Universidade do Minho
Escola de Engenharia

Isabel Cristina da Silva Lopes

Pattern sequencing models in cutting stock problems

Modelos para sequenciação de padrões em problemas de corte de stock

Fevereiro de 2011

Isabel Cristina da Silva Lopes **Pattern sequencing models in cutting stock problems**
Modelos para sequenciação de padrões em problemas de corte de stock

UMinho | 2011

POPH

QREN QUADRO DE REFERÊNCIA ESTRATÉGICO NACIONAL
PORTUGAL2007.2013

UNIÃO EUROPEIA
Fundo Social Europeu

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR Portugal

ESEIG POLITÉCNICO DO PORTO

Universidade do Minho
Escola de Engenharia

Isabel Cristina da Silva Lopes

Pattern sequencing models in cutting stock problems

Modelos para sequenciação de padrões em problemas de corte de stock

Programa Doutoral em Engenharia Industrial e de Sistemas

Trabalho efectuado sob a orientação do
Professor Doutor José Manuel Vasconcelos Valério de Carvalho

Fevereiro de 2011

Acknowledgements

This work was financially supported by the Portuguese Foundation for Science and Technology (FCT), by the financial program QREN-POPH-Tipology 4.1-Advanced Formation and European Social Fund and MCTES (Individual grant ref. SFRH/BD/32151/2006) between 2006 and 2009, and supported by ESEIG - Superior School of Industrial Studies and Management - Polytechnic Institute of Porto (PROTEC grant ref. SFRH/BD/49914/2009) between 2009 and 2010.

Este trabalho foi financiado pela FCT - Fundação para a Ciência e a Tecnologia, através do programa de financiamento QREN-POPH-Tipologia 4.1-Formação Avançada comparticipado pelo Fundo Social Europeu e por fundos do MCTES (Bolsa individual com a referência SFRH/BD/32151/2006) entre 2006 e 2009, e pela Escola Superior de Estudos Industriais e de Gestão do Instituto Politécnico do Porto (Bolsa PROTEC com a referência SFRH/BD/49914/2009) entre 2009 e 2010.

POPH QREN QUADRO DE REFERÊNCIA ESTRATÉGICO NACIONAL PORTUGAL 2007.2013 UNIÃO EUROPEIA Fundo Social Europeu

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR Portugal

ESEIG POLITÉCNICO DO PORTO

Pattern Sequencing Models in Cutting Stock Problems

Abstract

In this thesis, we address an optimization problem that appears in cutting stock operations research called the minimization of the maximum number of open stacks (MOSP) and we put forward a new integer programming formulation for the MOSP.

By associating the duration of each stack with an interval of time, it is possible to use the rich theory that exists in interval graphs in order to create a model based on the completion of a graph with edges. The structure of this type of graphs admits a linear ordering of the vertices that defines an ordering of the stacks, and consequently decides a sequence for the cutting patterns.

The polytope defined by this formulation is full-dimensional and the main inequalities in the model are proved to be facets. Additional inequalities are derived based on the properties of chordal graphs and comparability graphs. The maximum number of open stacks is related with the chromatic number of the solution graph; thus the formulation is strengthened by adding the representatives formulation for the vertex coloring problem.

The model is applied to the minimization of open stacks, and also to the minimum interval graph completion problem and other pattern sequencing problems such as the minimization of the order spread (MORP) and the minimization of the number of tool switches (MTSP). Computational tests of the model are presented.

Keywords

Cutting stock, Pattern sequencing, Minimization of open stacks, Interval graphs, Linear ordering, Minimum interval graph completion.

Modelos para Sequenciação de Padrões em Problemas de Corte de Stock

Resumo

Nesta tese é abordado um problema de optimização que surge em operações de corte de stock chamado minimização do número máximo de pilhas abertas (MOSP) e é proposta uma nova formulação de programação inteira.

Associando a duração de cada pilha a um intervalo de tempo, é possível usar a teoria rica que existe em grafos de intervalos para criar um modelo baseado no completamento de um grafo por arcos. A estrutura deste tipo de grafos admite uma ordenação linear dos vértices que define uma ordenação linear das pilhas e, por sua vez, determina a sequência dos padrões de corte.

O politopo definido por esta formulação tem dimensão completa e prova-se que as principais desigualdades do modelo são facetas. São derivadas desigualdades adicionais baseadas nas propriedades de grafos cordais e de grafos de comparabilidades. O número máximo de pilhas abertas está relacionado com o número cromático do grafo solução, pelo que o modelo é reforçado com a formulação por representativos para o problema de coloração de vértices.

O modelo é aplicado à minimização de pilhas abertas, e também ao problema de completamento mínimo de um grafo de intervalos e a outros problemas de sequenciação de padrões, tais como a minimização da dispersão de encomendas (MORP) e a minimização do número de trocas de ferramentas (MTSP). São apresentados testes computacionais do modelo.

Palavras Chave: Corte de stock, Sequenciação de padrões, Minimização de pilhas abertas, Grafo de intervalos, Ordenação linear, Completamento mínimo de grafos de intervalos

Table of Contents

List of Figures **xv**

List of Tables **xvii**

1 Introduction **1**

1.1 Main contributions 3

1.2 Outline 4

2 Literature Review on Pattern Sequencing **5**

2.1 MOSP and related problems 5

2.1.1 MOSP: Minimization of Open Stacks Problem 7

2.1.2 MOOP: Minimization of Open Orders Problem . . . 10

2.1.3 MORP: Minimization of ORder spread Problem . . . 11

2.1.4 MDP: Minimization of Discontinuities Problem . . . 12

2.1.5 MTSP: Minimization of Tool Switches Problem . . . 12

2.1.6 Relationship between the MOSP, MORP and MTSP 13

2.2 IP models for MOSP and related problems 14

2.2.1 Madsen's approach to the MORP and MDP 14

2.2.2 Bard's nonlinear integer model for the MTSP 16

2.2.3 Tang and Denardo's model for the MTSP and MOSP 17

2.2.4 The model of Laporte with Pinto's approach to MOSP 21

2.2.5 Pinto's integer programming model 23

2.2.6 Baptiste's MIP formulation 24

2.2.7 Comparison between some of these models 26

2.3 Other approaches and special cases 27

2.3.1 Yanasse's work on special cases of the MOSP 27

2.3.2 Heuristics 30

2.3.3 Dynamic Programming and Genetic Algorithms . . . 34

2.4 Conclusions . . . 35

3 Interval Graphs and Layout Problems 37

3.1 Basic Graph Definitions . . . 37

3.1.1 Directed and Undirected Graphs . . . 38

3.1.2 Problems with Cliques and Stable Sets . . . 39

3.2 Chordal Graphs . . . 42

3.2.1 Perfect Elimination Order . . . 42

3.2.2 Split graphs . . . 44

3.3 Comparability Graphs . . . 45

3.4 Interval Graphs . . . 46

3.5 Perfect Graphs . . . 51

3.6 Graph Layout Measures . . . 53

3.6.1 Vertex Separation and Minimum Sum Cut . . . 53

3.6.2 Profile . . . 54

3.6.3 Cutwidth . . . 55

3.6.4 Bandwidth . . . 56

3.6.5 Treewidth . . . 57

3.6.6 Pathwidth . . . 58

3.7 Linear ordering problem . . . 59

3.8 Edge Completion Problems . . . 60

3.8.1 Different Classes for H . . . 61

3.8.2 Minimum vs. Minimal . . . 61

3.8.3 Special Classes of Input Graphs . . . 62

3.8.4 Equivalent Problems in Graph Theory . . . 62

3.9 Applications of Graph Layout Problems . . . 63

3.9.1 DNA Physical Mapping . . . 63

3.9.2 Seriation Problem in Archaeology . . . 65

3.9.3 Sparse Matrix Computations . . . 65

3.9.4 GMLP: Gate Matrix Layout Problem . . . 66

3.10 Conclusions . . . 68

4 An Integer Formulation for the MOSP 69

4.1 Introduction . . . 69

4.2 The MOSP in a Graph . . . 70

4.3 A Preview of the Model . . . 74

4.4 The Decision Variables . . . 75

4.5 The Inequalities of the Model . . . 77

4.5.1 Linear Ordering of the Vertices . . . 77

4.5.2 Precedences of the Opening and Closing of Intervals . 77
4.5.3 Obtaining an Interval Graph 78
4.6 An objective function to evaluate the MOSP 84
4.7 The integer programming formulation for MOSP 87
4.8 Reducing the Number of Variables in the Model 87
4.9 Conclusions 90

5 Polyhedral Analysis and Valid Inequalities 91
5.1 Facets of the Polyhedron 91
5.2 Other Valid Inequalities 100
5.2.1 Neighbor of Successor Inequalities 100
5.2.2 Co-comparability Graph 102
5.2.3 Chords in k-cycles 106
5.2.4 Coloring the Vertices of the Interval Graph 111
5.3 The Revised Formulation of the Model for MOSP 116
5.4 Computational Tests 118
5.5 Conclusions 120

6 Variants of the MOSP Model 123
6.1 A Model for Minimum Interval Graph Completion 124
6.1.1 Introduction 124
6.1.2 The Formulation of the Model 125
6.1.3 Computational Results for IGC 127
6.2 Minimization of the Stack Occupation 128
6.2.1 Introduction 128
6.2.2 The formulation of the model 130
6.2.3 Computational results 131
6.3 Minimization of Tool Switches 133
6.4 Conclusions 135

7 Conclusions and Future Work 137

Bibliography 141

Index 149

List of Symbols and Acronyms

Symbol/Acronym	Description	Definition
$[uv]$	Edge from vertex u to vertex v	page 37
$N(u)$	Neighborhood of vertex u	page 38
$N[u]$	Closed neighborhood of vertex u	page 38
$\overline{N(u)}$	Anti-neighborhood of vertex u	page 38
$\overline{N[u]}$	Closed anti-neighborhood of vertex u	page 39
$\overline{N(U)}$	Anti-neighborhood of a set of vertices U	page 39
$\overline{G}$	Complement graph	page 39
K_n	Complete graph on n vertices	page 39
$\omega(G)$	Clique number of graph G	page 40
$k(G)$	Clique cover number of graph G	page 40
$\alpha(G)$	Stability number of graph G	page 40
$\chi(G)$	Chromatic number of graph G	page 41
C_n	Chordless cycle on n vertices	page 42
$\varphi(u)$	Linear ordering of vertex u	page 43
$\prec$	Precedes	page 43
$Pred(i)$	Set of predecessors of vertex i	page 43
$Succ(i)$	Set of successors of vertex i	page 43
$\|S\|$	Cardinality of set S	page 43
$VS(\varphi, G)$	Vertex Separation of graph G with layout φ	page 54
$SC(\varphi, G)$	Sum Cut of graph G with layout φ	page 54
$PR(\varphi, G)$	Profile of graph G with layout φ	page 54
$PR(A)$	Profile of matrix A	page 55
$BW(A)$	Bandwidth of matrix A	page 55
$BW(\varphi, G)$	Bandwidth of graph G with layout φ	page 57
$CW(\varphi, G)$	Cutwidth of graph G with layout φ	page 56

Symbol/Acronym	Description	Definition
$MC(\varphi, G)$	Modified Cutwidth of graph G with layout φ	page 56
$TW(G)$	Treewidth of graph G	page 58
$PW(G)$	Pathwidth of graph G	page 59
$\Delta(G)$	Maximum degree of a vertex of G	page 57
AT	Asteroidal Triple	page 46
FPT	Fixed Parameter Tractable	page 13
IP	Integer Programming	page 2
IGC	Interval Graph Completion Problem	page 61
IGS	Interval Graph Sandwich Problem	page 64
GMLP	Gate Matrix Layout Problem	page 67
KTNS	Keep Tool Needed Soonest	page 16
LB	Lower Bound	page 20
LOP	Linear Ordering Problem	page 59
MDP	Minimization of the number of Discontinuities Problem	page 12
MIP	Mixed Integer Programming	page 24
MOOP	Minimization of the number of Open Orders	page 10
MORP	Minimization of ORder spread Problem	page 11
MOSP	Minimization of the number of Open Stacks Problem	page 7
MTSP	Minimization of the number of Tool Switches Problem	page 12
PAP	Pattern Allocation Problem	page 6
PSP	Pattern Sequencing Problem	page 6
TSP	Travelling Salesman Problem	page 14
UB	Upper Bound	page 30
VLSI	Very Large Scale Integrated Circuits	page 66

List of Figures

2.1 A typical automatized sheet cutting system 8
2.2 Counting the number of open stacks of the example in Table 2.1 9
2.3 The solution of the example in Table 2.1 10

3.1 Chromatic number, stability number and clique number 41
3.2 Not an interval graph 47
3.3 G is a comparability graph but $\overline{G}$ is not an interval graph . . . 48
3.4 Replacing endpoints to remove coinciding endpoints 49
3.5 The p.e.o. gives the linear ordering of intervals 50
3.6 The p.e.s. is not fit for ordering the intervals 51
3.7 Intersection between subclasses of perfect graphs 53
3.8 A graph with a tree decomposition of width 2 58
3.9 Incidence matrix of Graves vs. Varieties 66
3.10 The Gate Matrix Layout Problem in a VLSI circuit 67

4.1 MOSP Graph of the instance in Table 4.1 71
4.2 Non simultaneous items can share stack space 73
4.3 Interval Graph of the instance in Table 4.1 73
4.4 Interval i opens and closes before j starts 77
4.5 Olariu's characterization of interval graphs 78
4.6 Interval i closes before j opens, with k simultaneous to i 83
4.7 Interval i closes before j and k open, with k opening before j . 83
4.8 Interval i closes before j and k open, with j opening before k . 83
4.9 Optimal solution of the example from Table 4.1 86

5.1 $\overline{H}$ must be transitively orientable 102
5.2 MOSP Graph of the instance in Table 5.2 103
5.3 Interval graph of the instance in Table 5.2 104

5.4 Optimal solution of the example in Table 5.2 105
5.5 A 4-cycle must have a chord 106
5.6 5-cycles with two chords: (a) is chordal, (b) is not chordal . . . 107
5.7 Solution of the example in Table 5.2 113
5.8 Interval representations where i is a representative of j and k . 113
5.9 A minimum colored 5-cycle after adding 2 chords 114

6.1 Stack occupation for the instance wbo10101 129
6.2 A two step model to minimize the occupation of stacks 130
6.3 The MOSP graph of Tang and Denardo's instance 135
6.4 Solution of the MOSP for Tang and Denardo's instance 135

7.1 MOSP graph of the Miller instance 138

List of Tables

2.1 An example of the MOSP with 8 patterns and 6 items 9

4.1 An instance of the MOSP with 7 cutting patterns and 6 items . 70
4.2 Possible cases when arc $[ik] \in F$ 82

5.1 Points P_{ij}^{xR} and P_{ij}^{xL} for $N = 4$ 94
5.2 An instance for the MOSP with 5 items and 8 patterns 103
5.3 Computational results for the MOSP models 121

6.1 Results for the minimum interval graph completion 128
6.2 Results for minimizing the stack occupation 132
6.3 Instance of the MTSP with 6 jobs and 9 tools 134

Chapter 1

Introduction

> "The organizational and economical aspects of the cutting operations show that cutting optimization is not only a matter of waste minimization, but also a complex planning problem whose solution has to balance conflicting objectives." [20]

The work presented in this thesis began with the study of one famous problem in Operations Research: the cutting stock problem. It was introduced in 1961 by Gilmore and Gomory, who put forward a linear programming model [27]. From then on there has been an increasing number of papers involving cutting stock operations. There are many different problems that can emerge from a cutting industry setting.

Cutting stock operations require advanced planning. The classic cutting stock problem consists in defining the cutting patterns with a cost minimization criterion that usually depends on the waste of the cutting process. But even after the cutting patterns are defined, there is more optimization that can be done in order to reduce the cost of the operations. The sequence in which the cutting patterns will be processed on the cutting equipment can be a relevant factor for the efficiency of the operations, for the organization of the work area space, for the fulfillment of the customers' orders on time, or for the fastness of the deliveries to customers. These concerns gave rise to several pattern sequencing problems, such as the minimization of open stacks and the minimization of the order spread.

In literature, pattern sequencing problems have been studied both alone and integrated with the determination of the cutting patterns. The most used approach is to solve the problem combining two stages, a first stage where the cutting patterns are defined and a second stage where the sequence of the implementation of the cutting patterns is decided. This work is devoted to the second stage, when the cutting patterns are already determined but the sequence in which they will be processed is still an open issue. The main problem addressed here is the minimization of the maximum number of open stacks, also called MOSP.

The Minimization of Open Stacks Problem (MOSP) comes from the flat glass cutting industry, but it also has many applications in other cutting industries (wooden panels, steel tubes, paper,...) as well as in other fields such as production planning, VLSI circuit design and in classic problems from graph theory. The MOSP problem is based on the premise that the different items obtained from cutting patterns are piled in stacks in the work area until all items of the same size have been cut. Usually, machines process one cutting pattern at a time and the sequence in which preset cutting patterns are processed can affect the number of stacks that remain around the machine.

Due to space limitations and danger of damages on the stacked items, it is advantageous to find a sequence for the patterns that minimizes the number of different items that are being cut and therefore the number of open stacks.

This problem has been widely studied in literature, but there are several other pattern sequencing problems, such as the minimization of the order spread (MORP) and the minimization of discontinuities (MDP).

Most of the pattern sequencing problems are NP-hard problems, which means that they cannot be solved in polynomial time unless P=NP.

The minimization of open stacks problem is known to have tight relations with problems in graph theory such as treewidth, vertex separation and the profile of a matrix. Thus it becomes the next step to learn these graph concepts. In studying these problems, we found a type of graphs called interval graphs that will play an important role in this work. They can be used to describe a solution of the pattern sequencing problems, by modeling the duration of the intervals in time in which the same piece type is being cut. Using several properties of this type of graphs we will see that it is possible to build an integer programming model for the minimization of open stacks problem.

An Integer Programming (IP) model is a formulation of the problem that has a linear objective function and linear inequalities and uses only integer or binary variables.

The linear relaxation of an integer program is the exact same formulation but without the integrality constraints. Linear programs can be solved by the *simplex* method, but the rounding up of the solution of the linear relaxation is not a good method to find the solution of an integer programming problem, because it does not always lead to the optimal integer solution.

An integer programming model can be solved with a *branch and bound* technique. This is an enumeration scheme that searches for an optimal integer solution by iteratively subdividing the feasible region and comparing the fractional solutions obtained in the linear relaxation with the integer solutions. The optimal integer solution is obtained when, at the moment, the difference between the best integer solution and the best value of the linear relaxation, called the *gap*, is proved to be zero.

Commercial solvers can perform satisfyingly branch and bound procedures and numerous heuristics in a very small time.

There are more complex methods for solving integer programming models, such as column generation and branch and price. These methods are much more efficient than the simple branch and bound, because they take advantage of the structure of the problem, but to implement them it is necessary to create an algorithm specific to each formulation, or at least to write code to interact with commercial software and adapt it to the problem.

1.1 Main contributions

The main contribution of this thesis is to model the minimization of open stacks problem using an integer programming new formulation. There have been approaches to this problem using graphs before, but none used the addition of arcs to the graph and the properties of interval graphs to achieve a solution.

MOSP is modeled as an interval graph completion problem. An initial integer programming model was derived, based on a characterization of interval graphs by Olariu.

We further strengthened the initial inequalities of the model by combination and rounding to derive stronger inequalities. We also proved them to be facets of the interval graph completion problem.

The fact that MOSP is modeled as interval graph completion problem provides new insight into the structure of the problem. Constraints to cut non-chordal 4- and 5-cycles can be used to strengthen the model. These constraints can be derived from the characterization of interval graphs. We showed that they can also be derived by combination and rounding of the previous facets of the interval graph completion problem.

Last but not the least, we derive a general framework that can be used to model many related problems.

With the choice being integer programming, the formulation developed in this work can later be integrated in other integer programming models for cutting stock problems, namely to create a combined model of the stages one and two where the cutting stock patterns are defined and sequenced.

1.2 Outline

For the rest of this thesis, in Chapter 2 the pattern sequencing problems are explained and compared. A literature review on this type of cutting stock problems is made, emphasizing on the integer programming approaches.

Chapter 3 is dedicated to recall basic graph theory, special types of graphs such as interval graphs, chordal graphs and perfect graphs, and graph layout measures of a graph. There is a section that presents and compares edge completion graph problems such as the minimal fill-in and the minimum interval graph completion. The last section describes applications of the graph layout problems in Molecular Biology, Archaeology, Numerical Analysis and VLSI circuit design.

In Chapter 4 we propose a new integer programming model for the minimization of open stacks based on the edge completion of the MOSP graph and the structure of the linear ordering of the vertices in an interval graph. The main variables and inequalities are presented and a variation with less variables is discussed.

The formulation is strengthened in Chapter 5. Using simple polyhedral theory it is proved that most of the inequalities are facets. The model is investigated further originating a new set of variables related to the coloring of the vertices to be added to the model and more inequalities are derived. Computational tests are discussed at the end of the Chapter.

