-}

j_1	j_2	j_3		j_n
		$K^3\setminus\{1,2,3\}$		
			$K^3\setminus\{1,2,3\}$	

R_{3-}

Table C.18: Circuit in $\mathcal{C}_3 \setminus \mathcal{C}_2$

We first show that this set of 3 rectangles does not contain a member of $\mathcal{C}_1$. This is easy to see since all three rectangles are individually completable to Latin rectangles. We next need to show that they do not contain a member of $\mathcal{C}_2 \backslash \mathcal{C}_1$. Notice in Tables C.19 and C.20 that the combinations of completed Latin rectangles form orthogonal pairs.

j_1	j_2	j_3	
k_1	1	k_2	...
1	k_3	k_4	...

R_1

j_1	j_2	j_3		j_n
1	2	$K^3\backslash\{1,2,3\}$		3
3	1	2	$K^3\backslash\{1,2,3\}$	

R_2 or R_1

Table C.19: An OLR2

j_1	j_2	j_3		j_n
1	2	$K^3\backslash\{1,2,3\}$		3
3	1	2	$K^3\backslash\{1,2,3\}$	

R_2

j_1	j_2	j_3		j_n
1	2	$K^3\backslash\{1,2,3\}$		3
2	3	1	$K^3\backslash\{1,2,3\}$	

R_3

Table C.20: An OLR2

We now show that $R_{1-} \cup R_{2-} \cup R_{3-}$ contains a member of $\mathcal{C}_3 \backslash \mathcal{C}_2$ and therefore satisfies the exclusion property. For this, notice that up to equivalence, R_{2-} and R_{3-} can be filled in two ways as shown in Table C.21.

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3		j_n
1	2	$K^2\backslash\{1,2,3\}$		3
	1 or 3		$K^2\backslash\{1,2,3\}$	

R_{2-}

j_1	j_2	j_3		j_n
1	2	$K^2\backslash\{1,2,3\}$		3
	1 or 3		$K^3\backslash\{1,2,3\}$	

R_{3-}

Table C.21: Completing rectangles of Table C.18

It is shown in Tables C.22 to C.24 that any selection of value in cell $(2, j_2)$ of R_{2-} and R_{3-} will lead to at least one repetition of a pair of values in $R_{1-} \cup R_{2-} \cup R_{3-}$.

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3		j_n
1	2	$K^2\backslash\{1,2,3\}$		3
2	3	1	$K^2\backslash\{1,2,3\}$	

R_{2-}

j_1	j_2	j_3		j_n
1	2	$K^3\backslash\{1,2,3\}$		3
2	3	1	$K^3\backslash\{1,2,3\}$	

R_{3-}

Table C.22: Completing rectangles of Table C.18

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3		j_n
1	2	$K^2\backslash\{1,2,3\}$		3
2	3	1	$K^2\backslash\{1,2,3\}$	

R_{2-}

j_1	j_2	j_3		j_n
1	2	$K^3\backslash\{1,2,3\}$		3
3	1	2	$K^3\backslash\{1,2,3\}$	

R_{3-}

Table C.23: Completing rectangles of Table C.18

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3		j_n
1	2	$K^2\backslash\{1,2,3\}$		3
3	1	2	$K^2\backslash\{1,2,3\}$	

R_{2-}

j_1	j_2	j_3		j_n
1	2	$K^3\backslash\{1,2,3\}$		3
3	1	2	$K^3\backslash\{1,2,3\}$	

R_{3-}

Table C.24: Completing rectangles of Table C.18

It now remains to establish that $R_{1-} \cup R_{2-} \cup R_{3-}$ is minimal. For this it suffices to show that if any cell is emptied then the triple can be completed to a set of 3 MOLR2 i.e. it becomes a basis of the associated IS. We give an example in Table C.25 by emptying cell $(1, j_2)$ of R_{1-} and a similar example can be shown if any other cell of Table C.18 is emptied.

j_1	j_2	j_3	
k_1	k_2	k_3	...
1	k_4	k_5	...