In Chapter 6, the model is applied to different problems. By making small modifications in the objective function and using only some of the inequalities, the MOSP model is applied to the minimum interval graph completion problem. Another pattern sequencing problem that aims to minimize, not only the number of stacks, but also the order spread (the minimization of the stack occupation problem) is considered, and the model is tested.

Finally, Chapter 7 concludes this thesis and presents some directions of future work.

Chapter 2

Literature Review on Pattern Sequencing Problems

In this chapter, we present the Minimization of Open Stacks Problem (MOSP) and some related problems, and refer to their relationship. Then, we focus on the integer programming models developed for the MOSP and the related problems, considering that the contributions of this thesis are in this area of work. We also refer to other approaches used for these problems, as heuristics, genetic algorithms and dynamic programming.

2.1 The Minimization of Open Stacks Problem and Related Problems

Cutting Stock problems and Packing problems (C&P) have an identical structure and are often considered in common. As defined in [60], both of the problems consist of, given a set of large objects (supply) and a set of small items (demand) which are defined exhaustively in n geometric dimensions, selecting some or all small items, grouping them into one or more subsets and assigning each of the resulting subsets to one of the large objects such that the geometric condition holds (i.e., the small items of each subset have to be laid out on the corresponding large object such that all small items of the subset lie entirely within the large object and the small items do not overlap), while optimizing a given objective function.

The differences between the two problems rely mainly on the variety of the small items. The *Cutting Stock Problem* requires a weakly heterogeneous assortment of small items to be allocated to a selection of large objects with extension fixed in all dimensions. The *Bin Packing Problem* is characterized by a strongly heterogeneous assortment of small items that are to be assigned to a set of large objects. The value, number, or total size of the necessary large objects has to be minimized in both problems.

Cutting and packing problems have been categorized, the most recent typology being the one defined by Wäscher et al [60]. In this typology, the problems studied in this thesis belong to a problem extension, of the pattern sequencing subtype, because they include additional aspects which extend the view of planning beyond the core of cutting or packing.

Industrial cutting operations involve taking large objects of standard sizes (stock material such as wooden panels, paper rolls, aluminium profiles, flat glass) and cutting them into smaller pieces of different sizes to meet customers' demands. A specification of how many small items of each size will be cut from each large panel and where the cuts will be made defines a *cutting pattern*. Each cutting pattern can produce different items or just several copies of one same item.

Cutting stock problems usually deal with the generation of a set of cutting patterns that minimizes waste. But beyond pattern generation, there are additional aspects that are important to deal with, in the process of planning industrial cutting operations. An important issue is the decision of the processing sequence of the set of patterns on the cutting equipment. Most probably the first researcher raising awareness of these aspects is Dyson [19].

The *Pattern Sequencing Problems* (PSP), also referred to as *Pattern Allocation Problems* (PAP), consist in finding a permutation of the predetermined cutting patterns while optimizing a given objective function. There are several relevant objective functions: the number of tool changes, the average order spread, the number of discontinuities, and the number of open stacks.

A set of m cutting patterns relating n item types can be represented in a $n \times m$ matrix A, whose element a_{ij} equals 1 if pattern j contains item i and equals 0, otherwise. Pattern sequencing problems consist of constructing a permutation of the columns of this matrix, while minimizing some given objective function. From the permutation of the columns comes the ordering for processing the patterns. There are, evidently, $m!$ solutions.

The pattern sequencing problem with the number of stacks as the objective function to be minimized has been the purpose of most of the papers on pattern sequencing problems, maybe because the problem itself has a complex structure and it is very rich in applications to other fields of science.

Most of the authors encounter the minimization of the number of open stacks problem while solving a two stage procedure: first, the classic problem of finding the best patterns to cut stock sheets of glass, paper or wood into smaller rectangular pieces is dealt, usually with an waste minimization purpose, and only after that do they deal with the stage two, which is to determine the sequence in which those patterns should be cut, in order to minimize the number of open stacks. There are also researchers who tried to solve both the problems of pattern generation and pattern sequencing in an integrated way, and others use the number of open stacks rather as a constraint than as the objective function.

The number of publications in the subject of pattern sequencing problems reaches over fifty. Although it is a relatively recent topic, it has been addressed with many different methods over the last thirty five years. The purpose of this chapter is to study the cutting stock problems dedicated to sequencing the cutting patterns with an integer programming perspective.

2.1.1 MOSP: Minimization of the number of Open Stacks Problem

The Minimization of Open Stacks Problem (MOSP) is a pattern sequencing problem that was first addressed in 1991 by Yuen [68] and Richardson [70]. It arose in the Australian flat glass industry, but it can appear in other cutting industries like steel tubes, paper, wooden panels, and others.

Consider a cutting machine that processes just one cutting pattern at a time. The different items cut from patterns are queued around a machine in separate stacks. The stack of an item type remains near the machine if an item of that type still has to be cut from a forthcoming pattern. A stack is closed and removed from the work area only after all items of that size have been cut, and immediately before starting to process the next cutting pattern. During this process, the number of open stacks is counted after a pattern is completely cut and before any finished stack is removed.

As there are often space limitations around the cutting machines, and in some cases as in the flat glass industry there is danger of accidental breakages of the pieces in the partially completed stacks and difficulty in distinguishing similar items, it becomes important to minimize the maximum queue of partially cut orders, i.e. to minimize the maximum number of simultaneously open stacks. That can be achieved simply by finding

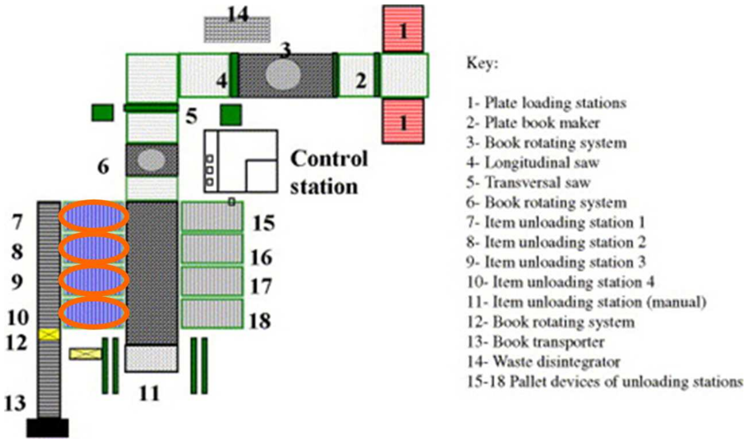


Figure 2.1: A typical automatized sheet cutting system with limited unloading stations [46]

an optimal sequence to process the cutting patterns. The minimization of open stacks problem consists of constructing a sequence of the patterns with respect to minimizing the number of simultaneously open stacks.

More formally, consider the matrix $A_{n\times m}$ with the constitution of the m cutting patterns, whose element a_{ij} equals 1 if pattern j contains item i and equals 0, otherwise. A sequence to process the cutting patterns is a permutation $\Pi = (\pi_1, ..., \pi_m)$ of the columns of this matrix, where π_j denotes the pattern that is positioned currently in column j.

A stack i is *open* at position t of the pattern sequence if

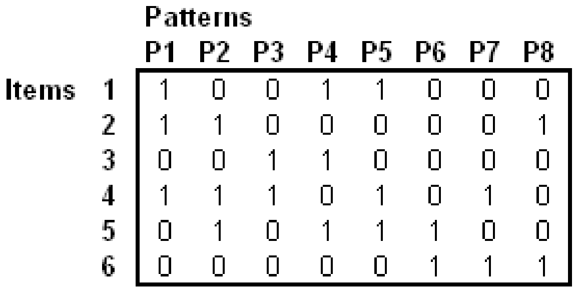
$$\sum_{j=1}^{t} a_{i\pi_j} \cdot \sum_{j=t}^{m} a_{i\pi_j} > 0$$

We define the *MOSP number* of a permutation $\Pi = (\pi_1, ..., \pi_m)$ of the m patterns as

$$MOSP(\Pi) = \max_t \left\{ i : \sum_{j=1}^{t} a_{i\pi_j} \cdot \sum_{j=t}^{m} a_{i\pi_j} > 0 \right\}$$

The optimal solution of the minimization of open stacks problem is a permutation Π of the columns of matrix A such that $MOSP(\Pi)$ is minimum over all such permutations.

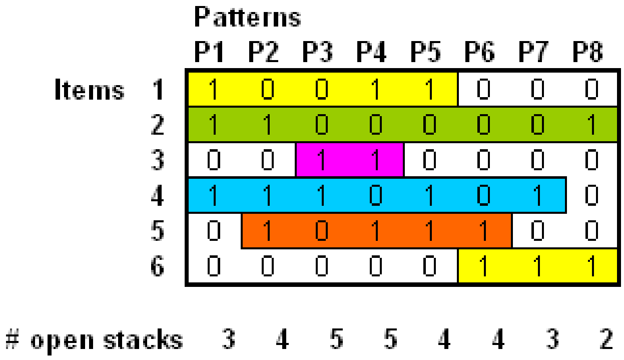
Observe an example of this problem with 8 cutting patterns and 6 items taken from [67]. The composition of the cutting patterns is described in Table 2.1.



Items	Patterns P1	P2	P3	P4	P5	P6	P7	P8
1	1	0	0	1	1	0	0	0
2	1	1	0	0	0	0	0	1
3	0	0	1	1	0	0	0	0
4	1	1	1	0	1	0	1	0
5	0	1	0	1	1	1	0	0
6	0	0	0	0	0	1	1	1

Table 2.1: An example of the MOSP with 8 patterns and 6 items

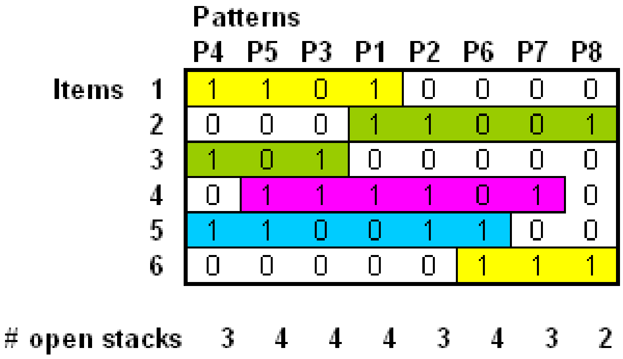
If the patterns are processed in the original sequence (Figure 2.2), there is a point where there are 5 simultaneously open stacks. But if the patterns are sequenced in a different order, the maximum number of open stacks may be different. If the ordering of the patterns is as in Figure 2.3, there are only 4 simultaneous open stacks at most.



Items	Patterns P1	P2	P3	P4	P5	P6	P7	P8
1	1	0	0	1	1	0	0	0
2	1	1	0	0	0	0	0	1
3	0	0	1	1	0	0	0	0
4	1	1	1	0	1	0	1	0
5	0	1	0	1	1	1	0	0
6	0	0	0	0	0	1	1	1
# open stacks	3	4	5	5	4	4	3	2

Figure 2.2: Counting the number of open stacks of the example in Table 2.1

The applications of the minimization of open stacks problem are usually in the cutting industry, but this problem can also be found in production planning as explained in Section 2.1.2, and yet in completely different scenarios as VLSI circuits design with the Gate Matrix Layout Problem and PLA Folding, or in classical problems from Graph Theory such as Pathwidth, Modified Cutwidth and Vertex Separation, as we will be seeing further ahead.



	Patterns							
	P4	P5	P3	P1	P2	P6	P7	P8
Items 1	1	1	0	1	0	0	0	0
2	0	0	0	1	1	0	0	1
3	1	0	1	0	0	0	0	0
4	0	1	1	1	1	0	1	0
5	1	1	0	0	1	1	0	0
6	0	0	0	0	0	1	1	1
# open stacks	3	4	4	4	3	4	3	2

Figure 2.3: The solution of the example in Table 2.1

2.1.2 MOOP: Minimization of the number of Open Orders Problem

Consider the case of a manufacturer who has a number of orders from customers to fulfill. Each order requires the making of different products, but only one product can be made at a time. A virtual stack is opened for each client when the first product of that order is processed. The stack is closed as soon as all products of that order have been manufactured, so the order is completed and ready to be sent to the client. The setting costs usually prevent from switching between manufacturing different products, and transportation costs dissent from partial order deliveries to clients. Also it is not advisable to have delays on deliveries, because, besides having resources tied up to stock, it will cause delays on the customers' payments. The objective is to find an optimal sequence to manufacture the products in order to minimize the maximum number of simultaneously open orders. Manufacturing the products in this optimal sequence reduces the amount of space needed to store the incomplete orders.

The minimization of the number of open orders, also called MOOP [61], is just a different perspective of the MOSP problem, but it is the same problem, whether it is considered in production planning or when sequencing cutting stock patterns. Comparing the two scenarios, *cutting patterns* have the same behaviour as *products*, and *items cut* will correspond to *customers' orders*. While in the MOSP problem we have a real stack for each piece size that is cut, in this scenario real physical stacks for each order that is not fully fulfilled may not exist.

2.1.3 MORP: Minimization of ORder spread Problem

The Minimization of Order Spread Problem (MORP) is a pattern sequencing problem very similar to the MOSP. When considered from the viewpoint of a cutting stock problem, it also deals with the sequencing of the patterns, but the problem has an objective function different from MOSP.

In some cases like in the glass cutting industry, the glass pieces have to be handled and stored individually, which is very time- and storage-consuming and hence expensive; therefore it is desirable to reduce the handling and storing costs [45] by reducing the time elapsed between the cutting pieces correspondent to the same order.

An order from a customer often consists of several items that may be cut from different stock sheets. A possibility is to create a stack for each customer, with the pieces that are being cut, until each order is complete. An order is completed when the corresponding demand has been met entirely.

The *order spread* is the distance between the first and the last item cut that belongs to the same order. The distance can be measured in number of stock sheets: if the whole order is cut from just one stock sheet, the order spread is 0, if it is cut completely in two consecutive sheets, the order spread is 1 [45].

In practice, it is desirable to keep all the pieces cut belonging to one order as close in time as possible. The objective of the MORP is therefore to minimize the order spread, that is minimizing the time that a stack remains open.

According to [23], the MORP can be defined more formally in the following manner.

Consider the matrix $A_{n\times m}$, whose element a_{ij} equals j if order i is to be cut from pattern j and equals 0 otherwise, and the permutation $\Pi = (\pi_1, ..., \pi_m)$ of the columns of this matrix, which defines the sequence to process the cutting patterns, where π_j denotes the pattern that is cut in position j.

The *spread* of order i in the sequence Π is

$$s_{i\Pi} = \max_j \{a_{ij} : a_{ij} > 0\} - \min_j \{a_{ij} : a_{ij} > 0\}$$

and the *average order spread* of a pattern sequence Π is

$$s(\Pi) = \frac{1}{n} \sum_{i=1}^{n} s_{i\Pi}$$

The MORP problem consists in selecting a pattern sequence that minimizes the average order spread over all permutations of the patterns.

$$MORP = \min_{\Pi} s(\Pi)$$

The minimization of the order spread is related to the minimization of the bandwidth of a symmetric matrix, which we will discuss in section 3.6.

According to Linhares and Yanasse [44], the MORP may also be defined as the minimization of the *maximum* order spread, instead of the *average* order spread, as defined above. Both versions are NP-hard problems in the general case [21, 25].

2.1.4 MDP: Minimization of the number of Discontinuities Problem

We say that a *discontinuity* occurs when an item that is being cut in a given pattern is not cut in the following pattern and it is cut later again. The problem of minimization of discontinuities (MDP) is to find a sequence to process the cutting patterns such that the number of this type of discontinuities is minimum. The difference from the previous problem is that in this case the duration of the discontinuities is not considered to influence the cost of the solution, just its existence.

This is a NP-hard problem, and it is also known in the literature as the Consecutive Blocks Minimization Problem [26].

2.1.5 MTSP: Minimization of the number of Tool Switches Problem

Also related to the MOSP problem we have the minimization of the number of tool switches problem (MTSP). This is a job scheduling problem that arises in flexible manufacturing machines, and it has been applied to the metal working industry, the cutting operations in a manufacturer of military avionics and communications equipment, and the mounting operations in electronic manufacturing of printed circuit boards.

Machines in such systems are capable of different tasks, but may need a certain combination of tools. This problem involves machines that can hold a set of tools, which can be changed in order to have the adequate tools for each job. As these machines only have a capacity for C tools simultaneously, some tool switching must be made between different tasks sometimes. These tool changing operations may include retrieval from storage, transportation, loading and calibration, and have a cost which is proportional to the number of switches. The MTSP problem consists of finding a sequence of the tasks, in order to minimize the number of tool switches.

There are usually some assumptions on this scenario [5]: we are scheduling N jobs on a single machine, the number of tools required to process N jobs is greater than the capacity of the machine processing times, switching times are independent and batch sizes are small; we also consider that only one tool is changed at a time, the magazine can accommodate any combination of tools, tool changing times are constant and identical, all tools weight about the same and each takes only one position in the magazine. Additionally, job processing is independent of switching and tool wear does not influence the planning decision.

This is a NP-hard problem for any fixed $C \geq 2$, as shown by Crama et al. as quoted in [16], because it has similarities with the problem of minimal length traveling salesman path on edge graphs.

It involves machines that can hold a set of tools, which can be changed in order to have the adequate tools for each job. This problem is similar to the MOSP problem, where the patterns here are the jobs and the items are the tools.

2.1.6 Relationship between the MOSP, MORP and MTSP problems

In this section we will see several relations between these problems.

Only the two perspectives of the MOSP enunciated above (MOSP and MOOP) are equivalent. The MORP, the MDP and the MTSP are NP-hard problems that are not equivalent to the MOSP, and not even to each other. Linhares and Yanasse [44] have presented counterexamples to all the equivalency conjectures, except for the equivalence between the MTSP and MDP. They prove that if MTSP is fixed parameter tractable (FPT), then MDP is also FPT.

Although the MOSP is not equivalent to MTSP in general case, they are equivalent just if the optimum of the MOSP equals the capacity C of the machine in the MTSP. Denote the optimum number of stacks in MOSP by C^*. If $C > C^*$, an optimal solution for the MOSP is always optimal for the MTSP, but the converse is not true. If $C < C^*$, an optimal solution for the MOSP may not be an optimal solution for the MTSP and vice-versa. Yanasse [63] proves that the MOSP is not equivalent to the MTSP in a general case, but they are equivalent only when $C = C^*$.

We will see two formulations for the MOSP that were adapted from formulations originally developed for the MTSP [54, 63].

MORP and MDP are not equivalent, but in the next chapter we will see an effort made by Madsen [45] to solve the MORP by minimizing the discontinuities.

2.2 Integer programming models for MOSP and related problems

In this section we will explain some of the integer programming models that exist is literature for the minimization of open stacks problem and related problems.

2.2.1 Madsen's approach to the MORP and MDP

In an attempt to solve the minimization of order spread problem, Madsen [45] has developed exact models based on the Travelling Salesman Problem (TSP).

These methods result in decreasing the maximum order spread but sometimes at the cost of increasing the average order spread. The exact formulation was presented by Madsen as a model for the MORP, but in fact it only solves the MDP because it does not consider the duration of the discontinuities when counting the order spread.

Madsen also makes an analogy of the order spread with the bandwidth of a symmetric matrix, rearranging the rows and the columns in order to diagonalize the matrix to the maximum. The bandwidth problem is explained in section 3.6.4.

The method suggested by Madsen [45] is composed of an iterative procedure alternating between stages one (cutting stock problem) and two (pattern sequencing problem). The first stage is suggested to be solved with Gilmore and Gomory algorithm, and the second stage is solved with a Travelling Salesman Problem approach. Madsen's objective is to minimize the order spread and he tries to accomplish it by sequencing the patterns that contain the same panel type, one after another, without interruptions, until the demand is fully completed. Let $C = \{c_{ij}\}$ be a symmetric matrix where

$$c_{ij} = k - \text{number of orders on stock sheet } i \text{ which also occur on stock sheet } j$$

where k is a large constant greater than the maximum number of orders in common, in order to guarantee that $c_{ij} \geq 0$, and c_{ii} is set to a constant greater than k. Defining the binary variables

$$x_{ij} = \begin{cases} 1 & \text{if stock sheet } j \text{ is cut immediately after stock sheet } i \\ 0 & \text{otherwise} \end{cases}$$

The order spread reduction problem is formulated in the following manner:

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

subject to

$$\sum_{i=1}^{N} x_{ij} = 1 \quad j = 1, ..., N \tag{2.2}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \quad i = 1, ..., N \tag{2.3}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subset \{1, ..., N\} \tag{2.4}$$

$$x_{ij} \in \{0, 1\} \quad i, j = 1, ..., N \tag{2.5}$$

The objective function (2.1) is chosen in such a way that variables x_{ij} are set to 1 when c_{ij} is minimum, which happens for sheets i and j that have a large number of panel types in common. The minimization of the order spread is made with an objective function that chooses to sequence the patterns that have the maximum number of common panel types.

The inequality (2.2) enforces that there is only one cutting pattern before j and (2.3) guarantees that there is only one cutting pattern after i. The left hand side of the inequality (2.4) counts how many adjacencies there are in any subset S of cutting patterns; the variables satisfy it at equality when all the cutting patterns in S are adjacent in the sequence.

The weakness of this TSP formulation resides in not adequately modeling the distance from the moment the cutting of an order is stopped to the moment when it is started again. It only reduces the number of discontinuities, i.e, the number of times that an order that was partially completed by one cutting pattern is not processed in the next pattern. Therefore, it does not always lead to a solution for the MORP but rather to the MDP.[23]

2.2.2 Bard's nonlinear integer model for the MTSP

A paper of Bard [5] uses Nonlinear Integer Programming to solve the MTSP. Let us recall that the minimization of the number of tool switches problem (MTSP), consists in selecting the sequence in which to process the N jobs and the M tools to switch on a machine with only C slots, in order to minimize the number of tool switches. Bard uses the following variables:

$$r_{ij} = \begin{cases} 1 & \text{if tool } i \text{ is required for job } j \\ 0 & \text{otherwise} \end{cases}$$

$$x_{jn} = \begin{cases} 1 & \text{if job } j \text{ is assigned to position } n \\ 0 & \text{otherwise} \end{cases}$$

$$y_{in} = \begin{cases} 1 & \text{if tool } i \text{ is on the magazine at instant } n \\ 0 & \text{otherwise} \end{cases}$$

The mathematical formulation of the problem is

$$\text{Minimize} \quad F(x,y) = \sum_{n=1}^{N} \sum_{i=1}^{M} y_{i,n}(1 - y_{i,n-1})$$

subject to:

$$\sum_{n=1}^{N} x_{jn} = 1 \quad j = 1, ..., N \tag{2.6}$$

$$\sum_{j=1}^{N} x_{jn} = 1 \quad n = 1, ..., N \tag{2.7}$$

$$\sum_{i=1}^{M} y_{in} \leq C \quad n = 1, ..., N \tag{2.8}$$

$$r_{ij} x_{jn} \leq y_{in} \qquad \forall i, j, n \tag{2.9}$$

$$x_{jn}, y_{in} = 0, 1 \quad \forall i, j, n \tag{2.10}$$

This formulation has experienced bad results, so the author suggests a problem reduction combined with heuristics. After a first preprocessing, a Lagrangian relaxation is executed on conditions (2.9) and the dual of (2.8) is computed. A Hungarian algorithm along with backward dynamic programming transforms the problem in one of order $\mathcal{O}(N^3)$ or $\mathcal{O}(NM)$. Then, to construct feasible solutions, a policy called "Keep Tool Needed Soonest (KTNS)" is used. This policy was introduced by Tang and Denardo [57], and relies on two principles:

- At any instant no tool is inserted unless it is required for the next job.
- If a tool must be inserted, the tools kept are those needed the soonest.

This policy is proved to minimize the total number of tool switches for the tool replacement problem and is of order $\mathcal{O}(MN)$.

Finally a branch and bound scheme solves the problem resulting in $\mathcal{O}(N^4 M)$.

With this combined method, good results were obtained. The method can also be extended for P machines and can include fixed costs. However the author finds it difficult to include in this model the wear of the tools. Bard claims that finding the job sequence and tool replacement that minimizes the total number of switches is equivalent to minimizing the *makespan*, i.e., the time difference between the start and finish of a sequence of manufacturing tasks.

2.2.3 Tang and Denardo's model for the MTSP and MOSP

Regarding the MTSP, Tang and Denardo's work [57] uses integer programming to schedule the N jobs on a single machine with only C slots having M tools available.

Tang and Denardo consider instant n to be the moment after the n^{th}-job is processed but before any tools are switched, for $n = 1, 2, ..., N-1$, and use the following variables:

$$x_{jn} = \begin{cases} 1 & \text{if job } j \text{ is the } n^{th}\text{-job in the sequence} \\ 0 & \text{otherwise} \end{cases}$$

$$w_{ni} = \begin{cases} 1 & \text{if tool } i \text{ is on the machine at instant } n \\ 0 & \text{otherwise} \end{cases}$$

which forms a $M \times 1$ vector W_n giving the tools in the machine at instant n,

$$p_{ni} = \begin{cases} 1 & \text{if tool } i \text{ is switched at instant } n \\ 0 & \text{otherwise} \end{cases}$$

which forms a $M \times 1$ vector P_n showing the tool switches that occur at instant n,

$$a_{ji} = \begin{cases} 1 & \text{if tool } i \text{ is required for job } j \\ 0 & \text{otherwise} \end{cases}$$

which forms a $M \times 1$ vector A_j representing the set of tools required for job j. Denote by e a $1 \times M$ vector full of 1's. Consider also that each tool magazine has C tool slots and there is no tool sharing among different machines. The mathematical formulation of the problem is:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{n=1}^{N-1} eP_n & & \qquad (2.11)\\
\text{subject to} \quad & eW_n = C & n = 1, ..., N & \qquad (2.12)\\
& x_{jn}A_j \leq W_n & j = 1, ..., N, \quad n = 1, ..., N & \qquad (2.13)\\
& \sum_{j=1}^{N} x_{jn} = 1 & n = 1, ..., N & \qquad (2.14)\\
& \sum_{n=1}^{N} x_{jn} = 1 & j = 1, ..., N & \qquad (2.15)\\
& P_n \geq W_{n+1} - W_n & n = 1, ..., N-1 & \qquad (2.16)\\
& x_{jn} \in \{0, 1\} & j = 1, ..., N, \quad n = 1, ..., N & \qquad (2.17)\\
& W_n \in \{0, 1\}^M & n = 1, ..., N & \qquad (2.18)\\
& P_n \geq 0 & n = 1, ..., N-1 & \qquad (2.19)
\end{aligned}$$

In the objective function (2.11), eP_n counts how many tool switches happen at instant n, hence the sum counts the total number of tool switches. Note that there is no need to make any tool switch at the last instant N. The reason for the equality in equation (2.12) is the following: as there are C slots in the machine, it is advantageous to load C tools on the machine and only switch the necessary ones at each instant. Clearly, if $C \geq M$, then there is no need for tool switches.

Inequality (2.13) assures that if job j happens at instant n, then the necessary tools A_j must be loaded in the magazine at that instant. The next two inequalities (2.14) and (2.15) assign exactly one job to each instant and vice versa.

The last inequality (2.16) states that if, at instant $n + 1$, there is the need for a tool that was not in the magazine at instant n, which is stated by an entry of the first vector of the right hand side to be one and the corresponding entry on the second vector to be zero, then this inequality forces the corresponding entry in the vector of the left hand side to be one too, meaning that there is a tool switch at instant n.

The authors reported rather disappointing results, and tried to replace equation (2.13) with a stronger condition:

$$\sum_{j=1}^{N} x_{jn} A_j \leq W_n \quad n = 1, ..., N \tag{2.20}$$

A solution of this problem would also be a solution of the first, but unfortunately this did not improve the results.

In [41], Laporte points out that the linear relaxation of this model is always zero when no job is fixed, and therefore it requires almost complete enumeration.

Another tactic that Tang and Denardo suggested was to consider a subproblem, named the Tool Replacement Problem, where the job sequence is fixed but only the tooling decisions are open. The objective of this problem is to determine the set of tools to place on the machine at each instant so that the total number of tool switches is minimized. For this subproblem the following variables are needed:

- $T(i,n)$ is the set of all instants at (or after) instant n at which tool i is needed;
- $L(i,n)$ is an integer variable that stands for the first instant at (or after) instant n at which tool i is needed;
- a vector J such as

$$J_i = \begin{cases} 1 & \text{if tool } i \text{ is on the machine at instant } n \\ 0 & \text{otherwise} \end{cases}$$

Tang and Denardo invoke the "Keep tool needed soonest"policy, described in the previous section, to minimize the total number of tool switches.

Tang and Denardo also suggested a graph approach for this problem: to consider graphs where the nodes represent the jobs, and the length of an arc ij is the number of tool switches when job i is followed by job j. In this perspective, every job sequence corresponds to a *Hamiltonian path* on the graph, *i.e.* a path that encounters each node exactly once and visits all nodes. The length of a Hamiltonian path represents the number of tool switches in that job sequence.

Minimizing the number of tool switches for a given job sequence is solved by finding the shortest Hamiltonian path, that is equivalent to solving the Travelling Salesman Problem, which is NP-complete.

If we denote the fewest possible number of tool switches if job j follows job i by $LB(i,j)$, the length of the shortest Hamiltonian path is a lower bound

(LB) to the number of tool switches. One drawback of this lower bound is that it ignores all tooling decisions before i and after j.

For solving the Job Scheduling problem, this method of finding lower bounds and consequently generating a good job schedule is used as the first step of a Greedy Procedure. Before doing this first step, a preprocessing phase consists in removing the jobs that require a subset of tools also required for job i, as they can be scheduled in any order immediately after i. The second step of this heuristic is to determine the tooling decision $W^J(n)$ for any job schedule J, accordingly to the KTNS policy. The third and last step of this greedy procedure is to execute a perturbation in the sequence, i.e., to change each job sequence J such that the perturbed job sequence can be performed by the same sequence of tools that is used to process the job sequence J.
This Greedy Procedure finds a job sequence J and a tooling decision W^J that is an equilibrium, i.e., for a fixed job schedule J, W^J is an optimal tooling decision for the job schedule J and for a fixed tooling decision W^J, there is no perturbed job sequence of J that entails less tool switches than the number of tool switches in J.

Tang and Denardo's model for the MTSP adapted for the MOSP by Yanasse

It was suggested in the literature that some adaptations may be done to the exact models proposed for the MTSP, in order to use those results and make them suitable models for the MOSP. In [63], Yanasse demonstrates two propositions that make the model originally proposed by Tang and Denardo [57], suitable for modeling the MOSP.

As the MTSP is only a problem if the capacity of the machine is less than the number of tools needed ($C < M$), then $M-C$ is a lower bound for the number of tool switches, because all tools must be used at least once.

Considering the jobs as patterns and the tools as panel types, we can establish a link between the MTSP and the MOSP.

Consider C^* to be the optimal solution value of the corresponding MOSP. By construction, if $C \geq C^*$ then $M - C$ is the optimal value of the MTSP. The optimal sequence of the MOSP determines what are the first C tools to be loaded initially in the C slots of the machine.

Yanasse says that when $C < C^*$ the optimal value of the MTSP is strictly greater than the lower bound $M - C$. This happens because any sequence of the patterns produces at least C^* stacks, so if we have only C

slots, $C < C^*$, at least one switch has to be made to take a tool out of the magazine temporarily and brought back again afterwards to the tool magazine. So we must necessarily have more than $M - C$ switches. So the MOSP can be formulated as

$$\text{Minimize} \quad C \tag{2.21}$$

$$\text{subject to} \quad \sum_{n=1}^{N-1} eP_n = M - C \tag{2.22}$$

$$\text{and conditions (2.12) to (2.19)} \tag{2.23}$$

Yanasse assumed that, similarly to Tang and Denardo's model, disappointing computational results would be expected from this model, and he presented an alternative branch-and-bound scheme instead.

2.2.4 The model of Laporte et al. for the MTSP with Pinto's approach to the MOSP

Laporte, González and Semet [41] have developed a model for the Minimization of Tool Switching Problem in job sequencing that can be adapted to solve the MOSP.

Consider the set of jobs $J = \{1, ..., n\}$ and the tools $T = \{1, ..., m\}$. Let us denote by J_t the subset of jobs requiring tool t and T_j the subset of tools in job j. The machine holding capacity of at most c tools at a time means that $c \geq \max_{j \in J}\{|T_j|\}$. The binary variables are defined in the following way:

- $x_{ij} = 1$ if job i is immediately followed by job j

- $y_{it} = 1$ if tool t is in the machine while processing job i

- $z_{it} = 1$ if tool t is inserted in the machine at the start of job i

The mathematical model for the MTSP is

$$\text{Minimize} \quad \sum_{i\in J}\sum_{t\in T_i} z_{it} \tag{2.24}$$

$$\text{subject to} \quad \sum_{j\in J\cup\{0\}\setminus\{i\}} x_{ij} = 1 \quad \forall i \in J\cup\{0\} \tag{2.25}$$

$$\sum_{i\in J\cup\{0\}\setminus\{j\}} x_{ij} = 1 \quad \forall j \in J\cup\{0\} \tag{2.26}$$

$$\sum_{i,j\in S} x_{ij} \leq |S|-1 \quad S\subseteq J\cup\{0\}, 2\leq |S| \leq n-1 \tag{2.27}$$

$$\sum_{t\in T} y_{jt} \leq c \quad \forall j \in J \tag{2.28}$$

$$x_{ij}+y_{ij}-y_{it} \leq z_{ji}+1 \quad \forall i\in J\cup\{0\}, \forall j\in J, \forall t\in T \tag{2.29}$$

$$y_{it} = 1 \quad \forall i\in J, \forall t\in T_i \tag{2.30}$$

$$z_{it} = 0 \quad \forall i\in J, \forall t\in T\setminus T_i \tag{2.31}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j\in J\cup\{0\} \tag{2.32}$$

$$y_{it} \in \{0,1\} \quad \forall i\in J, \forall t\in T \tag{2.33}$$

$$z_{it} \in \{0,1\} \quad \forall i\in J, \forall t\in T \tag{2.34}$$

The constraints (2.25) and (2.26) are the usual Travelling Salesman Problem (TSP) assignment constraints. The condition (2.27) prevents the formation of subtours while (2.28) is a capacity constraint. Restriction (2.29) says that if tool t was not in machine for job i and was not inserted for j, then job j cannot follow job i and require tool t at the same time. Restriction (2.30) forces all tools for job j to be in the machine before its processing and (2.31) prevents switching a tool if it is not needed to execute a job. Finally, (2.32), (2.33) and (2.34) are the integrality constraints.

To obtain a mathematical model for the MOSP, Pinto [54] modifies the objective function to

$$\min c \tag{2.35}$$

and another restriction is added, forcing the number of switches to be the difference between the total number of tools and the capacity of the machine.

$$\sum_{t=1}^{n}\sum_{i=1}^{m} z_{it} \leq m-c \tag{2.36}$$

To improve the lower bounds achieved with the linear relaxation of the model for the MTSP problem, Laporte et al. [41] recommend using extra constraints and a lifted objective function. If there is an $i^* \in J$ such that

$|T_{i^*}| = c$, then any job j immediately following i^* will generate $|T_j \backslash T_{i^*}|$ tool switches and therefore $z_{jt} \geq x_{i^*j}$ for all $j \neq i^*$ and $t \in T_j \backslash T_{i^*}$. By using the following objective function (2.37), the constraints (2.29) can be removed for $i^*, j \neq i^*, t \in T_j \backslash T_{i^*}$.

$$\min \sum_{i \in J} \sum_{t \in T_i} z_{it} + \sum_{i \in J : |T_i| = c} \sum_{j \neq i} |T_j \backslash T_i| x_{ij} \tag{2.37}$$

When j immediately follows i, a lower bound for the number of tool switches is $l_{ij} = \max\{0, |T_i \cup T_j| - c\}$. If $|T_j| = c \ \forall j \in J$ then this is the actual number of tool switches and a TSP is obtained in that case.

Laporte et al. suggested also three more valid inequalities that improved the linear relaxation of the model:

$$\sum_{t \in T_j} z_{jt} \geq \sum_{i \neq j : |T_i| \neq c} l_{ij} x_{ij} \quad \forall j \in J \tag{2.38}$$

$$\sum_{i \in J_t \backslash \{j\}} x_{ij} + z_{jt} \leq 1 \quad \forall t \in T \ \ \forall j \in J \tag{2.39}$$

$$\sum_{t \in T_i \backslash T_j} y_{jt} \geq (c - |T_i|) x_{ij} \quad \forall i, j \in J : |T_i| = c, |T_j| < c \tag{2.40}$$

The first is a lower bound for the number of tool switches for processing each job j, the second says that a required tool is introduced at the start of the job or the previous job already required that tool. The last constraint (2.40) states that it does not pay to make unnecessary tool switches.

2.2.5 Pinto's integer programming model based on the completion of stacks

In [66], a new integer model is presented which aims on the sequence of the completion of the stacks more than it focus on the sequencing of the patterns. For that reason the main variables are:

$$x_{jn} = \begin{cases} 1 & \text{if item } j \text{ is the } n^{th} \text{ item to be completed in the sequence} \\ 0 & \text{otherwise} \end{cases}$$

Consider the same variables of the previous models, plus a constant $K \geq M$, and also a $M \times 1$ vector S_j that stores the information about related items:

$$s_{ij} = \begin{cases} 1 & \text{if item } i \text{ and item } j \text{ are present in the same pattern} \\ 0 & \text{otherwise} \end{cases}$$

The integer programming formulation is:

$$
\begin{aligned}
\text{Minimize} \quad & C && && (2.41)\\
\text{subject to} \quad & eW_n \leq C + n - 1 && n = 1, ..., M && (2.42)\\
& \sum_{j=1}^{M}\sum_{t=1}^{n} x_{jt}S_j \leq KW_n && n = 1, ..., M && (2.43)\\
& \sum_{n=1}^{M} x_{jn} = 1 && j = 1, ..., M && (2.44)\\
& \sum_{j=1}^{M} x_{jn} = 1 && n = 1, ..., M && (2.45)\\
& x_{jn} \in \{0, 1\} && j = 1, ..., M, \quad n = 1, ..., M && (2.46)\\
& W_n \in \{0, 1\}^M && n = 1, ..., M && (2.47)
\end{aligned}
$$

The constraint (2.42) says that the total number of open and closed stacks at instant n, counted by the left hand side, is less than or equal to the maximum number of open stacks plus the number of stacks that have been closed before. The constraint (2.43) determines that, when a stack i ends, then every item j that appears with item i in a pattern will also form a stack.

2.2.6 Baptiste's MIP formulation for the Constraint Modeling Challenge 2005

In 2005 the 5th Workshop on Modeling and Solving Problems with Constraints took place at IJCAI 2005 in Scotland. This workshop promoted a challenge in constraint modeling. The problem to be addressed by the challenge entrants was the minimization of open stacks problem. This challenge brought many different approaches to the MOSP, and although most of them used pure constraint programming, there were also other interesting approaches, such as local search or dynamic programming, and there was one entrant that used integer programming.

Philippe Baptiste is one of the participants of the Constraint Modeling Challenge. He has developed a mixed integer programming (MIP) formulation for MOSP [4], to find a lower bound, and a local search method to calculate an upper bound, that uses binary variables to assign each pattern to a sequence position and to indicate if each item starts before, ends after or is in process at the given position in the sequence. The model includes constraints referring to the start and end variables and the linking of patterns and items.

Consider

$$K_{cp} = \begin{cases} 1 & \text{if item } c \text{ is cut from pattern } p \\ 0 & \text{otherwise} \end{cases}$$

The general assignment variables are

$$x_{pt} = \begin{cases} 1 & \text{if pattern } p \text{ is sequenced at position } t \\ 0 & \text{otherwise} \end{cases}$$

There are customer variables referring to the orders starting, ending, or being in progress at a certain position

$$s_{ct} = \begin{cases} 1 & \text{if stack for item } c \text{ starts before or at position } t \\ 0 & \text{otherwise} \end{cases}$$

$$e_{ct} = \begin{cases} 1 & \text{if stack for item } c \text{ ends after or at position } t \\ 0 & \text{otherwise} \end{cases}$$

$$i_{ct} = \begin{cases} 1 & \text{if stack for item } c \text{ is open at position } t \\ 0 & \text{otherwise} \end{cases}$$

And finally there is the stack variable

$$\sigma \in \{0, ..., m\}$$

that represents the number of open stacks in the solution.

The mathematical model is

$$\begin{aligned}
&\text{Minimize} \quad \sigma && && (2.48)\\
&\text{subject to} \quad \sum_{p=0}^{n-1} x_{pt} = 1 && \forall t \in \{0,...,n-1\} && (2.49)\\
&\sum_{t=0}^{n-1} x_{pt} = 1 && \forall p \in \{0,...,n-1\} && (2.50)\\
&\sum_{c=0}^{m-1} i_{ct} \leq \sigma && \forall t \in \{0,...,n-1\} && (2.51)\\
&s_{ct} \geq s_{ct-1} && \forall c \in \{0,...,m-1\} \;\; \forall t \in \{0,...,n-1\} && (2.52)\\
&e_{ct} \leq e_{ct-1} && \forall c \in \{0,...,m-1\} \;\; \forall t \in \{0,...,n-1\} && (2.53)\\
&i_{ct} = s_{ct} + e_{ct} - 1 && \forall c \in \{0,...,m-1\} \;\; \forall t \in \{0,...,n-1\} && (2.54)\\
&x_{pt} \leq i_{ct} && \begin{array}{l}\forall c \in \{0,...,m-1\} \;\; \forall p \in \{0,...,n-1\}\\ \text{such that } K_{cp} = 1, \forall t \in \{0,...,n-1\}\end{array} && (2.55)\\
&x_{pt} \in \{0,1\} && \forall p \in \{0,...,n-1\} \;\; \forall t \in \{0,...,n-1\} && (2.56)\\
&s_{ct}, e_{ct}, i_{ct} \in \{0,1\} && \forall c \in \{0,...,m-1\} \;\; \forall t \in \{0,...,n-1\} && (2.57)
\end{aligned}$$

The constraints (2.49) mean that we can only have one pattern at a time, (2.50) force all patterns to be sequenced and (2.51) that at any position t, the number of open stacks is bounded. The restrictions (2.52), (2.53) and (2.54) guarantee the coherence in the sequence of opening and closing stacks. The last ones link the sequence of patterns to the stacks that are open.