R_1

j_1	j_2	j_3		j_n
1	2	$K^2\backslash\{1,2,3\}$		3
2	3	1	$K^2\backslash\{1,2,3\}$	

R_2

j_1	j_2	j_3		j_n
1	2	$K^3\backslash\{1,2,3\}$		3
3	1	2	$K^3\backslash\{1,2,3\}$	

R_3

Table C.25: A set of 3 MOLR2

Circuit 5 in $\mathcal{C}_3 \backslash \mathcal{C}_2$

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3	
1			
	1		

R_{2-}

j_1	j_2	j_3	
			$K^3\backslash\{1,2,3\}$
			$K^3\backslash\{1,2,3\}$

R_{3-}

Table C.26: Circuit in $\mathcal{C}_3 \backslash \mathcal{C}_2$

We first show that this set of 3 rectangles does not contain a member of $\mathcal{C}_1$. This is easy to see since all three rectangles are individually completable to Latin rectangles. We next need to show that they do not contain a member of $\mathcal{C}_2 \backslash \mathcal{C}_1$. Notice in Tables C.27 to C.29 that the combinations of completed Latin rectangles form orthogonal pairs.

j_1	j_2	j_3	
k_1	1	k_2	...
1	k_3	k_4	...

R_1

j_1	j_2	j_3	
1	k_5	k_6	...
k_7	1	k_8	..

R_2

Table C.27: An OLR2

j_1	j_2	j_3	
k_1	1	k_2	...
1	k_3	k_4	...

R_1

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
3	1	2	$K^3\backslash\{1,2,3\}$

R_3

Table C.28: An OLR2

j_1	j_2	j_3	
1	k_5	k_6	...
k_7	1	k_8	..

R_2

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
2	3	1	$K^3\backslash\{1,2,3\}$

R_3

Table C.29: An OLR2

We now show that $R_{1-} \cup R_{2-} \cup R_{3-}$ contains a member of $\mathcal{C}_3 \setminus \mathcal{C}_2$ and therefore satisfies the exclusion property. For this, notice that up to equivalence, R_{3-} can be filled in two ways as shown in Table C.30.

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3	
1			
	1		

R_{2-}

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
	1 or 3		$K^3\backslash\{1,2,3\}$

R_{3-}

Table C.30: Completing rectangles of Table C.26

It is shown in Tables C.31 and C.32 that any selection of value in cell $(2, j_2)$ of R_{3-} will lead to a repetition of a pair of values in $R_{1-} \cup R_{2-} \cup R_{3-}$.

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3	
1			
	1		

R_{2-}

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
2	3	1	$K^3\backslash\{1,2,3\}$

R_3

Table C.31: Completing rectangles of Table C.26

j_1	j_2	j_3	
	1		
1			

R_{1-}

j_1	j_2	j_3	
1			
	1		

R_{2-}

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
3	1	2	$K^3\backslash\{1,2,3\}$

R_{3-}

Table C.32: Completing rectangles of Table C.26

It now remains to establish that $R_{1-} \cup R_{2-} \cup R_{3-}$ is minimal. For this it suffices to show that if any cell is emptied then the triple can be completed to a set of 3 MOLR2 i.e. it becomes a basis of the associated IS. We give an example in Table C.33 by emptying cell $(1, j_2)$ and a similar example can be shown if any other cell of Table C.26 is emptied.

j_1	j_2	j_3	
k_1	k_2	k_3	...
1	k_4	k_5	...

R_1

j_1	j_2	j_3	
1	k_6	k_7	...
k_8	1	k_9	...

R_2

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
2	3	1	$K^3\backslash\{1,2,3\}$

R_3

Table C.33: A set of 3 MOLR2

Circuit 6 in $\mathcal{C}_3 \setminus \mathcal{C}_2$

j_1	j_2	j_3	
		1	
	1		

R_{1-}

j_1	j_2	j_3	
1			
	1		

R_{2-}

j_1	j_2	j_3	
			$K^3\backslash\{1,2,3\}$
	1		

R_{3-}

Table C.34: Circuit in $\mathcal{C}_3 \setminus \mathcal{C}_2$

This case is very similar to the previous one. The only difference now is that cell $(1, j_2)$ of R_{3-} can take two possible values: 2 or 3. The remaining elements of the proof are the same.