Some cuts are made to tighten e and s variables, observing that if an item c appears in q patterns,

$$q = \sum_{p=0}^{n-1} K_{cp}$$

hence $e_{c(q-1)} \geq 1$ and $\forall t \geq q, e_{ct} \geq 1 - s_{c(t-q)}$, hence $s_{c(n-q)} \geq 1$ and $\forall t < n - q, s_{ct} \geq 1 - e_{c(t-q)}$.

Redundant constraints on aggregated items are added, as well as additional constraints to break symmetry, but Baptiste found that his tentatives to remove symmetries were complex and ineffective for this MIP approach.

2.2.7 Comparison between some of these models

In this section we will point out advantages and disadvantages of these models of MOSP and compare lower bounds obtained from them.

In [66], the model extended to the MOSP by Pinto from Laporte's MTSP model is compared to the model extended for the MOSP by Yanasse from the Tang and Denardo's model for the MTSP, and the results show that each of these two models performed better in half of the 20 instances tested, although the model from Laporte et al. extended for the MOSP presents a greater variance of runtime. In that paper, these models were also compared with a new integer model that outperforms both in most of the instances tested.

The model of Laporte, Gonzalez and Semet with the MOSP approach of Pinto, shows that the number of variables in the linear relaxation is $O(n^2 + nm)$, although the number of constraints is exponential. With the additional constraints (2.38) to (2.40), the value of the linear relaxation rises from 2.91 to 6.0, for an instance where the optimal solution is 7.0.

The model of Baptiste based on MIP with cuts optimally solves medium size instances. The disadvantages are that the lower and upper bounds have worse behavior than with specific cuts (it causes either an infeasible or an optimal solution). Unfortunately he was not able to break many symmetries with cuts to MIP.

2.3 Other approaches and special cases

The minimization of open stacks problem and the other related pattern sequencing problems have been addressed using other techniques, such as heuristics, genetic algorithms and dynamic programming. Some special cases of the MOSP problem have been specifically studied to help tackle the problem.

2.3.1 Yanasse's work on special cases of the MOSP

Yanasse is probably the researcher who has published more papers concerning this problem. Using a graph representation of the problem, he has proved that there are polynomial algorithms that solve some special cases of the MOSP, has presented some very good heuristics based on the properties of the corresponding graphs and has even tried some integrated approaches of both generating and sequencing the cutting patterns problems.

MOSP in a graph perspective

A MOSP problem where there are at most two different panel types per pattern can be represented through a graph that associates nodes to orders and arcs to patterns [64]. By making each piece type (or customer order) correspond to a vertex, and considering two vertices to be adjacent if and only if the corresponding piece types are simultaneously present in a pattern (or if the corresponding orders include the same product) we obtain a *MOSP graph*.

This type of MOSP problems can be seen as a problem of traversing arcs of the graph. A feasible solution corresponds to a sequence of arcs. A stack j is opened the first time that an arc incident to vertex j is traversed, and closed when all arcs incident to j have already been sequenced. In a MOSP graph where all nodes have degree n or more, there must be at least $n+1$ open stacks at some time during cutting, making $n+1$ a lower bound for the solution [64]. This also links the MOSP with the maximum clique problem, because the size of the maximum clique in a MOSP graph is then a lower bound for MOSP. The maximum clique problem will be discussed in section 3.1.2.

The condition of having at most two different panel types per pattern represents no loss of generality, because a solution of the general case can be transformed in a solution of the first case in polynomial time (and vice-versa).

Given a general MOSP problem, a MOSP graph can be obtained by introducing a clique of size k for each pattern composed by k panel types. By analogy to each arc in the clique, this pattern can be divided in subpatterns with at most 2 items in each, transforming it in a MOSP graph corresponding to a problem where there are at most 2 items per pattern. In [64], Yanasse proves that the maximum number of stacks in both problems is the same.

Polynomial Algorithms for special cases

For the cases of the MOSP where the items being produced belong to at most two different clients and the corresponding graph is a tree, a 1-tree, or a 0-1 common vertex polygon, there are polynomial algorithms that solve the problem [61]. A *1-tree* is a connected graph containing exactly one cycle, and a *0-1 common vertex polygon* is a graph that is constituted by polygons such that any two polygons can have at most one vertex in common.

For trees where each node has degree at most two, which corresponds to a graph which is a path with start and end vertices, the strategy is to start with a node with degree one and then sequence the remaining vertices in order to finish any open stacks. In a cycle the same process can be used if a vertex is chosen randomly to be the first stack. For a general tree, or a 0-1 common vertex polygon, a solution is built by appending together optimal solutions of its subparts and making sure that the number of open stacks is kept to the least possible. The computational complexity of this algorithm is bounded by $\mathcal{O}(n^2)$, where n is the number of patterns [62].

The algorithms presented for these cases are polynomial, however they give no guarantee that the solution obtained by amending the parts is optimal. In fact, in [44] the MOSP is proven to be NP-hard in general.

Preprocessing

The purpose of this phase is to eliminate redundancies and to detect conditions for which a solution can be found in polynomial time. In a preprocessing phase, all components that are trees in a MOSP problem should be removed, because, as has been discussed, it is possible to order these components in polynomial time, and incorporate them later into the final solution.

In a recent paper [67], Yanasse and Senne reviewed some preprocessing operations that can reduce the size of the MOSP problem by finding dominance or equivalence in the patterns.

We say that a pattern P_j is *dominated* by another pattern P_i, when all items that are contained in pattern P_j are also contained in pattern P_i. In that case, Becceneri et al. [6] proved that we can remove P_j in the search and insert it in the solution immediately after P_i without increasing the number of open stacks. To identify dominance between all patterns requires an effort of at most $\mathcal{O}(n^2m)$.

A *pseudo-equivalence* appears in a MOSP graph if there are two different nodes i and j such as i has exactly the same neighbors of j, possibly including themselves. If this is the case, then there is always an optimal solution of the MOSP where i and j are closed one immediately after the other [67].

When considering pseudo-equivalence with closed neighborhoods (see Section 3.1 for a definition), nodes i and j can be considered as if they were a single node, opening together and closing together, reducing the number of items in the instance.

When considering simple neighborhoods, it allows to reduce the number of patterns. A pattern that contains item i and a pattern that contains item j can be considered as a single pattern, in the sense that it is possible to construct a new sequence with these patterns appearing consecutively, whose maximum number of open stacks is less than or equal to the original sequence.

Identifying pseudo-equivalence has a complexity of $\mathcal{O}(m^3)$.

2.3.2 Heuristics

The first works on pattern sequencing problems addressed the MORP problem. For this problem, Dyson and Gregory [19] developed heuristics based on the Travelling Salesman Problem. Later, the MOSP became the most popular pattern sequencing problem and it has been solved with a profusion of heuristics. We will only present the most relevant ones in this section.

Yuen and Richardson classic heuristics for the MOSP

Yuen and Richardson [69] introduced the minimization of open stacks problem (MOSP) and presented six heuristics for it. Because of their good practical behavior in processing time and simplicity, these heuristics were for a long time considered a classic reference.

The first heuristic presented finds a Lower Bound for the maximum number of open stacks. As all piece types of a pattern will form incomplete stacks while this pattern is cut, we can say that a lower bound is the largest number of piece types in any pattern.

The next heuristic uses graph theory to detect disjoint subgroups of patterns which have piece types exclusive to their group. This can be used to break the problem in several smaller problems, one for each group of patterns. Having that solved, the optimal final sequence is an arbitrary articulation of the optimal sequences in the subgroups. Each pattern is represented by a node of a graph and an arc joining two nodes means that there is at least one piece common in those patterns. The disjoint subgroups of patterns will be the disjoint connected components of the graph. The disjoint connected components can be found computing the n^{th}-power of the sum of the adjacency matrix of the graph with the corresponding identity matrix.

The third heuristic uses an Upper Bound (UB) and incorporates backtracking. The initial value for the Upper Bound is best chosen from the best solution given by one of the six heuristics. A tree search is made to find the optimum queue number. When we identify a node that has a number of open stacks equal to or exceeding the upper bound, we must exclude that node and all its subnodes, as they will not improve the upper bound. If it is so, we go back up in the tree to the next valid unsearched subsequence (this is called back tracking). The upper bound is revised when a complete sequence of patterns has been traversed and its maximum number of open stacks is less than the current Upper Bound. In that case the Upper bound is tested against the Lower Bound and if they are equal then the optimum has been found.

A fourth heuristic tests if there are other nodes in the tree that have no common stacks with the current node. Those can be excluded from the current search because they will not improve the solution. In the first heuristic we have assured that all patterns are linked in the graph, so there is always a pattern with at least one piece type in common with the current open stacks.

The next heuristic consists in comparing the number of open stacks in a subsequence of patterns at the origin of the tree with the number obtained from reversing that subsequence of patterns. If the maximum number of open stacks in the reverse order is greater than or equal to that already searched, then the reversed subsequence needs not to be investigated.

The last heuristic rearranges the patterns accordingly to their involvement with other patterns. If we consider

$$\delta_{ij} = \begin{cases} 1 & \text{if piece } i \text{ is in pattern } j \\ 0 & \text{otherwise} \end{cases}$$

and

$$o_i = \sum_j \delta_{ij}$$

as the number of patterns in which piece i occurs, than for any pattern j, the involvement with other patterns can be measured by

$$P_j = \sum_i o_i \delta_{ij}$$

and the patterns are ordered by their ascending P_j values.

The third heuristic from Yuen and Richardson was, for over a decade, used as a reference to compare the performance of the new heuristics being generated for this problem.

A model for the MOSP by Becceneri, Yanasse and Soma

Becceneri, Yanasse and Soma [6] have presented heuristics and a branch-and-bound exact method for MOSP.

According to them, the problem should first be pre-processed by viewing the input data of an instance as a graph, removing all components that are trees, making a dominance analysis among patterns and finding cliques in its structure. The dominance relations between the nodes of the graph lead to a partial ordering of the stacks, determining the order in which they should be closed. After this analysis, the Heuristic of Minimal Cost Node should be applied, subsequently the Arc Contraction Heuristic to find lower bounds and, finally, a Branch-and-bound technique should be executed to obtain an optimal solution. Let us see in more detail how these procedures work.

Heuristic of minimal cost node

The Minimal Cost Node heuristic finds an order for the patterns by traversing the arcs of the graph using the least quantity of arcs to close a node. In this heuristic, all vertices are first ordered by non-decreasing order of the number of arcs needed to close that node and a search is made for adjacent nodes that have arcs not yet traversed.

While all arcs of the graph have not yet been sequenced:

- take the arc of the first two adjacent vertices
- close any node with no remaining arcs
- search for adjacent nodes that have arcs not yet traversed and sequence those arcs

This heuristic generates a sequence of arcs and a maximum number of open stacks ξ'.

Arc contraction heuristic

Now we check the minor of the graph for equivalent nodes (nodes that have the same neighborhood). According to the authors, if a problem has a feasible solution, it is possible to construct another solution where the equivalent nodes appear consecutively and the maximum number of open stacks is less than or equal to the original sequence. The Arc Contraction heuristic finds equivalent nodes and performs a contraction of the arcs with vertices which have the smallest indexes. It also calculates a lower bound for MOSP, based on identifying the minor as a clique. This lower bound for MOSP is the maximum of the following numbers:

- Maximum number of pieces in any pattern

$$lbp = \text{Max}\,\{|P_j| : j = 1, \ldots, n\}$$

- Open stacks for all pieces with a common pattern with v

$$lbd = 1 + \text{Min}\,\{\text{degree}\,(v) : v \text{ is a vertex of } G_p\}$$

- Identifying the minor G_p as a clique

$$lbc = \text{Max}\,\{i : K_i \text{is a minor of } G_p\}$$

While the minor G_p is not a clique and $|G_p| > lb = \text{Max}\,\{lbp, lbd\}$

1. sort the vertices in a non-decreasing order
2. check the minor of the graph for equivalent nodes (nodes that have the same neighbourhood)
3. contract the arc with vertices which have the smallest indices (delete one of that vertices and all arcs incident to it)
4. update the lower bound

This process should be applied until exhaustion, preserving the hierarchy of relations (the last ones to be contracted are the first to be added to the solution).

Branch and bound

Finally, this exact method executes a branch and bound approach on the minor obtained from the heuristics above in order to reach for an optimal solution. If $\xi' \neq lb$ then the equivalent nodes are deleted and with the heuristic of Arc Contraction a new lbc is determined. Then, a branch-and-bound on the minor G_p is executed to obtain an optimal solution. Finally, the deleted equivalent nodes and the vertices of D are inserted into the final sequence.

Ashikaga's heuristic based on Hamiltonian circuits

Another heuristic to solve the MOSP has been developed recently by Ashikaga [2]. It explores the fact that the MOSP graph has large cliques and high edge density and suggests an heuristic based on the extension-rotation algorithm of Pósa (1976) to obtain Hamiltonian circuits. An initial path for the algorithm is defined by the vertices that form a maximal clique, which is built by finding iteratively in the complement graph a vertex with minimum degree and removing its neighborhood.

Comparison of heuristics for the MOSP

Compared to the heuristics proposed by Yuen and Richardson, the model of Becceneri, Yanasse and Soma was able to find better solutions with the Smaller Cost Node heuristic than with Yuen3, but it needs much larger computational effort, yet being less than 5 minutes. For matrices with a large number of 1's in each line of the matrix (C) the quality of the lower bound/solution is quite good, but for smaller values of C the gap is relatively large. This model visits less nodes than non optimized branch and bound and uses significantly less time (some cases 7 times faster). However, the arc contraction heuristic may produce different bounds depending on the nodes chosen for contraction.

Ashikaga's heuristic has been compared with these heuristics and it is the fastest available presently. According to the author, this heuristic dominates the other two heuristics in terms of mean errors, possesses smaller complexity order and it has a simpler implementation.

2.3.3 Dynamic Programming and Genetic Algorithms for the MOSP

The methods suggested for the MOSP include local search, some tentatives of applying genetic algorithms and a very successful model using dynamic programming. There is also a variety of constraint programming implementations submitted for the Constraint Modeling Challenge 2005 [56].

Foerster and Wäscher compared the performance of simulated annealing and a traditional 3-opt procedure in randomly generated instances of the MORP, and recommend the use of simulated annealing [23]. Fink and Voss studied several pattern sequencing problems and addressed the minimization of open stacks and the minimization of average order spread using heuristics based on tabu search and simulated annealing [21].

Authors Faggioli and Bentivoglio suggest the use of greedy heuristics combined with tabu search and generalized local search to solve the MOSP [20]. They have tested this methods in real instances from a furniture manufacturer.

Oliveira and Lorena applied a constructive genetic algorithm to the MOSP using 2-opt population training based on the greedy procedure of Faggioli and Bentivoglio [50, 49].

Banda and Stuckey won the Constraint Modeling Challenge with a dynamic programming approach to MOSP [3]. This method is highly effective because, by investigating only the minimum stacks for each subset of patterns, it reduces the raw search space from $|P|!$ to $2^{|P|}$, where P is the set of cutting patterns. In [11], Cambazard and Jussien suggest an improvement to the dynamic programming approach by using nogood recording schemes, which explore the structure of the MOSP problem in terms of inconsistent sequences of patterns. The nogood recording technique is also used in [13] by Chu and Stuckey, combined with a branch-and-bound strategy based on choosing which stack to close next and several pruning schemes, to create an exact solver for the MOSP that it is faster than the previous ones.

2.4 Conclusions

As we are specially interested in pursuing an integer programming approach to pattern sequencing problems, we analyzed more thoroughly the papers using this technique. The work of Yanasse [63] adapts to the MOSP a model originally thought for a similar problem (MTSP) by Tang and Denardo [57]. There is also an IP model by Laporte et al. [41] for the MTSP based on TSP, that is solved with linear relaxation, branch-and-cut and branch-and-bound, that later was adapted to the MOSP by Pinto [54]. And there is a MIP formulation by Baptiste [4] for the MOSP, submitted to the Constraint Modeling Challenge.

We discussed special properties in the structure of the MOSP problem that allow preprocessing procedures, such as trees in the MOSP graph, dominance between cutting patterns and pseudo-equivalency.

Some other methods for solving pattern sequencing problems were also briefly reviewed, as Yuen and Richardson's first heuristics for the MOSP [70], dynamic programming approaches by Banda and Stuckey [3], and the latest heuristic by Ashikaga [2] which is currently the fastest.

The inspection of the papers on this subject, like [44], suggested some relationships between the minimization of open stacks problem and graph layout problems. In the search for ideas that could be used to derive a new integer programming model for the minimization of open stacks problem, it became important to deepen the study of graph theory, namely the concepts behind perfect graphs and layout measures. In the next chapter the main results from this theory are recalled, along with interesting applications in Biology, Archeology and Computer Science.

Chapter 3

Interval Graphs and Linear Layout Problems

The main problems that are addressed on this thesis are solved here by using a special type of graphs called interval graphs. This chapter presents the main theory on interval graphs and discusses some problems concerning the layout of a graph that are somehow connected to the pattern sequencing problems.

3.1 Basic Graph Definitions

Let us start by recalling some definitions and notations that will be useful to study interval graphs.

Definition 3.1.1. A *graph* $G = (V, E)$ consists of a set V (that we call the set of *vertices* or *nodes*) and a set E of tuples from $V \times V$, i.e, $E = \{e = [vw] : v, w \in V\}$ (that we call *edges* or *arcs*).

An edge with both endpoints on the same vertex is called a *loop.* An edge $e = [uv]$ is a *multiedge* or *k-fold edge* if there exists exactly k edges $e_1, e_2, ..., e_k$ such that $e_1 = e_2 = ... = e_k = [uv]$. For $k = 2$, or $k = 3$, it is called a double edge, or a triple edge, respectively. We call a graph with no loops or multiedges a *simple graph.*

In this work, all graphs that will be used are assumed to be simple graphs.

3.1.1 Directed and Undirected Graphs

Definitions 3.1.2. A *directed graph* or *digraph* is a graph $G = (V, E)$ in which every edge in E may have a direction defined, so that E is a collection of ordered pairs of elements in V. In a directed(oriented) edge $[vw]$, the first vertex v is called the *tail* and the last vertex w is called the *head.* For a given vertex $v \in V$, the number of edges where v is a head is called the *indegree* and denoted by $indeg(v)$, and the number of edges where v is a tail is called the *outdegree* and denoted by $outdeg(v)$.

Definition 3.1.3. The graph $G^{-1} = (V, E^{-1})$ is said to be the *reversal* of G if

$$E^{-1} = \{[uv] \in V \times V : [vu] \in E\}$$

A graph $G = (V, E)$ is called *undirected* if $E = E^{-1}$.
A graph $G = (V, E)$ is called *oriented* if $E \bigcap E^{-1} = \varnothing$.

In this work, we will be mostly using undirected graphs; for this reason, from this point forward, if nothing is said otherwise, by graph we mean an undirected graph and we will use without distinction $[uv]$ and $[vu]$.

Definition 3.1.4. Two vertices $v, w \in V$ are *adjacent* if $[vw]$ is an edge. Two edges are *adjacent* if they share a common vertex.

The neighborhood of a vertex is the set of its adjacent vertices. As we are considering simple graphs, in which we do not allow edges from a vertex to itself, one vertex does not belong to its own neighborhood. So it is usual to also define the closed neighborhood of a vertex if we want to include it.

Definition 3.1.5. For a given vertex $u \in V$, we define the *neighborhood* or *adjacency set* of u as:

$$N(u) = \{v \in V : [uv] \in E\}$$

and define the *closed neighborhood* of v as:

$$N[u] = \{u\} \bigcup N(u)$$

We can also name the set of vertices that are not adjacent to any neighbors of a given vertex.

Definition 3.1.6. For a given vertex $u \in V$, we define the *anti-neighborhood of u* as:

$$\overline{N(u)} = V \backslash \left(N(u) \bigcup \{u\} \right)$$

We can also extend this definition to the *closed anti-neighborhood*:

$$\overline{N[u]} = \overline{N(u)} \bigcup \{u\}$$

and define the *anti-neighborhood of a set of vertices* $U \subseteq V$:

$$\overline{N(U)} = \{v \in V : [uv] \notin E \ \forall u \in U\}$$

Definition 3.1.7. We define the *degree* of a vertex $v \in V$ to be the number of times that v is an endpoint of an edge. A graph is said *k-regular* if every vertex has degree k.

In a simple graph, the degree of a vertex v is also the cardinality of $N(v)$.

Definition 3.1.8. The *complement* of G is the graph $\overline{G} = (V, \overline{E})$ where

$$\overline{E} = \{[uv] \in V \times V : u \neq v \wedge [uv] \notin E\}$$

Hence the complement graph is formed by the vertices together with the missing edges. When a graph has no missing edges, it is called a complete graph:

Definition 3.1.9. A graph is *complete* if every pair of distinct vertices are connected by one edge.

The complete graph on n vertices is usually denoted by K_n. As it contains all possible edges such that the graph is simple, the number of edges is $\binom{n}{2}$.

3.1.2 Problems with Cliques and Stable Sets

There are several problems that are commonly studied in the field of Graph Theory, some of which we will find useful to apply to the MOSP problem. Let us recall the most relevant ones.

Maximum Clique Number

This is a common graph problem that consists in finding the maximum number of vertices that form a clique in a graph.

Definitions 3.1.10. A *clique* is a set C of vertices of a graph G such that all pairs of vertices in C are adjacent.

A clique C is *maximal* if there is no clique of G which properly contains C as a subset. A clique is *maximum* if there is no clique of G of larger cardinality.

The size of the maximum clique in a graph G is called the *clique number* and it is denoted by $\omega(G)$.

Minimum Clique Cover

This problem consists in finding a set of cliques to cover all the vertices in the graph.

Definition 3.1.11. A *clique cover* of a graph G is a set of cliques such that every vertex in G belongs to one clique.

The size of the minimum clique cover is called the *clique cover number* and it is denoted by $k(G)$.

Maximum Independent Set

For this problem we are interested in finding a set with the maximum number of nonadjacent vertices.

Definition 3.1.12. A *stable set* or *independent set* of a graph G is a subset I of vertices such that no edge has both endpoints in I.

The number of vertices in a stable set of maximum cardinality is called the *stability number* and it is denoted by $\alpha(G)$.

Given a graph $G = (V, E)$ and a subset S of V, S is a clique of G if and only if S is a stable set of $\overline{G}$.

As a consequence of this, for any graph G, we have:

$$\omega(G) = \alpha(\overline{G})$$

Most of the methods for finding the clique number of a graph are based on finding independent sets in the complement graph.

There are more general relationships between these graph measures. For example, it is valid that

$$\alpha(G) \leq k(G)$$

since every vertex of a maximum stable set must be contained in a different partition segment in any minimum clique cover.

Chromatic Number

This is the famous problem of determining how many different colors are necessary to color the vertices of a graph using different colors for adjacent vertices.

Definition 3.1.13. A *proper k-coloring* of $G = (V, E)$ is a partition of the vertices $V = X_1 \cup X_2 \cup ... \cup X_k$ such that each X_i is a stable set.

In a proper k-coloring there is an assignment of integers $\{1, 2, ..., k\}$ (corresponding to k different colors) to the vertices V such that for any edge the two endpoints have been assigned different colors.

The smallest number k such as G has a k-coloring is called the *chromatic number* of G and it is denoted by $\chi(G)$.

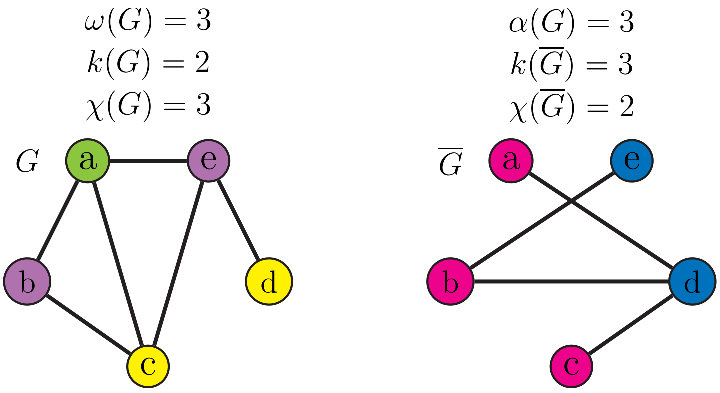


Figure 3.1: Chromatic number, stability number, clique cover number and clique number of a graph and its complement graph

As a stable set in a graph G corresponds to a clique in the complement graph $\overline{G}$, we have, for any graph G, the equality:

$$\chi(G) = k(\overline{G})$$

For any graph G, there is also a lower bound for the chromatic number

$$\omega(G) \leq \chi(G),$$

because if it contains a clique of size k then we need at least k different colors to color the vertices in that clique.

For general graphs the minimum coloring problem is NP-complete [26]. However, for a special type of graphs that we will see further ahead called interval graphs, linear time algorithms are known for coloring interval graphs with a minimum number of colors.

3.2 Chordal Graphs

Chordal graphs will play an important role in the models that we will present for solving the MOSP.

Definitions 3.2.1. A *path* is a sequence of vertices $[v_0, v_1, ..., v_k]$ such that $[v_{i-1}v_i]$ is an edge for $i = 1, ..., k$ and its *length* is the number of edges in the sequence. If no vertex is repeated, it is called a *simple path.*

A graph is *connected* if there exists a path from any vertex to any other vertex in the graph. A *connected component* of a graph is a maximal subgraph that is connected.

Longest path

A common graph problem is to find the longest simple path in G, i.e., a path with the maximum number of edges and such that no vertex appears on it twice.

Definitions 3.2.2. A path that begins and ends at the same vertex is called a *cycle*. If no vertex occurs more than once, the cycle is called a *simple cycle.* A graph without any cycle is called a *forest.* A graph with n vertices is a *tree* if it is a forest and it has exactly $n - 1$ edges.

A simple cycle $[v_0, v_1, ..., v_k, v_0]$ is said to be *chordless* if $[v_i v_j] \notin E$ for i and j differing by more than 1 mod $k + 1$. The chordless cycle on n vertices is usually called a *n-cycle* and denoted by C_n.

Definition 3.2.3. A graph is a *chordal graph* if it does not contain an induced k-cycle for $k \geq 4$.

The name “chordal”comes from the fact that in every simple k-cycle with $k \geq 4$ that may exist in this graph, there must be a *chord*, which is an edge between two non-consecutive vertices of the cycle.

Because of its geometric properties, these graphs are also called *triangulated* graphs.

Being chordal is a hereditary property inherited by all the induced subgraphs of G.

3.2.1 Perfect Elimination Order

Chordal graphs can be recognized by finding a special type of vertices and applying an iterative procedure to its induced subgraphs.

Definition 3.2.4. A vertex $v \in V$ is called *simplicial* if its neighborhood $N(v)$ induces a complete subgraph of G, i.e. $N(v)$ is a clique (not necessarily maximal).

From Dirac (1961), as cited in [28], it is known that simplicial vertices appear in all chordal graphs:

Theorem 3.2.5. *Every chordal graph G has a simplicial vertex and if G is not a complete graph then it has two nonadjacent simplicial vertices.*

Definition 3.2.6. Given a graph $G = (V, E)$, such that $|V| = N$, a *linear ordering* of the vertices is a bijective function $\varphi : V \to \{1, ..., N\}$. The *reversed linear ordering*, $\varphi^R : V \to \{1, ..., N\}$, is a linear ordering such that $\varphi^R(u) = N - \varphi(u) + 1$.

A linear ordering of the vertices of a graph is sometimes called a *layout* of the graph, a *numbering*, a *linear arrangement* or a *labeling* of the vertices. We will also use the symbol $\prec$ to express the linear ordering on the set of vertices.

Definitions 3.2.7. Given a graph $G = (V, E)$ and a linear ordering φ of its vertices, we say that vertex i *precedes* vertex j, and denote by $i \prec j$, if $\varphi(i) < \varphi(j)$. We denote the set of predecessors of a vertex by $Pred(i) = \{j \in N(i) : \varphi(j) < \varphi(i)\}$ and the set of successors by $Succ(i) = \{j \in N(i) : \varphi(j) > \varphi(i)\}$.

This ordering of the vertices can also lead to edge directions, directing the edge $[ij]$ if $i \prec j$. If the graph is simple then $|Pred(v)| = indeg(v)$ and $|Succ(v)| = outdeg(v)$, where $|.|$ designates the cardinality of the set.

Definition 3.2.8. A linear ordering $\sigma = [v_1, v_2, ..., v_n]$ of the vertices of a graph $G = (V, E)$ is called a *perfect elimination scheme* (or *p.e.s.*) if each v_i is a simplicial vertex of the induced subgraph $G_{v_i, \ldots, v_n}$.

A simplicial vertex can start a perfect elimination scheme, or start a similar linear ordering called a perfect elimination order:

Definition 3.2.9. [7] A *perfect vertex elimination order* (or *p.e.o.*) is a linear ordering of the vertices of the graph in which the sets of predecessors of each vertex $Pred(i)$ form a clique, $\forall i \in V$.

In a perfect elimination scheme, each of the sets $Succ(i)$ are complete sets. In a perfect elimination order, the sets $Pred(i)$ are complete. This means that if φ is a perfect elimination order, then the reversed ordering φ^R may not be a perfect elimination order as well, but it will be a perfect elimination scheme. The reason for this is because the set of predecessors of a vertex for a given linear ordering is the set of successors of that vertex for the reversed linear ordering, since:

$$\varphi^R(j) > \varphi^R(i) \Leftrightarrow N - \varphi(j) + 1 > N - \varphi(i) + 1 \Leftrightarrow \varphi(j) < \varphi(i)$$

Definition 3.2.10. A subset $S \subset V$ is a *vertex separator* for nonadjacent vertices a and b (or an (a, b)*-separator*) if the removal of S from the graph separates a and b into distinct connected components. If no proper subset of S is an (a, b)-separator, then S is a *minimal vertex separator* for a and b.

All the minimal vertex separators of a chordal graph are cliques [28]:

Theorem 3.2.11. *Let G be an undirected graph. The following statements are equivalent:*

(i) *G is chordal*

(ii) *G has a perfect vertex elimination scheme. Moreover, any simplicial vertex can start a perfect scheme.*

(iii) *Every minimal vertex separator induces a complete subgraph of G.*

The equivalence of items (i) and (ii) is due to Fulkerson and Gross [24]. Chordal graphs can thus be characterized by the existence of a perfect elimination scheme. This gives origin to an iterative procedure to recognize chordal graphs: locating a simplicial vertex and eliminating it from the graph, then locating a new simplicial vertex and eliminating it too, and by repeatedly doing this, at the end no vertices remain. If at some stage there are no more simplicial vertices it means that the graph is not chordal.

As cited in [28], this procedure was used by Lueker (1974) and Rose and Tarjan (1975) to write a linear time algorithm to recognize chordal graphs.

Theorem 3.2.12. *Chordal graphs can be recognized in linear time.*

3.2.2 Split graphs

Definition 3.2.13. An undirected graph $G = (V, E)$ is a *split graph* if there is a partition $V = S + K$ of its vertex set into a stable set S and a complete set K.

There is no restriction on edges between vertices of S and K and the partition may not be unique. Since a stable set of G is a complete set of $\overline{G}$, we have that G is a split graph if and only if $\overline{G}$ is a split graph. But the reason for mentioning this type of graphs is the existence of a strong relation between split graphs and chordal graphs, due to Földes and Hammer (1977), as quoted in [28].

Theorem 3.2.14. *For an undirected graph G the following conditions are equivalent:*

(i) *G is a split graph*

(ii) *G and $\overline{G}$ are chordal graphs*

(iii) *G contains no induced subgraph isomorphic to $2K_2, C_4$, or C_5.*

3.3 Comparability Graphs

Comparability graphs are a special case of graphs that can be transitively oriented.

Definition 3.3.1. A *comparability graph* is an undirected graph $G = (V, E)$ in which each edge can be assigned a one-way direction in such a way that the resulting oriented graph (V, F) satisfies:

$$[uv] \in F \wedge [vw] \in F \Rightarrow [uw] \in F \ \forall u, v, w \in V$$

This transitive orientation is acyclic, i.e., a comparability graph does not contain any directed cycle. With the orientation fixed, a comparability graph is also called a *partially ordered set* or *poset.*

Definition 3.3.2. A graph G is said a *co-comparability graph* if $\overline{G}$ is a comparability graph.

An undirected graph that is simultaneously a comparability and a co-comparability graph is a special type of graph called *permutation graph.*

Theorem 3.3.3. *[28] Comparability graphs can be recognized in linear time.*

3.4 Interval Graphs

In this section we recall Golumbic's [28] definition of an interval graph and some other concepts and theorems that will be used in our model.

Definition 3.4.1. An *interval graph* is an undirected graph G such as its vertices can be put into a one-to-one correspondence with a set of intervals I of a linearly ordered set (like the real line) such that two vertices are connected by an edge of G if and only if their corresponding intervals have nonempty intersection. I is called an *interval representation* for G.

Graphs which represent intersecting intervals on a line are an useful concept for us because if we associate each open stack of our MOSP problem to an interval in the real line (the interval of time that the stack stays open), then we can associate a solution of the MOSP to an interval representation of an interval graph.

Being an interval graph is a hereditary property, i.e., an induced subgraph of an interval graph is an interval graph. The decision of whether a given graph is an interval graph can be carried out in linear time.

Besides the definition, there are some theorems that we can use to characterize interval graphs. Lekkerkerker and Boland (1962) characterization [42] focusses on the fact that an interval graph cannot branch into more than two directions nor circle back onto itself.

Theorem 3.4.2. *[28] An undirected graph G is an interval graph if and only if the following two conditions are satisfied:*

(i) *G is a chordal graph and*

(ii) *any three vertices of G can be ordered in such a way that every path from the first vertex to the third vertex passes through a neighbor of the second vertex.*

The last sentence in this theorem introduces the concept of asteroidal triple:

Definition 3.4.3. An independent set of three vertices is an *asteroidal triple* (AT) if between each pair of vertices in the triple there is a path that avoids the neighborhood of the third vertex. A graph is *asteroidal triple free* or *AT-free* if it contains no asteroidal triple.

Using this terminology, Theorem 3.4.2 says that a graph is an interval graph if and only if it is chordal and AT-free. The AT-free structure of interval graphs has been used to develop a linear time algorithm to recognize interval graphs [15].

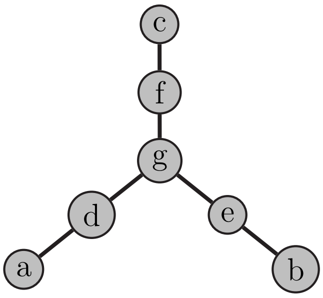


Figure 3.2: Not an interval graph

The graph in Figure 3.2 is not an interval graph because it contains an asteroidal triple: the vertices a, b, c. A path from a to b is a, d, g, e, b which avoids $N(c) = f$.

Some other characterizations of interval graphs are known, namely the following theorem (Gilmore and Hoffman, 1964) as cited in [28]:

Theorem 3.4.4. *Let G be an undirected graph. The following are equivalent:*

- *G is an interval graph*
- *G is chordal and $\overline{G}$ is a comparability graph*
- *The maximal cliques of G can be linearly ordered such that, for every vertex v of G, the maximal cliques containing v occur consecutively.*

This theorem remarks that the complement of an interval graph is a comparability graph. However, the reverse does not hold, i.e., the complement of a comparability graph is not always an interval graph.

The last sentence in theorem 3.4.4 is related to another characterization of interval graphs by Fulkerson and Gross (1965), which refers to the clique matrix of a graph.

Definition 3.4.5. The *clique matrix* of a graph is the incidence matrix of the maximal cliques versus the vertices of the graph. The entries of the clique matrix are of the form $a_{ij} = 1$ if vertex j belongs to the maximal clique i.

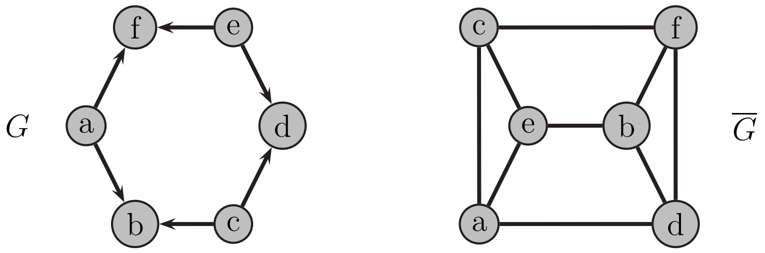


Figure 3.3: G is a comparability graph but $\overline{G}$ is not an interval graph

Definition 3.4.6. A matrix of zeros and ones is said to have the *consecutive 1's property for columns* if its rows can be permuted in such a way that the 1's in each column occur consecutively.

Theorem 3.4.7. *[24] An undirected graph G is an interval graph if and only if its clique matrix M has the consecutive 1's property for columns.*

This property is not only present in the clique matrix, but also in the adjacency matrix of an interval graph (Tarjan, 1976) as cited in [8].

Theorem 3.4.8. *$G = (V, E)$ is an interval graph if and only if there exists an ordering of G such that the associated adjacency matrix A verifies:*

$$\forall i \in \{1, ..., N\} \;\; a_{ij} = 1 \textit{ for } j = f_i(A), f_i(A) + 1, ..., i$$

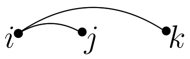
where $f_i(A) = min\{j : a_{ij} \neq 0\}$

We will also find it very interesting to use an alternative characterization of interval graphs by Olariu[15] that uses the linear ordering of the vertices as well.

Theorem 3.4.9. *A graph $G = (V, E)$ is an interval graph if and only if there exists a linear ordering $\varphi : V \to \{1, ..., N\}$ such that $\forall i, j, k \in V$: $\varphi(i) < \varphi(j) < \varphi(k)$ we have $\quad [ik] \in E \Rightarrow [ij] \in E$.*

This characterization is the one that we will use in our model for the MOSP to assure that the graph of the solution is an interval graph.

Definition 3.4.10. An edge $[ik]$ is called an *umbrella* if there is a vertex j such that $i \prec j \prec k$ and $[ij] \notin E$. A linear order that satisfies the condition of Olariu's characterization of interval graphs is said to be *umbrella free*.



The ordering of the maximal cliques in an interval graph referred to in theorems 3.4.4 and 3.4.9 will allow us to set an interesting ordering for the vertices, helped by the following theorems from [7].

Theorem 3.4.11. *An interval graph G has an interval representation such that all endpoints of intervals are distinct integers.*

Proof. Suppose there are $t \geq 2$ intervals with coincident endpoints at x. These endpoints of the intervals can be of four types:

- Intervals $I_1, ..., I_k$ have open right endpoints at x
- Intervals $I_{k+1}, ..., I_l$ have closed left endpoints at x
- Intervals $I_{l+1}, ..., I_s$ have closed right endpoints at x
- Intervals $I_{s+1}, ..., I_t$ have open left endpoints at x

In all cases replace the endpoint of interval I_i at x for an endpoint at $x + \frac{i}{t}\varepsilon$

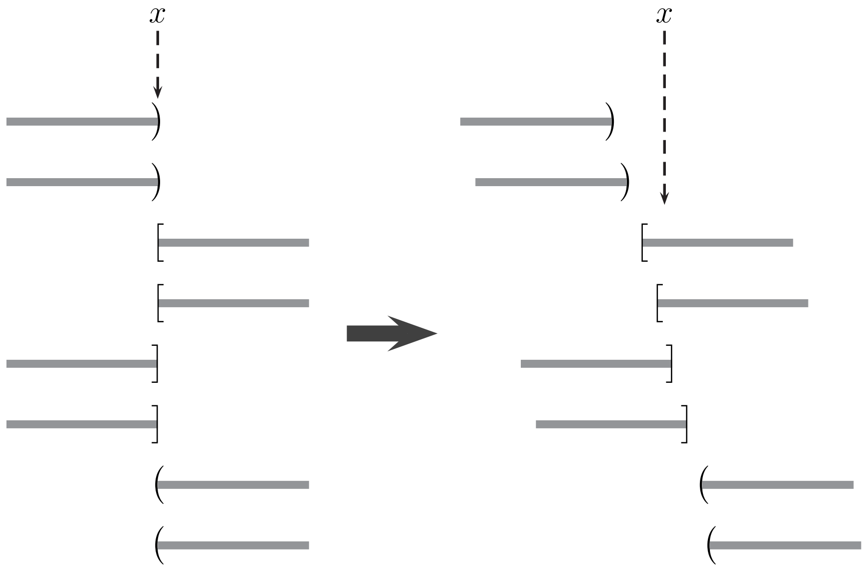


Figure 3.4: Replacing endpoints to remove coinciding endpoints [7]

Once all endpoints are distinct, sort the endpoints and assign them distinct integers from the set $1, 2, ..., 2n$ in sorted order. □

By using, for instance, the left endpoints of the intervals, we can define a natural ordering of the vertices of the graph: declare $i \prec j$ if the left endpoint of interval i precedes the left endpoint of interval j. Therefore, we can assign edge directions based on this vertex order, choosing to direct each edge from left to right, directing the edge $[ij]$ if $i \prec j$.

For an interval graph with an interval representation such that all endpoints are distinct, if the vertices are ordered by the left endpoint of the intervals, every maximal clique referred to in Theorem 3.4.4 occurs consecutively and has the form $Pred(u) \cup \{u\}$ for some vertex u.