Circuit 7 in $\mathcal{C}_3 \setminus \mathcal{C}_2$

j_1	j_2	j_3	j_4	
				$K^1\backslash\{1,2,3,4\}$
				$K^1\backslash\{1,2,3,4\}$

R_{1-}

j_1	j_2	j_3	
			$K^2\backslash\{1,2,3\}$
			$K^2\backslash\{1,2,3\}$

R_{2-}

j_1	j_2	j_3	
			$K^3\backslash\{1,2,3\}$
			$K^3\backslash\{1,2,3\}$

R_{3-}

Table C.35: Circuit in $\mathcal{C}_3 \setminus \mathcal{C}_2$

We first show that this set of 3 rectangles does not contain a member of $\mathcal{C}_1$. This is easy to see since all three rectangles are individually completable to Latin rectangles. We next need to show that they do not contain a member of $\mathcal{C}_2 \setminus \mathcal{C}_1$. Notice in Tables C.36 and C.37 that the combinations of completed Latin rectangles form orthogonal pairs.

j_1	j_2	j_3	j_4	
1	2	3	4	$K^1\backslash\{1,2,3,4\}$
2	3	4	1	$K^1\backslash\{1,2,3,4\}$

R_1

j_1	j_2	j_3	
1	2	3	$K^2\backslash\{1,2,3\}$
3	1	2	$K^2\backslash\{1,2,3\}$

R_1 or R_3

Table C.36: An OLR2

j_1	j_2	j_3	
1	2	3	$K^2\backslash\{1,2,3\}$
3	2	1	$K^2\backslash\{1,2,3\}$

R_2

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
2	3	1	$K^3\backslash\{1,2,3\}$

R_3

Table C.37: An OLR2

We now show that $R_{1-} \cup R_{2-} \cup R_{3-}$ contains a member of $\mathcal{C}_3 \backslash\mathcal{C}_2$ and therefore satisfies the exclusion property. For this, notice that up to equivalence, R_{2-} and R_{3-} can be filled in two ways as shown in Table C.38.

j_1	j_2	j_3	j_4	
1	2	3	4	$K^1\backslash\{1,2,3,4\}$
2	3	4	1	$K^1\backslash\{1,2,3,4\}$

R_1

j_1	j_2	j_3	
1	2	3	$K^2\backslash\{1,2,3\}$
	1 or 3		$K^2\backslash\{1,2,3\}$

R_{2-}

j_1	j_2	j_3	
1	2	3	$K^3\backslash\{1,2,3\}$
	1 or 3		$K^3\backslash\{1,2,3\}$

R_{3-}

Table C.38: An OLR2

It is easy to see that a pair will always be repeated either in cells $(1, j_1)$, $(2, j_2)$ or $(1, j_3)$, $(2, j_2)$. Similarly to the previous cases, it can be shown that Table C.35 is completable to a set of 3 MOLR2 if any of the cells are emptied.

Bibliography

[1] M. H. Alsuwaiyel. *Algorithms: Design Techniques and Analysis*. World Scientific, Singapore, 1999.

[2] G. Appa, D. Magos, and I. Mourtos. A branch & cut algorithm for a four-index assignment problem. *Journal of Operational Research Society*, 55:298–307, 2004.

[3] G. Appa, D. Magos, and I. Mourtos. An lp-based proof for the non-existence of a pair of orthogonal latin squares of order 6. *Operations Research Letters*, 32:336–344, 2004.

[4] G. Appa, D. Magos, and I. Mourtos. The wheels of the orthogonal latin squares polytope: Classification and valid inequalities. *Journal of Combinatorial Optimization*, 10:365–389, 2005.

[5] G. Appa, D. Magos, and I. Mourtos. Searching for mutually orthogonal latin squares via integer and constraint programming. *European Journal of Operational Research*, 173:519–530, 2006.

[6] G. Appa, D. Magos, I. Mourtos, and J. C. M. Janssen. On the orthogonal latin squares polytope. *Discrete Mathematics*, 306(2):171–187, 2006.

[7] G. Appa, D. Magos, I. Mourtos, and L. Pitsoulis. *Handbook on Modeling for Discrete Optimization, International Series in Operations Research and Management Science*, chapter Formulations of the mutually orthogonal Latin squares problem. Springer, Berlin, 2006.

[8] K. R. Apt. *Principles of Constraint Programming*. Cambridge University Press, Cambridge, UK, 2003.

[9] J. Beasley and B. Cao. A dynamic programming based algorithm for the crew scheduling problem. *Computers and Operations Research*, 53:567–582, 1998.

[10] N. Beldiceanu and E. Contejean. Introducing global constraints in chip. *Journal of Mathematical and Computer Modelling*, 20:97–123, 1994.

[11] K. P. Bogart and J. Q. Longyear. Counting 3 by n latin rectangles. *Proceedings of the American Mathematical Society*, 54:463–467, 1976.