But not every set $Pred(u) \cup \{u\}$ has to be a maximal clique:

Lemma 3.4.12. *[7] Let $v_1, ..., v_n$ be perfect elimination order. Then $C = Pred(v_i) \cup v_i$ is not a maximal clique if and only if there exists a successor v_j of v_i such that v_i is the last predecessor of v_j and $indeg(v_j) = indeg(v_i) + 1$.*

The perfect elimination order of the vertices of the graph can give origin to a linear ordering of the left endpoints of the intervals of an interval representation constructed for the graph.

Given an interval graph and some vertex v_i represented by an interval that starts at s_i, $Pred(v_i)$ is the set of all vertices representing intervals that start before s_i and do not end before s_i. Therefore, all these intervals contain the point s_i, and hence overlap each other, which means that $Pred(v_i)$ is a clique and therefore the vertex order is a perfect elimination order [7].

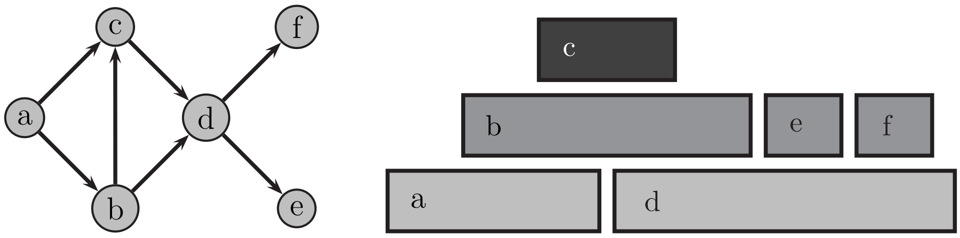


Figure 3.5: The p.e.o. a, b, c, d, e, f gives the linear ordering of the left endpoints of the intervals

The vertex ordering defined by the left endpoints of the intervals creates in fact the sequence of cliques that will appear in the interval graph, and that we are interested in finding, in order to discover the solution of a MOSP problem.

It is known that an interval graph H is chordal and it has at least two simplicial vertices where a perfect vertex elimination scheme can be started.

Locating a simplicial vertex and eliminating it will create another simplicial vertex and its subsequent elimination and so on.

A perfect elimination scheme is not appropriate for ordering the left endpoints of the intervals, as can be confirmed in Figure 3.6. In fact, the order in which the intervals must start can be set by following the reverse order of the eliminated vertices, which is a perfect elimination order.

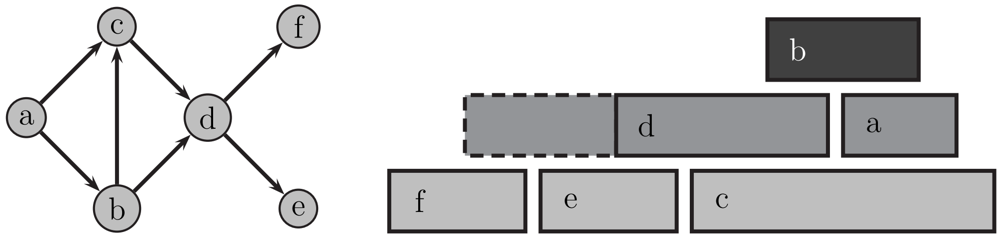


Figure 3.6: The p.e.s. f, e, d, c, b, a is not fit for ordering the left endpoints of the intervals

The reverse of a p.e.o. is always a p.e.s and vice versa. Generally, the reverse of a p.e.o. is not a p.e.o.; this is only true for proper interval graphs [51], which are interval graphs where no interval properly contains another.

Although every interval graph has a perfect elimination order, the reverse does not hold. For example, trees may not be interval graphs (for example Figure 3.2), but have a p.e.o. because they are chordal graphs.

3.5 Perfect Graphs

Interval graphs are part of a more general class of graphs beautifully called perfect graphs.

Definition 3.5.1. A graph $G = (V, E)$ is a *perfect graph* if it satisfies both the properties:

(i) $\omega(G_A) = \chi(G_A) \quad \forall A \subseteq V$

(ii) $\alpha(G_A) = k(G_A) \quad \forall A \subseteq V$

Actually, it is sufficient to show one of these properties, as the Perfect Graph Theorem (Lovász, 1972) implies that these two properties are equivalent.

Theorem 3.5.2. ***(Perfect Graph Theorem)*** *[28] A graph G is perfect if and only if its complement $\overline{G}$ is perfect.*

An odd length cycle is called an *odd hole* and its complement is called an *odd anti-hole*.

Conjecture 3.5.3. ***(Strong Perfect Graph Conjecture)*** *A graph is perfect if and only if it does not have an odd hole or an odd anti-hole as an induced subgraph.*

This conjecture was posed in 1961 by Claude Berge and it was proved by Chudnovsky, Seymour, Robertson and Thomas in 2003 [14].

Theorem 3.5.4. *[28] Every comparability graph G is a perfect graph.*

To see this let us define on the oriented graph $G = (V, F)$ the height function

$$h(v) = \begin{cases} 0 & \text{if } v \text{ is a sink} \\ 1 + \max\{h(w) : [vw] \in F\} & \text{otherwise} \end{cases}$$

A sink is a vertex of the oriented graph that has outdegree zero. This is always a proper coloring of the vertices of a graph. The number of colors used is equal to the number of vertices in the longest path of F.

If G is a comparability graph with a transitive orientation F, every path in F will correspond to a clique of G because of transitivity. So in this case, the height function will yield a coloring which uses exactly $\omega(G)$ colors, which is the best possible. As being a comparability graph is hereditary, the clique number and the chromatic number are also equal for all induced subgraphs of G. As the complement of an interval graph is a comparability graph, then interval graphs are perfect.

The computational complexity of finding perfect graphs is due to Cornuéjols, Li and Viskovic (2003), as quoted in [7]:

Theorem 3.5.5. *There is a polynomial time algorithm to test whether a graph is perfect.*

Many problems which are generally NP-hard, can be solved in polynomial time when circumscribed to perfect graphs.

Theorem 3.5.6. *[7] On a perfect graph, the problems Clique, Coloring and Maximum Independent Set can be solved in polynomial time.*

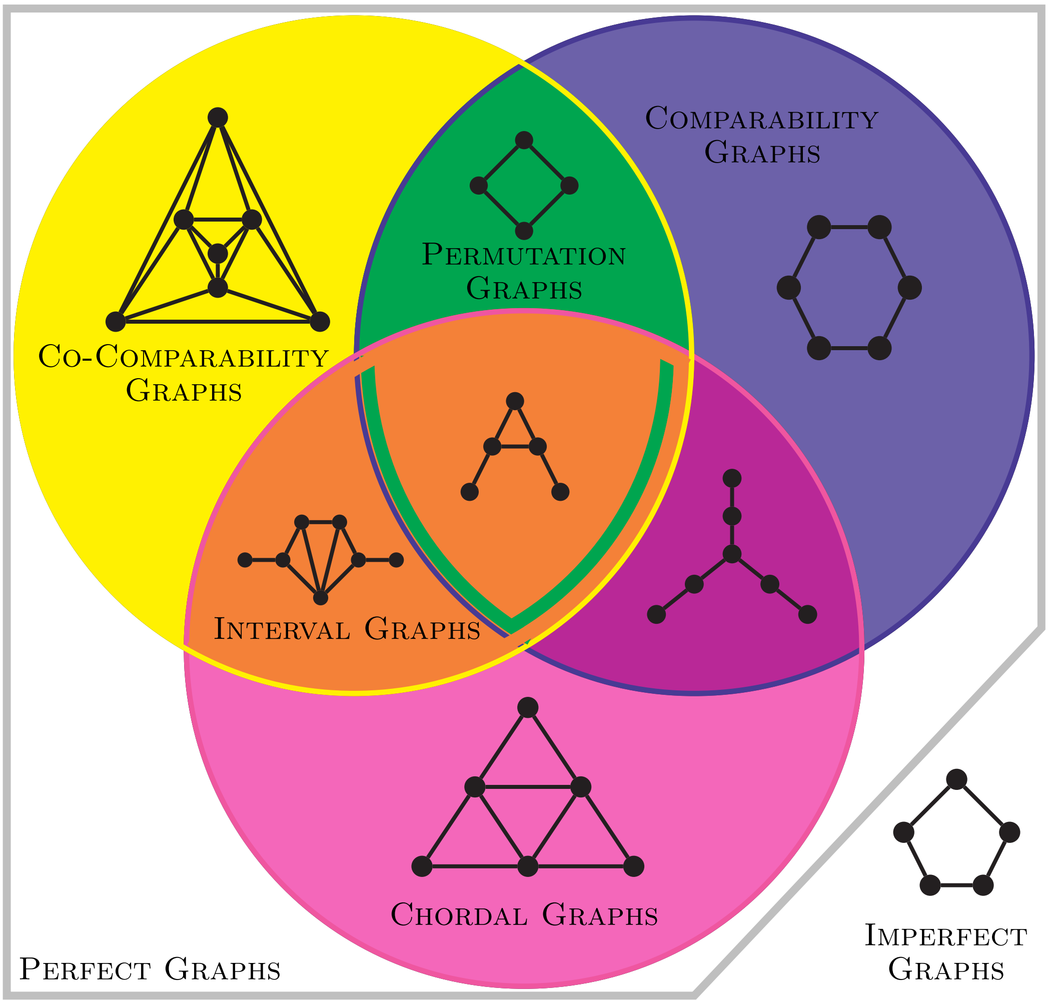


Figure 3.7: Intersection between subclasses of perfect graphs

3.6 Graph Layout Measures

The linear ordering of the vertices of a graph is also called the layout of the graph, because when the vertices are arranged by that ordering, there are several measurements that naturally can be taken and used to describe geometric properties of the graph.

3.6.1 Vertex Separation and Minimum Sum Cut

Imagine all vertices of the graph in a horizontal line, ordered from left to right by its linear ordering or layout φ. That is the motivation for defining, for each integer i, the sets of the vertices on the left and on the right of i.

Definition 3.6.1. The *vertex cut* or *separation* at position i of φ is

$$\delta(i,\varphi,G) = |\{u \in L(i,\varphi,G) : \exists v \in R(i,\varphi,G) : uv \in E\}|$$

where $L(i,\varphi,G) = \{u \in V : \varphi(u) \leq i\}$ and $R(i,\varphi,G) = \{u \in V : i < \varphi(u)\}$

The vertex cut counts the number of vertices on the left of i (including i) that are connected by an edge to any vertex on the right of i.

Definition 3.6.2. The *vertex separation* of a layout φ of a graph G is the maximum vertex cut:

$$VS(\varphi,G) = \max_{i\in\{1,\ldots,n\}} \delta(i,\varphi,G)$$

The *vertex separation problem* consists of finding a layout of G that minimizes the vertex separation.

Definition 3.6.3. The *sum cut* of a layout φ of a graph G is the sum of all vertex cuts:

$$SC(\varphi,G) = \sum_{i\in\{1,\ldots,n\}} \delta(i,\varphi,G)$$

The *minimum sum cut problem* consists of finding a layout of G that minimizes the sum cut.

3.6.2 Profile

Definition 3.6.4. The *profile* of a graph $G = (V,E)$ with a layout φ is

$$PR(\varphi,G) = \sum_{u\in V} \left(\varphi(u) - \min_{v\in N[u]} \varphi(v)\right)$$

where $N[u] = \{u\} \cup \{v \in V : [uv] \in E\}$.

The *profile minimization problem* consists of finding a layout of G with minimum profile.

Representing the reversed ordering of the vertices by φ^R, we have

$$PR(\varphi,G) = SC(\varphi^R,G)$$

because, in the reversed ordering φ^R, each vertex u contributes one unit $\varphi(u) - \underline{\min}_{v\in \overline{N(u)}} \varphi(v)$ times to the sum cut [18]. Hence Profile Minimization and Minimum Sum Cut are equivalent problems.

Definition 3.6.5. The *adjacency matrix* of a graph $G = (V, E)$ on n vertices is a $n \times n$ symmetric matrix A such that each row or column of A is a vertex of G and $[uv] \in E$ iff $a_{uv} \neq 0$.

Definitions 3.6.6. The *column height* P_j of column j of a matrix A is [40]

$$P_j = \begin{cases} 0 & \text{if } a_{ij} = 0 \text{ for } 1 \leq i \leq j \\ j - i & \text{if } i \text{ is the smallest integer such that } a_{ij} \neq 0 \end{cases}$$

The *profile of the matrix* A is

$$PR(A) = \sum_{j=1}^{n} P_j$$

and the *bandwidth of the matrix* A is

$$BW(A) = \max_{j=1,\ldots,n} P_j$$

The profile of a matrix A is equivalent to the profile of the graph G for which A is the adjacency matrix, and the bandwidth of a matrix A is equivalent to the bandwidth of the graph G for which A is the adjacency matrix.

Definition 3.6.7. The *envelope* of the adjacency matrix A of a graph is:

$$Env(A) = \{[ij] : f_i(A) \leq j < i\} \text{ where } f_i(A) = \min\{j : a_{ij} \neq 0\}$$

An adjacency matrix has a *full envelope* if

$$a_{ij} \neq 0 \ \forall [ij] \in Env(A)$$

If A is the adjacency matrix of the graph with the vertices sorted by the linear ordering, the total number of edges in the envelope of the matrix is equal to the profile of the graph.

3.6.3 Cutwidth

Definition 3.6.8. The *edge cut* at position i of φ is

$$\theta(i, \varphi, G) = |\{[uv] \in E : u \in L(i, \varphi, G) \wedge v \in R(i, \varphi, G)\}|$$

The edge cut counts the number of edges connecting a vertex on the left of i with a vertex on the right. If the vertex i itself is not included in the left set, we have the modified edge cut.

Definition 3.6.9. The *modified edge cut* at position i of φ is

$$\zeta(i,\varphi,G) = |\{[uv] \in E : u \in L(i,\varphi,G) \wedge v \in R(i,\varphi,G) \wedge \varphi(u) \neq i\}|$$

Definition 3.6.10. The *cutwidth* of a layout φ of a graph G is the maximum edge cut:

$$CW(\varphi,G) = \max_{i\in\{1,...,n\}} \theta(i,\varphi,G)$$

A graph has cutwidth at most k if its vertices can be ordered as $v_1,...,v_n$ such that every cut in this order crosses at most k edges. More precisely, for any $1 \leq j < n$ there can be at most k edges $[v_iv_l]$ such that $i \leq j < l$.

Definition 3.6.11. The *modified cutwidth* of a layout φ of a graph G is the sum of all modified edge cuts:

$$MC(\varphi,G) = \sum_{i\in\{1,...,n\}} \zeta(i,\varphi,G)$$

The *cutwidth problem* and the *modified cutwidth problem* consist of finding a layout of G that minimizes the cutwidth and the modified cutwidth, respectively.

Computing the cutwidth is NP-hard in general [18], but testing whether a given graph has cutwidth at most k can be done in linear time in n but exponential time in k.

3.6.4 Bandwidth

The bandwidth of a graph is the maximum length of an edge of the graph, when the length is measured using the linear ordering of the vertices.

Definition 3.6.12. Given a layout φ of a graph $G = (V,E)$ and an edge $[uv] \in E$, the *length* of $[uv]$ on φ is

$$\lambda([uv],\varphi,G) = |\varphi(u) - \varphi(v)|$$

Definition 3.6.13. The *bandwidth* of a layout φ of a graph G is the maximum length of any edge:

$$BW(\varphi, G) = \max_{[uv]\in E} \lambda([uv], \varphi, G)$$

The *bandwidth problem* consists of finding a layout of G that minimizes the bandwidth.

A graph G has *bandwidth at most k* if we can permute the vertices in such a way that all entries in the resulting adjacency matrix are within k positions of the diagonal.

For a graph G with n vertices and a linear ordering of the vertices φ, there are some known bounds for these layout measures.

$$\begin{aligned} SC(\varphi, G) \leq nVS(\varphi, G) \ &\Rightarrow\ \min SC(\varphi, G) \leq n \min VS(\varphi, G) \\ VS(\varphi, G) \leq nBW(\varphi, G) \ &\Rightarrow\ \min VS(\varphi, G) \leq \min BW(\varphi, G) \\ CW(\varphi, G) \leq \Delta(G)BW(\varphi, G) \ &\Rightarrow\ \min CW(\varphi, G) \leq \Delta(G) \min BW(\varphi, G) \end{aligned}$$

where $\Delta(G)$ is the maximum degree of a vertex of G.

3.6.5 Treewidth

The treewidth is a layout measure that counts the number of adjacent vertices of a given graph G that we can group together and replace each group by a vertex of a tree T appropriately built from G by connecting the vertices of the tree T that are covering the same vertices of the graph G.

Definition 3.6.14. A *tree decomposition* of a graph $G = (V, E)$ is a tree $T = (I, F)$ where each node $i \in I$ has a label $X_i \subseteq V$ such that:

- $\bigcup_{i\in I} X_i = V$. We say that *all vertices are covered.*
- For any edge $[vw]$ there exists an $i \in I$ with $v, w \in X_i$. We say that *all edges are covered.*
- For any $v \in V$ the nodes in I containing v in their label form a connected subtree of T. We call this the *connectivity condition.*

A given tree can be the tree decomposition of several different graphs. The graph implied by a tree decomposition is the graph obtained by adding all edges between vertices that appear in a common label.

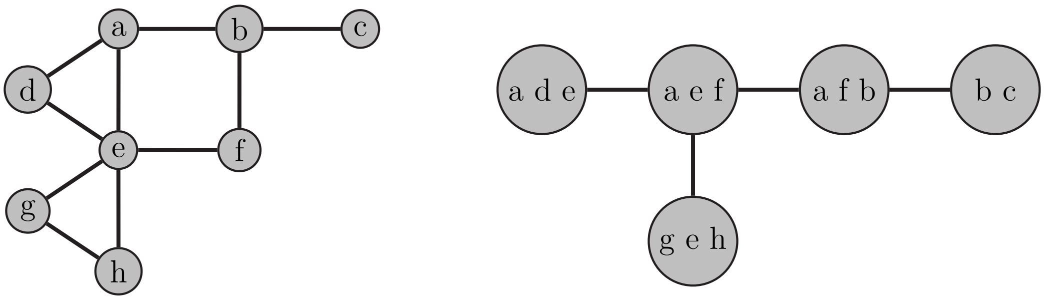


Figure 3.8: A graph with a tree decomposition of width 2 [9]

Definition 3.6.15. Given a tree decomposition $T = (I, F)$, the *width of the tree decomposition* is $\max\limits_{i \in I} |X_i| - 1$.

Definition 3.6.16. The *treewidth* of a graph G is the minimum k such that G has a tree decomposition of width k:

$$TW(G) = \min \left\{ \max_{i \in I} |X_i| - 1 : T = (I, F) \text{ is a tree decomposition of } G \right\}$$

The *treewidth problem* consists in, given $k \geq 0$ and a graph G, finding if $TW(G) \leq k$.

Notice that each clique in a graph must be part of at least one node in the tree decomposition, and hence the clique number minus one is a lower bound for treewidth. For this reason, all trees have treewidth 1.

Lemma 3.6.17. *[9] If G is chordal then G has a tree decomposition of width $\omega(G) - 1$.*

The minimum degree and the degeneracy of a graph are lower bounds for the treewidth:

$$\delta(G) \leq TW(G)$$

$$\delta D(G) = \max_{H \subseteq G} \delta(H) \leq TW(G)$$

Computing the treewidth is a NP-hard problem [1, 10].

3.6.6 Pathwidth

Definitions 3.6.18. A graph G has *pathwidth* bounded by k if G has a tree decomposition T of width k such that T is a path. This is also called a *path decomposition of width k*.

Computing the pathwidth is NP-hard in general but, for a given constant k, testing whether G has pathwidth bounded by k can be done in linear time.

From the definition, we immediately have $TW(G) \leq PW(G)$.

Any graph of bandwidth at most k also has pathwidth at most k.

3.7 Linear ordering problem

Another layout problem that comes up in a complete digraph $D_n = (V, E)$ on n nodes is the *Linear Ordering Problem* (LOP). This problem consists in finding a linear ordering of the vertices such that the number of directed edges in the graph that are not in accordance with this ordering is minimized. This can be done by associating a cost value to each directed edge. The linear ordering problem belongs to the class of NP-hard combinatorial optimization problems.

Definition 3.7.1. Given the complete digraph $D_n = (V, A_n)$ on n nodes, we define a *tournament* in A_n to be a subset of arcs of A_n containing, for every pair of nodes $i, j \in V$, either the arc $[ij]$ or $[ji]$ but not both. An *acyclic tournament* is a tournament without directed cycles.

An acyclic tournament corresponds to a linear ordering of the vertices and vice-versa, well-defined by the indegree of the nodes.

An integer programming formulation has been studied [30, 55, 22] for the linear ordering problem. The main variables are defined as:

$$x_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ precedes vertex } j \\ 0 & \text{otherwise} \end{cases}$$

The objective function is computed for all permutations π of the n vertices in the digraph with the costs defined by

$$c_{ij} = \begin{cases} 1 & \text{if arc } [ij] \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

The linear ordering problem has the following integer formulation:

$$\min_{\pi \in S(n)} \quad \sum_{i,j \in A} c_{ij} x_{ij} \tag{3.1}$$

$$\text{subject to} \quad x_{ij} + x_{ji} = 1 \qquad \forall i, j \in V, i < j \tag{3.2}$$

$$x_{ij} + x_{jk} + x_{ki} \leq 2 \qquad \forall i, j, k \in V, i < j, i < k, j \neq k \tag{3.3}$$

$$0 \leq x_{ij} \leq 1 \qquad \forall i, j \in V, i < j \tag{3.4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V, i < j \tag{3.5}$$

The first statement (3.2) means that, in the linear ordering, either vertex i is before j or vice versa, reducing this to a minimal equation system. The inequalities (3.3) are called the 3-dicycle inequalities and along with the first equations guarantee that the graph is free from cycles. The last inequalities (3.4) are called hypercube constraints. The inequalities (3.3) and (3.4) define facets (see Section 5.1 for a definition) of the linear ordering polytope [30]. For $n \leq 5$ the linear ordering polytope is completely described by these inequalities, but for $n \geq 6$ more constraints are needed [55].

For every integer $k \geq 3$ a digraph (V, A) of order $2k$ is called a *simple k-fence* if the vertices can be partitioned in two sets $U = \{u_1, ..., u_k\}$ and $W = \{w_1, ..., w_k\}$ such that

$$A = \bigcup_{i=1}^{k} \Big(\{[u_i w_i]\} \cup \{[w_i v] : v \in U \backslash \{u_i\}\} \Big)$$

For $n \geq 6$ and the arc set $A \subseteq A_n$ of a simple k-fence, $k \geq 3$, the *simple k-fence inequality* is

$$\sum_{[ij] \in A} x_{ij} \leq k^2 - k + 1$$

Let M be the arc set of a simple *Möbius-ladder* in D_n consisting of $k \geq 3$ dicycles $C_1, ..., C_k$ of length four such that each pair of adjacent dicycles (C_1, C_k) and $(C_i, C_{i+1}) \forall i = 1, ..., k-1$ intersects in exactly one arc, say $[a_i b_i] \forall i = 1, ..., k$ and such arcs form a matching in D_n. The *simple Möbius-ladder inequality* is

$$\sum_{[ij] \in M} x_{ij} \leq 3k - \frac{k+1}{2}$$

The simple k-fence inequalities and the Möbius-ladder inequalities define facets of the linear ordering polytope on n vertices [30].

3.8 Edge Completion Problems

An *edge completion problem* consists in, given a graph $G = (V, E)$, finding a *supergraph* $H = (V, E \cup F)$ with the same set of vertices V and an extra set of edges F (called the *fill edges*) that are added to the previously existing ones E, chosen in a way such as H belongs to some predefined class of graphs $\mathscr{C}$, like chordal graphs, interval graphs, split graphs, while optimizing some cost function, like the number of added edges $|F|$, or the clique number of the graph $\omega(H)$. Note that we consider $E \cap F = \emptyset$ for distinguishing the fill edges in F from the original ones in E.

3.8.1 Different Classes for H

Several edge completion problems have been studied in literature, concerning different aimed classes of graphs $\mathscr{C}$ and different cost functions to optimize.

The class of chordal graphs is the most popular class of edge completions. If the desired supergraph H of G is required to be chordal, H is called a *triangulation* of G.

Another class for edge completion problems can be the class of interval graphs. If the supergraph H is required to be an interval graph, the edge completion problem is called an *interval graph completion*.

Edge completion problems where $\mathscr{C}$ is the class of AT-free graphs [39], split graphs [32], proper interval graphs [35] and comparability graphs [33] have also been studied.

3.8.2 Minimum vs. Minimal

These problems can also vary regarding the cost function that is selected to optimize. For example, if the cost function is one less than the size of the largest clique $\omega(H)-1$, its optimization can lead to problems like treewidth or pathwidth.

There exists a triangulation $H = (V, E \cup F)$ of G with maximum clique sizes $k+1$ if and only if the treewidth of G is k [9]. By Lemma 3.6.17, the *treewidth* of a graph G coincides with $\min\limits_{H} \omega(H) - 1$ for all triangulations H of G.

Theorem 3.8.1. *The treewidth is the problem of finding a triangulation H of G that minimizes the size of the largest clique $\omega(H) - 1$. The pathwidth problem consists of finding an interval graph completion that minimizes also $\omega(H) - 1$.*

Definition 3.8.2. A *minimum* $\mathscr{C}$-completion of $G = (V, E)$ is a supergraph $H = (V, E \cup F) \in \mathscr{C}$ that minimizes the number of added edges $|F|$.

A triangulation of G that minimizes the number of added edges $|F|$ is called a *minimum triangulation* or *minimum fill-in*, and an interval graph completion that minimizes the number of added edges $|F|$ is called a *minimum interval graph completion* (IGC).

The minimum fill-in is a NP-hard problem, as well as the minimum interval completion, the treewidth and the pathwidth problems. Most researchers choose to address an easier problem that is related to these, which is to find a *minimal fill-in* or a *minimal interval graph completion* [31].

Definition 3.8.3. A minimal $\mathscr{C}$-completion of $G = (V, E)$ is a supergraph $H = (V, E \cup F) \in \mathscr{C}$ such that every $H' = (V, E \cup F')$ for $F' \subset F$ is not a $\mathscr{C}$-completion of G.

In the case of minimal triangulations, it is equivalent to saying that the removal of a single fill edge of a solution H will result in loosing chordality. For the problem of finding a minimal interval graph completion that does not hold, i.e. removing a single fill edge of a minimal interval graph completion H of G might give a subgraph that it is not interval, but removing more than a single fill edge might give an interval graph completion of G.

A solution to the minimum completion problem must always be a minimal completion, but minimal triangulations or interval completions do not imply that the number of edges is minimum.

3.8.3 Special Classes of Input Graphs

These problems have been studied for special classes of input graphs like trees, caterpillars, two-layer stars, complete bipartite graphs, AT-free graphs, split graphs [43, 47, 52] and for graphs obtained from graph operations [40, 58]. The parameterized versions of these problems have been addressed considering bounded clique size, bounded number of fill edges, and fixed number of colors [35].

These problems can also be posed in two different ways: a *decisional* way that is, given a graph G, to decide whether or not there exists any set of edges F such that H belongs to the class of graphs $\mathscr{C}$ that we want, and the cost is a parameter that does not exceed some given integer value k, or the *optimization version* of the problem, that is to find the actual set of edges F that makes $H \in \mathscr{C}$, while optimizing the cost. The decisional version of the minimum interval graph completion problem has been proved to be NP-complete in [17, 43]. In this thesis we will focus on the optimization version of these problems.

3.8.4 Equivalent Problems in Graph Theory

The minimum interval graph completion problem is known to be equivalent to two other layout problems: the *profile minimization problem* and the *minimum sum cut problem*.

It has been proved by Billionnet [8] that the profile minimization problem is equivalent to the interval graph completion problem, using the consecutive 1's characterization of interval graphs.

The problem of minimizing the profile of a graph is equivalent to finding a linear ordering that minimizes the envelope of the adjacency matrix of the graph with the vertices sorted by that linear ordering. As a result of Theorem 3.4.7, an adjacency matrix with a full envelope corresponds to an interval graph; hence the problem is equivalent to finding the minimum number of edges to add to the graph to obtain an interval graph, which is the minimum interval graph completion problem.

And because the profile minimization problem is equivalent to the sum cut problem, the minimum interval graph completion problem is equivalent to the sum cut problem as well.

3.9 Applications of Graph Layout Problems

Graph Layout problems like bandwidth, pathwidth, cutwidth, profile and minimum interval graph completion have several applications.

The minimum interval graph completion problem has applications in Biology as in the problem of DNA physical mapping [29], and in Archaeology as in the sequence dating or seriation problem [36].

In Numerical Analysis, both profile and bandwidth have applications as a computer storage problem for sparse matrix computations in solving systems of linear equations [40].

There is a linear layout problem that appears in the context of VLSI circuits design, called the Gate Matrix Layout Problem (GMLP), that is equivalent to several problems such as the MOSP, vertex separation, modified cutwidth and pathwidth [44].

We can also apply the minimum interval graph completion problem to a situation in the cutting stock industry, related to the minimization of open stacks problem (MOSP) as we will see in section 6.2.

3.9.1 DNA Physical Mapping

In Molecular Biology, the study of DNA requires that a chromosome must be cut in small fragments called clones and later reconstituted the original sequence. But in the fragmentation process the information about the order of the segments is lost. The information that exists is about the overlapping of pairs of segments and it is based on experiments. In DNA physical mapping, the problem of sequence reconstruction consists in reconstructing the relative position of the fragments along the DNA chain, based on the information of intersection of pairs of clones and assuming that the DNA chain assigns a linear order to the segments of the chromosome, like Seymour Benzer conjectured in 1959 and has later been proved correct.

To determine if two clones intersect, some experiments can be made with the fragments and a set of characteristics is determined for each - this is called the fingerprinting of the clone. If the fingerprints of two clones are sufficiently similar, they are considered to intersect each other. Ideally, this overlap information is measured in probability, but the simplified version admits only three cases:

1. the two clones must intersect (intersection probability=1)

2. the two clones must not intersect (intersection probability=0)

3. it is not known if the two clones intersect or not (intersection probability=0,5)

By considering that each vertex is a clone and there is a given edge in case 1, a forbidden edge in case 2, and a possible fill edge in case 3, this problem corresponds to the Interval Graph Sandwich Problem [34].

Definition 3.9.1. Given a vertex set V and two disjoint sets of edges E^* and E^f over V, a graph $G(V, E)$ with $E^* \subset E \subset \overline{E^f}$ is called a *sandwich graph* for (V, E^*, E^f), where E^f is the set of edges which are disallowed in G. The *Interval Graph Sandwich Problem* (IGS) consists in finding a sandwich graph for (V, E^*, E^f) that is an interval graph.

If all fragments have equal length, like experiments with phage or cosmid clones, the problem reduces itself to the problem of unit interval sandwich that is not equivalent.

In the case that the clones are grouped in sets where each set of clones is obtained by complete digestion of the genome using one or more restriction enzymes, then all clones in a set must be disjoint so that the corresponding vertices of the graph can be assigned different colors. The set of forbidden edges is given implicitly by the coloring of the vertices. This problem is equivalent to the colored interval graph sandwich problem [29].

3.9.2 Seriation Problem in Archaeology

The sequence dating or seriation problem has been formulated by Flinders Petrie in 1899 to face a situation of 900 pre-dynastic Egyptian graves found with 800 varieties of pottery [37, 38]. The problem consists in assigning a date to each grave, and to the objects found in it.

Each variety of pottery is an artifact type or a stylish feature in objects buried throughout the cemetery that can be ascribed to a defined segment of time - the period that those objects were 'in fashion' - which is unknown. The graves may not be actual graves, but may also be any other form of archaeological deposit associated with a single point in time, also unknown.

Under the premises that two graves that lie together in true temporal order of events are more likely to have similar contents, we can date the graves and consequently determine the intervals of time corresponding to the varieties of objects contained there. Normally, a precise date of the object is not obtained, but just a sequence of the intervals of time of each variety in a chronological series, therefore the name “sequence dating”or “seriation”.

An instance of this problem with g graves and v varieties of pottery can be put in an incidence matrix

$$A = [a_{ij}]_{\substack{i=1,..,g \\ j=1,...,v}}$$

of graves vs. varieties defined by

$$a_{ij} = \begin{cases} 1 & \text{if grave } i \text{ contains pottery of variety } j \\ 0 & \text{otherwise} \end{cases}$$

The problem is solved by reordering the rows of this matrix in order to put, simultaneously in each column, all the 1's together. The solution matrix gives the original chronology of the graves and assigns a range of dates to each object.

There is a solution for the problem if the incidence matrix has the consecutive-ones property [24].

As a tribute to Flinders Petrie, an incidence matrix which displays consecutive ones in every column, without needing any rearrangement, is called a *Petrie Matrix.*

3.9.3 Sparse Matrix Computations

Interval graphs are also used in Numerical Analysis for minimizing the computer storage needed in solving systems of linear equations $AX = B$ where

$$\begin{array}{cc|cccccc} & & \multicolumn{6}{c}{Varieties} \\ & & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline & A & 1 & 1 & 0 & 1 & 0 & 0 \\ \text{Graves} & B & 0 & 0 & 0 & 1 & 1 & 1 \\ & C & 0 & 1 & 1 & 1 & 0 & 1 \\ & D & 1 & 1 & 0 & 0 & 0 & 0 \end{array} \rightsquigarrow \begin{array}{cc|cccccc} & & \multicolumn{6}{c}{Varieties} \\ & & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline & D & 1 & 1 & 0 & 0 & 0 & 0 \\ \text{Graves} & A & 1 & 1 & 0 & 1 & 0 & 0 \\ & C & 0 & 1 & 1 & 1 & 0 & 1 \\ & B & 0 & 0 & 0 & 1 & 1 & 1 \end{array}$$

Figure 3.9: Incidence matrix of Graves vs. Varieties before and after rearrangement

A is a sparse matrix. The usual procedures for solving systems of linear equations repeatedly perform computations on the system matrix A. If A is sparse, most of this computations are unnecessarily performed with zeros, and can be avoided by keeping track only of the nonzero entries of A.

To save space storing the nonzero entries, it is advisable to record only the row to which that entry belongs and the distance from the diagonal in the row.

We say that a matrix is *banded* if all non-zero entries are grouped close to the main diagonal of the matrix. If the matrix does not have small band, we may permute rows and columns to translate the original system $AX = B$ into an equivalent system $A'X' = B'$ where A' is a symmetric matrix with smaller band than A.

The profile of the matrix represents the total amount of storage needed. The bandwidth of the matrix represents the maximum length of a column stored, which is a crucial factor, because the execution time of solving the system of equations is proportional to the sum of squares of the column heights.

3.9.4 GMLP: Gate Matrix Layout Problem

Many linear layout problems were motivated by Very Large Scale Integration (VLSI) circuits design.

The general VLSI layout problem consists in, given a set of modules, placing them on a board in a non-overlapping manner (placement problem) and wiring together the terminals on different modules according to a given wiring specification and in such a way that the wires do not interfere among them (routing problem) [18].

This type of circuit consists in a set of vertical wires called *gates*, with

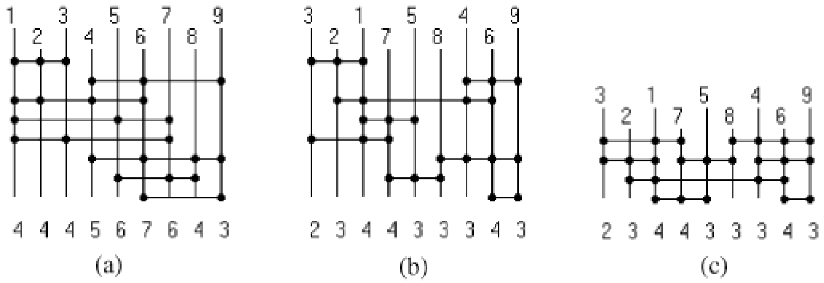


Figure 3.10: The Gate Matrix Layout Problem in a VLSI circuit [44]

transistors (the dots in Figure 3.10), and horizontal wires called *nets* that connect all gates that have transistors in the same position. Altering the sequence of the gates (as in Figure 3.10 (b)) does not change the logic equation of the circuit, so it is possible to rearrange the gates in order to compact the circuit in less area. This can be done if more than one net is accommodated in the same horizontal physical row, called a *track*, as it is shown in Figure 3.10 (c).

The Gate Matrix Layout Problem (GMLP) is a VLSI layout problem in which the objective is to find an optimal sequence of the gates in order to minimize the number of tracks to implement the circuit.

A VLSI circuit can be modeled with a graph in which the vertices represent modules and the edges represent the wires. The square of the cutwidth of the graph gives a lower bound for the area needed for the corresponding VLSI layout.

There are also other relations between the GMLP and other layout measures of the corresponding graph [18]:

$$\min VS(G) = \min PW(G) = \min GMLP(G) + 1$$

In GMLP, the horizontal metal wire of a net starts at the first gate on the left and passes through all “gates”of the circuit to the last gate on the right of the net, occupying physical space on the circuit board. In MOSP, a “stack”is open at the moment the first item is cut and stays open until the last item equal to that is cut, occupying physical space during that period of time. The number of tracks needed for GMLP is the same as the number of stacks simultaneously open in the MOSP, and these two problems are equivalent [44].

3.10 Conclusions

The graph theory that combines perfect graphs and linear layout measures is very rich. There are dozens of papers produced in the last fifty years about this subject, studying it from its structural form, or from its computational point of view. This theory has been applied to several different subjects as Molecular Biology, Archeology, Numerical Analysis, or Circuits Design.

Combining the characterization of interval graphs given by Olariu, with the theory behind the linear layout of a graph, and the interval graph completion problem, it is possible to develop a mathematical programming model that will be presented in the following chapters, which has applications in the industrial problems described in Chapter 2.

Chapter 4

An Integer Programming Formulation for the MOSP

In this chapter we present an integer programming formulation for the MOSP based on interval graphs and the existence of a perfect vertex elimination scheme. We first associate the MOSP problem with a graph with a vertex for each item stack and with an arc between two vertices if there is a pattern that produces both items. We solve the MOSP by converting this graph into an appropriate interval graph and defining an ordering of the vertices based on a sequence of cliques.

4.1 Introduction

Consider a cutting machine that processes just one cutting pattern at a time. The items already cut that are equal are piled in stacks by the machine. The stack of an item type remains near the machine if there are more items of that type to be cut in a forthcoming pattern. A stack is closed and removed from the work area only after all items of that size have been cut, and immediately before starting to process the next cutting pattern. After a pattern is completely cut and before any stack is removed the number of open stacks is counted. The maximum number of open stacks for that sequence of patterns is called the *MOSP number*.

There are often space limitations around the cutting machines, there is danger of damages on the stacked items, difficulty in distinguishing similar items, and in some cases there are handling costs of removing the stack temporarily to the warehouse. It is advantageous to minimize the number of open stacks, and that can be done simply by finding an optimal sequence to process the cutting patterns.

MOSP has been proved to be a NP-hard problem [44].

This problem has applications in several cutting industries like steel tubes, paper, flat glass, wooden panels, as well as in production planning (for rapidly fulfilling the customers' orders as we explain in Section 2.1.2), and also in other fields such as VLSI Circuit Design with the Gate Matrix Layout Problem and PLA Folding, as referred to in Section 3.9.4, and in classical problems from Graph Theory presented in Section 3.6 such as Path-width, Modified Cutwidth and Vertex Separation. All of these problems are explained in the previous chapters of this work.

4.2 The MOSP in a Graph

As suggested in [61], an instance of the MOSP can be associated with a graph having a vertex for each item that is cut and an edge between two vertices if the corresponding items are present in the same cutting pattern.

A pattern with k different items will correspond to a clique of size k in the MOSP graph, because the k vertices must be connected to each other.

As an example, we will see an instance of the MOSP with seven patterns and six different items, presented in Table 4.1.

Patterns:	P_1	P_2	P_3	P_4	P_5	P_6	P_7
Item 1	X			X		X	
Item 2		X	X				
Item 3	X						
Item 4		X		X	X		X
Item 5	X		X				
Item 6					X		

Table 4.1: An instance of the MOSP with 7 cutting patterns and 6 items

This instance originates a MOSP graph with 6 vertices, one corresponding to each item, and with edges between the vertices (items) that belong to the same pattern.

Notice that, for example, 2 is an item produced only by the cutting patterns P2 and P3, so vertices 2 and 4 are connected because pattern P2 produces both items 2 and 4, and 2 and 5 are connected because those items are both contained in pattern P3.

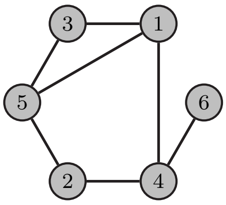


Figure 4.1: MOSP Graph of the instance in Table 4.1

To optimize the number of stacks, it is convenient to find the best sequence to process the cutting patterns. Considering that the patterns do not appear explicitly in the MOSP graph constructed in this way, how will we find that sequence for the cutting patterns? We will focus on finding a sequence to open the stacks, rather than on sequencing the cutting patterns, as seen on section 2.3.1. That is not a problem, because it is possible to take a solution for the ordering of the vertices of the graph and construct a sequence for the corresponding cutting patterns [64]. A linear ordering of the vertices sets an ordering for the opening of the stacks; following this ordering, a pattern will be put in the sequence when it is the first time that all vertices corresponding to all items present in that pattern have been opened.

When there are some patterns with only one item that is also produced by another pattern, we say that the first pattern is contained in the second pattern. It has been proved by Yanasse [63] that this type of patterns can be removed from the problem and inserted later in the solution just before the patterns in which they were contained.

There are some other situations that, due to their simplicity, can be removed from the original problem while solving it and inserted later in the solution. Items that are present in just one pattern will appear in the graph as isolated vertices if that pattern does not include any other item. In this case, that pattern can be the first or last in the sequence, and it will open and close a stack without any other stacks open at that same time, so it does not increase the maximum number of simultaneously open stacks.

If a pattern produces only one item type, but that item is also present in other cutting patterns (like patterns P6 or P7 in Table 4.1), then that pattern should be sequenced just before the first of the patterns containing that item, and the number of simultaneously open stacks will not increase [63].