[12] J. A. Bondi and U. S. R. Murty. *Graph Theory*. Gruaduate Texts in Mathematics. Springer, 2008.

[13] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. SIAM, 2009.

[14] A. Caprara, P. Toth, D. Vigo, and M. Fischetti. Modeling and solving the crew rostering problem. *Operations Research*, 46:820–830, 1998.

[15] P. Carraresi and G. Gallo. A multi-level bottleneck assignment approach to the bus drivers rostering problem. *European Journal of Operational Research*, 16:163–173, 1984.

[16] B. M. W. Cheng, K. M. F. Choi, J. H. M. Lee, and J. C. K. Wu. Increasing constraint propagation by redundant modeling: an experience report. *CONSTRAINTS*, 4:167–192, 1999.

[17] E. Coffman and M. Yannakakis. Permuting elements within columns of a matrix in order to minimize maximum row sum. *Mathematics of Operations Research*, 9:384–390, 1984.

[18] C. J. Colbourn. The complexity of completing partial latin squares. *Discrete Applied Mathematics*, 8:25–30, 1984.

[19] G. Dantzig. *Linear programming and extensions*. Princeton University Press, 1963.

[20] D. de Werra. Some comments on a note about time-tabling. *Information Systems and Operational Research*, 16:90–92, 1978.

[21] E. Delisle. The search for a triple of mutually orthogonal latin squares of order ten: Looking through pairs of dimension thirty-five and less, Msc Thesis, University of Victoria, 2010.

[22] J. Dénes and A. D. Keedwell. *Latin Squares and their applications*. Academic Press, North Holland, 1975.

[23] J. Dénes and A. D. Keedwell. *Latin Squares: New developments in Theory and Applications*. Annals of Discrete Mathematics, 1991.

[24] R. Diestel. *Graph Theory*. Gruaduate Texts in Mathematics. Springer, 2000.

[25] I. B. Djordjevic and B. Vasic. Combinatorial constructions of optical orthogonal codes for ocdma systems. *IEEE Communications Letters*, 8(6):391–393, 2004.

[26] T. Dokka, A. Kouvela, and F. C. R. Spieksma. Approximating the multi-level bottleneck assignment problem. *Operations Research Letters*, 40:282–286, 2012.

[27] S. T. Dougherty. A coding theoretic solution to the 36 officer problem. *Designs, Codes and Cryptography*, 4:123–128, 1994.

[28] P. G. Doyle. The number of latin rectangles. *GNU FDL.*

[29] R. Euler. On the completability of incomplete latin squares. *European Journal of Combinatorics*, 31(2):535–552, 2010.

[30] R. Euler, R. E. Burkard, and R. Grommes. On latin squares and the facial structure of related polytopes. *Discrete Mathematics*, pages 155–181, 1986.

[31] R. Euler and P. Oleksik. When is an incomplete 3xn latin rectangle completable? *Discussiones Mathematicae Graph Theory*, 33:57–69, 2013.

[32] R. A. Fisher and F. Yates. The 6x6 latin squares. *Proceedings of the Cambridge Philosophical Society*, 30:492–507, 1934.

[33] M. Garey and D. Johnson. *Computers and intractability, a guide to the theory of NP-completeness*. W.H. Freeman and Company, New York, 1979.

[34] I. M. Gessel. Counting three-line latin rectangles. *Combinatoire Énumérative, Lecture Notes in Mathematics*, 1234:106–111, 1986.

[35] I. M. Gessel. Counting latin rectangles. *Bulletin of the American Mathematical Society*, 16:79–82, 1987.

[36] M. Hall. An existence theorem for latin squares. *Bulletin of the American Mathematical Society*, 51:387–388, 1945.

[37] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 10:26–30, 1935.

[38] B. Hayes-Roth and R. Korf, editors. *Proceedings of the National Conference on Artificial Intelligence*, volume AAAI94, Seattle, Washington, 1994.

[39] J. N. Hooker. *Ten Years of CPAIOR*, chapter Hybrid Modelling. Springer, (forthcoming).

[40] L. Howard. *Nets of Order 4m + 2: Linear Dependence and Dimensions of Codes*. PhD thesis, University of Victoria, 2009.

[41] W.-L. Hsu. Approximation algorithms for the assembly line balancing crew scheduling problem. *Mathematics of Operations Research*, 9:376–383, 1984.

[42] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27:119–216, 1988.

[43] A. D. Keedwell. Concerning the existence of triples of pairwise almost orthogonal 10x10 latin squares. *Ars Combinatoria*, 9:3–10, 1980.

[44] L. Khachiyan, E. Boros, K. Elbassioni, and V. Gurvich. A global parallel algorithm for the hypergraph transversal problem. *Inform. Information Processing Letters*, 101:148–155, 2007.

[45] Y. Kou, S. Lin, and M. Fossorier. Low-density parity-check codes based on finite geometries: A rediscovery and new results. *IEEE Transactions on Information Theory*, 47:2711– 2736, 2001.

[46] E. Kurtas, B. Vasic, and A. Kuznetsov. *Wiley Encyclopedia of Telecommunications*, chapter Design and Analysis of Low-Density Parity-Check Codes for Applications to Perpendicular Recording Channels. J. Wiley & Sons, New York, 2003.

[47] E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan. Generating all maximal independent sets: Np-hardness and polynomial-time algorithms. *SIAM Journal on Computing*, 9:558–565, 1980.

[48] C. F. Laywine and G. L. Mullen. *Discrete Mathematics using Latin Squares*. Wiley, New York, 1998.

[49] H. B. Mann. On orthogonal latin squares. *Bulletin of the American Mathematical Society*, 50(249-257), 1944.

[50] B. D. McKay, A. Meynert, and W. Myrvold. Small latin squares, quasigroups and loops. *Journal of Combinatorial Designs*, 15:98–119, 2007.

[51] B. D. McKay and I. M. Wanless. On the number of latin squares. *Annals of Combinatorics*, 9:335–326, 2005.

[52] M. W. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5:199–215, 1973.

[53] E. T. Parker. Computer investigation of orthogonal latin squares of order ten. *Proceedings of Symposia in Pure Mathematics*, 15:73–81, 1963.

[54] P. Popovski and H. Yomo. The antipackets can increase the achievable throughput of a wireless multi-hop network. *IEEE International Conference on Communications*, 2006.

[55] J. Riordan. Three-line latin rectangles. *American Mathematics Monthly*, 51:450–452, 1944.

[56] F. Rossi, P. Beek, and T. Walsh. *Handbook of Constraint Programming*. Elsevier, The Netherlands., 2006.

[57] H. J. Ryser. *Combinatorial Mathematics*. Mathematical Association of America, Washington, D. C., 1963.

[58] J. A. Salehi. Code division multiple-access techniques in optical fiber networks. i. fundamental principles. *IEEE Transactions on Communications*, 37:824–833, 1989.

[59] B. Smith. *Principles and Practice of Constraint Programming*, volume 2239 of *Lecture Notes in Computer Science*, chapter Dual Models of Permutation Problems, pages 615–619. 2001.

[60] D. R. Stinson. A short proof of the non-existence of a pair of orthogonal latin squares of order 6. *Journal of Combinatorial Theory, series A*, 36:373–376, 1984.

[61] F. Stork and M. Uetz. On the generation of circuits and minimal forbiden sets. *Mathematical Programming*, 102:185–203, 2005.

[62] G. Tarry. Le problème des 36 officiers. *C. R. Assoc. Franc. Av. Sci.*, 29:170–203, 1900.

[63] B. Vasic, E. M. Kurtas, and A. V. Kuznetsov. Ldpc codes based on mutually orthogonal latin rectangles and their application in ldpc codes based on mutually orthogonal latin rectangles and their application in perpendicular magnetic recording. *IEEE Transactions on Magnetics*, 38(5):2346–2348, 2002.

[64] T. M. Vijayvaradharaj and B. S. Rajan. Wireless network coding for mimo two-way relaying using latin rectangles. *Preprint, arXiv:1201.4477*, 2012.

[65] H. Williams and J. Wilson. Connections between integer linear programming and constraint logic programming - an overview and introduction to the cluster of articles. *INFORMS Journal on Computing*, 10:261–264, 1998.

[66] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2010.

[67] L. A. Wolsey. *Integer Programming*. J. Wiley & Sons, New York, 1998.

[68] K. Yamamoto. Euler squares and incomplete euler squares of even degrees. *Mem. Fac. Sci. Kyushu Univ, Series A 8*, pages 161–180, 1954.

[69] S. Zhang, S. C. Liew, and P. P. Lam. Hot topic: Physical-layer network coding. *The Annual International Conference on Mobile Computing and Networking*, pages 358–365, 2006.