In this example, pattern P6 is contained in patterns P1 and P4, and pattern P7 is contained in patterns P2, P4 and P5. This instance can be reduced to only five relevant patterns (P1, P2, P3, P4 and P5) generating the same graph. Pattern P6 can be sequenced in the solution just before the first of the patterns P1 and P4 in which it is contained, and pattern P7 just before the first of the patterns P2, P4 and P5, without increasing the MOSP number. If there would exist a pattern that was not contained in another pattern, then it should be the first or the last in the sequence, it would open and close a stack without any other stacks open at that same time, so it would not increase the MOSP number.

It is sufficient to solve MOSP instances with at most two items per pattern [61]. If there are more then two items per pattern, we can subdivide it in subpatterns with two items, one corresponding to each arc. The solution of this new problem with only two items per pattern can be transformed in a solution of the original problem in polynomial time [61].

A feasible solution of this problem corresponds to a sequence of arcs in the graph. Using that sequence of arcs, a stack j is open when it is the first time that an arc with an end in j is traversed and the stack is closed when all arcs with an end in j have been traversed. Going along the sequence of arcs, and the corresponding order of the opening of the stacks, we sequence a pattern P_i of the original problem as soon as all nodes corresponding to all items in P_i have been opened.

For the example in Table 4.1, if the patterns are sequenced by their natural ordering $P_1P_2P_3P_4P_5P_6P_7$, there is a period of time when there are four open stacks simultaneously. If the ordering of the patterns is changed, the number of open stacks can be lowered.

A possible solution is the sequence of vertices 2-5-4-6-1-3 that corresponds to the stacks opening and also to a sequence of patterns $P_3P_7P_2P_5P_6P_4P_1$.

As there are some stacks that are not simultaneous at any time, like 3 and 4, or 1, 2 and 6, those stacks can use the same stack space; hence this sequence of patterns gives a maximum of three simultaneously open stacks, that is the optimum for this instance, as can be observed in Figure 4.2.

This means that it is natural to associate the lifetime of a stack in the solution with intervals of time measured not in minutes or hours but measured in terms of the patterns in the sequence.

We have seen that we can start solving a MOSP problem with a graph, and that in the solution of the problem we can consider an interval for the time that each stack is open.

By associating each open stack of our MOSP problem to an interval in the real line (the interval of time that the stack stays open), we can associate a solution of the MOSP to an interval representation of an interval graph.

An interval graph can be associated to the set of intervals in the solution

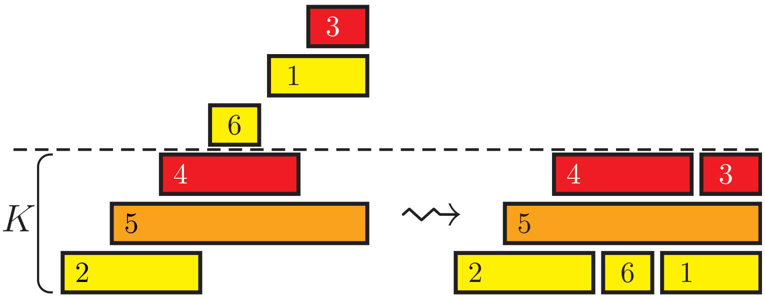


Figure 4.2: Non simultaneous items can share stack space

and the MOSP graph will be modified in order to become an interval graph. We will use some properties of interval graphs to find the solution of MOSP instances.

For the example mentioned before in Figure 4.1, the interval graph corresponding to the solution displayed in Figure 4.2 has the same vertices and edges of the MOSP graph and two additional edges, as depicted in Figure 4.3. This is an interval graph completion (as explained in Section 3.8.1) of the original MOSP graph. The fill edge [54] was added to make the graph chordal, because it is a chord of the previous 4-cycle $1, 4, 2, 5$. The fill edge [56] was added to eliminate the asteroidal triple $3, 2, 6$, transforming the MOSP graph in an AT-free graph. In the original MOSP graph $3, 2, 6$ is an AT because $3, 5, 2$ is a path from vertex 3 to vertex 2 that does not pass through any neighbor of vertex 6. With the edge [56] now vertex 5 is a neighbor of vertex 6.

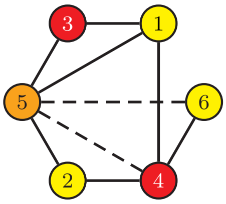


Figure 4.3: Interval Graph of the instance in Table 4.1

As discussed in Section 3.4, the vertex order defined by the left endpoints of the intervals is related to the sequence of cliques that will appear in the interval graph of the solution of a MOSP problem.

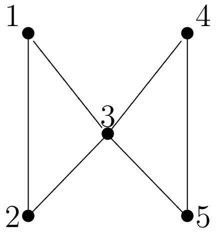
We will use in our model inequalities derived from the characterization in Theorem 3.4.9 to guarantee that the graph obtained in the solution of the problem is an interval graph.

4.3 A Preview of the Model

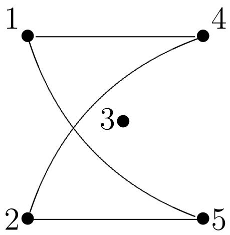
For an instance of the problem, we first build a graph $G = (V, E)$, associating each item cut from the patterns to a vertex and creating an arc joining vertex i and j if and only if items i and j are cut from the same pattern. This graph may not be an interval graph at the start, but we will add some arcs to it in such a way that it will become one. We need this graph to become an interval graph because, if we associate each item to the interval of time in which the stack of that item is open, we can use the graph to model what intervals should occur simultaneously and what intervals should precede others. According to the sequence in which the cutting patterns are processed, there may be more or less open stacks simultaneously. Each arc of the future interval graph means that, for a period of time, the two stacks (the respective vertices of the arc) will remain both open. The initial graph contains only the arcs that must be there, in any possible sequence in which the patterns can be processed. The rest of the arcs that are added later to the graph will differ according to the sequence of the patterns. It is the choice of these arcs that defines which are the other simultaneously open stacks. Our model consists in finding out which edges should be added to the original MOSP graph $G = (V, E)$ in order to get an interval graph $H = (V, E \cup F)$ that minimizes the maximum number of simultaneously open stacks.

The model is based on the assumption that it is always possible to add more arcs to G without losing the optimum value of the associated MOSP problem.

Consider the following graph G with five vertices and six arcs.

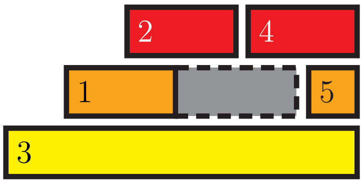


The complement graph $\bar{G}$ has the same vertices of G and the arcs that G does not have:

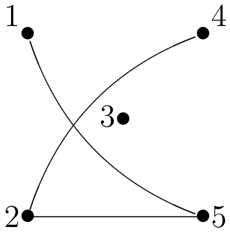


In $\bar{G}$ there are 4 arcs to orientate. If the conjecture is true, than one can add more arcs to the graph G, without losing the optimum value, and thus having less arcs in $\bar{G}$ to give an orientation. We want to find the minimum number of arcs that we need to add in G in order to have less arcs in $\bar{G}$ without altering the optimum value.

In the corresponding MOSP problem, intervals 1 and 4 do not overlap because there is no arc between vertex 1 and vertex 4. In this case the optimum is 3.



One can postpone the closing of interval 1 until interval 5 begins, without modifying the value of the optimum, but having to add the arc between vertices 1 and 4. Adding this type of arcs does not change the optimum and this results in less arcs in $\bar{G}$ to orientate.



4.4 The Decision Variables

We set an ordering for opening the stacks by assigning a number to each item cut, with a bijective function $\varphi : V \to \{1, ..., N\}$. This linear ordering of the vertices is set by the decision variables x_{ij}:

$$x_{ij} = \begin{cases} 1 & \text{if } \varphi(i) < \varphi(j) \\ 0 & \text{otherwise} \end{cases} \qquad \forall i, j \in V$$

Notice that $x_{ii} = 0$ for any $i \in V$ and also that we have

$$x_{ij} = 1 \Leftrightarrow x_{ji} = 0$$

These variables are setting an orientation into the arcs, for us to keep track of the sequence of the items in the current instance. If $x_{ij} = 1$ then item i starts being cut before the item j is, even though the corresponding stacks may overlap or not, i.e., in spite of having an arc between the two vertices or not.

The other decision variables that will be used are concerned to the arcs that are necessary to add to the original graph $G = (V, E)$ to get an interval graph $H = (V, E \cup F)$ and, together with variables x, determine which intervals will overlap in the desired interval graph. To decide which of these additional arcs are to be added, we define a variable y_{ij} for each arc ij that did not exist before in the graph:

$$y_{ij} = \begin{cases} 1 & \text{if } [ij] \notin F \text{ and } \varphi(i) < \varphi(j) \\ 0 & \text{if } [ij] \in F \text{ or } \varphi(i) \geq \varphi(j) \end{cases} \qquad \forall i, j \in V : [ij] \notin E$$

Notice that y_{ij} is 1 when the arc $[ij]$ is NOT added, because the variable y_{ij} works like an "eraser"variable. To get an interval graph, if we decided to add to the original graph all the arcs that were missing, and then remove some of them - the ones that we do not need to have an interval graph, then variable y is 1 for these additional arcs which are to be removed.

Variables y depend on the linear ordering of vertices, so it follows that there is an anti-reflexive relation:

$$y_{ij} = 1 \Rightarrow y_{ji} = 0$$

When $y_{ij} = 1$, the arc $[ij]$ is not needed in the interval graph, so, by definition of interval graph, if there is not an arc $[ij]$, then the intervals i and j do not intersect. Consequently, one of the intervals should finish before the other one starts. As $i \prec j$, the interval i opens and finishes before the interval j starts. It means that the stacks for items i and j will never be open at the same time, so they can share the same stack space, as seen in Figure 4.4.

We will use the variable $K \in \mathbb{N}$ to denote the maximum number of simultaneously open stacks.

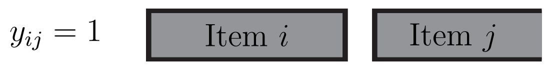


Figure 4.4: Interval i opens and closes before j starts

4.5 The Inequalities of the Model

Now we study the relations between the binary integer variables x and y and the integer variable K, to build the inequalities for our model.

4.5.1 Linear Ordering of the Vertices

The linear ordering of the vertices brings two basic constraints [55, 22] that were already presented in Section 3.7. The first one is an equation that states that either vertex i precedes vertex j or vice-versa. This is expressed by:

$$x_{ji} + x_{ij} = 1 \quad \forall i, j \in V : i \neq j \tag{4.1}$$

The second one prevents directed 3-cycles, by stating that if vertex i precedes vertex j and vertex j precedes vertex k, than vertex i should precede vertex k. This transitivity property of the linear ordering can be expressed by:

$$x_{ij} + x_{jk} + x_{ki} \leq 2 \quad \forall i, j, k \in V : i \neq j \neq k \tag{4.2}$$

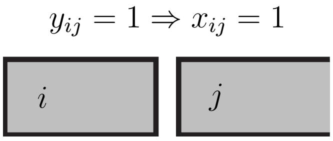
4.5.2 Precedences of the Opening and Closing of the Intervals

An important remark upon the variables y_{ij} is that they establish the precedences between the closing and opening of the intervals.

As one of the conditions for the variable y_{ij} to be equal to 1 is that vertex i precedes vertex j, equal to say that $x_{ij} = 1$, then we must have:

$$y_{ij} \leq x_{ij} \quad \forall i, j \in V : i \neq j, [ij] \notin E \tag{4.3}$$

When $y_{ij} = 1$, the arc $[ij]$ is not needed in the interval graph; so, by definition of interval graph, if there is not an arc $[ij]$ then intervals i and j do not intersect. Consequently, one of the intervals should finish before the other one starts. As $y_{ij} \leq x_{ij}$, we must also have $x_{ij} = 1$, determining that the interval i opens and finishes before the interval j starts.



4.5.3 Obtaining an Interval Graph

To guarantee that the graph $H = (V, E \cup F)$ is an interval graph, we use in the model the characterization given in Theorem 3.4.9, to express relations between the binary variables y_{ij} and x_{ij}. Recall that the referred Theorem characterizes interval graphs as graphs in which the vertices can be linearly ordered in such a way that, for any three vertices i, j, k such that $i \prec j \prec k$, if $[ik] \in E$ then $[ij] \in E$.

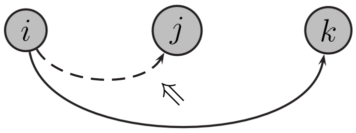


Figure 4.5: Olariu's characterization of interval graphs

We will consider three different vertices $i, j, k \in V$ that do not form a clique in the original MOSP graph, and analyze in what circumstances the arcs $[ik]$ and $[ij]$ exist or have to be added. Let us separate the analysis in two cases:

- arc $[ik] \in E$
- arc $[ik] \notin E$

Case 1: arc $[ik] \in E$

Let us suppose that arc $[ij] \notin E$, otherwise H already obeys the condition needed in an interval graph. If $i \prec j \prec k$ then as $[ik] \in E$ then for H to be an interval graph it must be $[ij] \in F$.

When $x_{ij} = 1$, an arc $[ij]$ will belong to the graph H if $y_{ij} = 0$. Clearly, if $x_{ij} = 1$, then $y_{ji} = 0$. Olariu's characterization can be expressed as follows. For each arc $[ij] \notin E$, the value of the corresponding variable y_{ij} should obey:

$$y_{ij} + x_{ij} + x_{jk} \leq 2.$$

The inequality states that, if vertex i precedes j and vertex j precedes k, or equivalently $x_{ij} = x_{jk} = 1$, then the variable y_{ij} must be equal to 0, i.e. arc $[ij] \in F$, as in Figure 4.5.

This inequality can be strengthened as follows. Combine the inequality with the inequalities $y_{ij} \leq x_{ij}$ and $x_{jk} \leq 1$ to obtain:

$$\begin{array}{rcl} y_{ij} + x_{ij} + x_{jk} & \leq & 2 \\ y_{ij} - x_{ij} & \leq & 0 \\ x_{jk} & \leq & 1 \\ \hline 2y_{ij} + 2x_{jk} & \leq & 3 \end{array}$$

Divide both sides by 2 and, as the variables are integers, we can round the fractional part of the right hand side to obtain a stronger inequality:

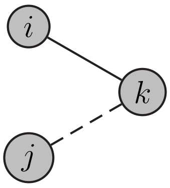
$$y_{ij} + x_{jk} \quad \leq \quad 1$$

Because $x_{kj} = 1 - x_{jk}$, this inequality is equivalent to the constraint in binary variables equivalent to the logical implication $y_{ij} \Rightarrow x_{kj}$.

$$y_{ij} \leq x_{kj} \quad \forall i, j, k \in V, [ij] \notin E, [ik] \in E \tag{4.4}$$

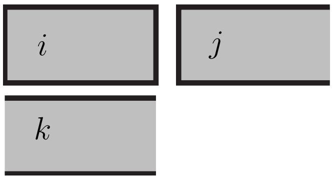
We have assumed that $i \prec j \prec k$ but this is valid too if $i \prec j$ but $k \prec j$, because we have $x_{kj} = 1$. If $j \prec i$ then $x_{ij} = 0$ and by (4.3) we have $y_{ij} = 0$ and the inequality is also valid.

In fact, when there is the arc $[ik]$ in the initial graph, but not the arc $[ij]$ (it is indifferent if the arc $[kj]$ exists or not), this means that intervals i and k must overlap.



If $y_{ij} = 1$, then interval i will close before interval j starts. As interval k must overlap interval i, because $[ik] \in E$, k must be already open when j starts. So we must have $x_{kj} = 1$, as depicted in the next figure.

$$y_{ij} = 1 \Rightarrow x_{kj} = 1$$



In the example previously presented in this chapter, the vertices 2, 6 and 4 form a set in these conditions, because $[24] \in E$ but $[2,6] \notin E$. Hence, in the model for this example there is the inequality:

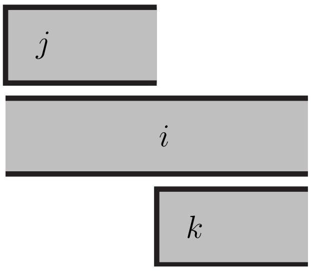
$$y_{26} \leq x_{46}$$

In the solution, as can be observed in Figure 4.2, interval 2 opens and closes before interval 6 opens ($y_{26} = 1$) and the linear ordering is $2 \prec 4 \prec 6$.

Note that if both arcs $[ik]$ and $[jk] \in E$ and $x_{ik} = x_{jk} = 1$, then both i and j are predecessors of k. Following the definition of an interval graph, the predecessors of k must form a clique. In the model, that is equivalent to having $y_{ij} = y_{ji} = 0$, meaning that there should be an arc between vertices i and j.

$x_{jk} = 1$ means that interval j opens before interval k. As $[ik] \in E$, interval k must overlap with interval i, even if i opens before j starts. Then interval i cannot be closed before j starts because i has to wait till k starts. The situation is captured in the following picture.

$$x_{ik} = x_{jk} = 1 \Rightarrow y_{ij} = y_{ji} = 0$$



In the example being analyzed, the set of vertices 5, 4 and 1 will admit in the model the inequality:

$$y_{54} \leq x_{14}$$

As in the solution the linear ordering of these vertices is $5 \prec 4 \prec 1$, this inequality forces $y_{54} = 0$ meaning that the arc [54] is added to the graph, as can be seen in Figure 4.3.

This could be modeled as the following constraint:

$$y_{ij} + y_{ji} + x_{ik} + x_{jk} \leq 2$$

or equivalently

$$y_{ij} + y_{ji} \leq (1 - x_{ik}) + (1 - x_{jk})$$

However, this constraint is a combination of the constraints $y_{ij} \leq x_{kj}$ and $y_{ji} \leq x_{ki}$, and therefore it is weaker than considering both constraints separately.

Case 2: arc $[ik] \notin E$

On the other hand, the arc $[ik]$ may not be originally in the set of arcs E, but may be added to it to make the graph chordal, as a result of other constraints in the model. In this situation, $[ik] \in F$, we will have the variable y_{ik} taking the value 0, and the function $(x_{ik} - y_{ik})$ taking the value 1, meaning that arc $[ik]$ is added to the set.

In this second case, also consider $[ij] \notin E$, because otherwise the result was guaranteed.

Clearly, Olariu's characterization should also apply to this case.

Therefore, for each arc $[ij] \notin E$, the value of the corresponding variable y_{ij} can be constrained as follows:

$$y_{ij} + (x_{ik} - y_{ik}) + x_{jk} \leq 2.$$

The inequality states that, if both vertices i and j precede k, or equivalently $x_{ik} = x_{jk} = 1$, when the variable y_{ik} is set to 0 by another constraint (meaning that the arc $[ik]$ is added to the graph G) then the variable y_{ij} must also be equal to 0 (meaning that $[ij] \in F$ or i does not precede j).

This inequality can be strengthened as follows. Combine the inequality with the following inequalities from the linear ordering polytope, as well as a non-negativity constraint, to obtain:

$$\begin{array}{rcl}
y_{ij} + x_{ik} - y_{ik} + x_{jk} & \leq & 2 \\
x_{ij} - x_{ik} + x_{jk} & \leq & 1 \\
y_{ij} - x_{ij} & \leq & 0 \\
-y_{ik} & \leq & 0 \\
\hline
2y_{ij} + 2x_{jk} - 2y_{ik} & \leq & 3
\end{array}$$

By dividing both sides by 2, and rounding the fractional part of the righthand side, we obtain:

$$y_{ij} + x_{jk} - y_{ik} \quad \leq \quad 1$$

This inequality is equivalent to a constraint in binary variables equivalent to the logical implication $y_{ij} \Rightarrow x_{kj} \vee y_{ik}$:

$$y_{ij} \leq x_{kj} + y_{ik} \qquad \forall i, j, k \in V, [ij], [ik] \notin E \tag{4.5}$$

Supposing that $i \prec j \prec k$, this means that $x_{jk} = 1$ or equivalently $x_{kj} = 0$. If we decide to add the arc $[ik]$, then $y_{ik} = 0$ and the inequality forces $y_{ij} = 0$, meaning that we must also add the arc $[ij]$ for the graph to be an interval graph.

This inequality is also true if the arc $[ik]$ is not added because then $y_{ik} = 1$ and y_{ij} would be free.

This inequality is also valid in all other possible orderings of the vertices i, j, k as can be seen in Table 4.2.

Vertices	y_{ij}	$\leq$	x_{kj}	$+$	y_{ik}
$i \prec j \prec k$	0		0		0
$i \prec k \prec j$	free		1		0
$j \prec i \prec k$	0		0		0
$j \prec k \prec i$	0		0		0
$k \prec i \prec j$	free		1		0
$k \prec j \prec i$	0		1		0

Table 4.2: Possible cases when arc $[ik] \in F$

In the second, fifth and sixth cases ($k \prec j$), the adding of the arc $[ik]$ to the graph does not force to add the arc $[ij]$. In the remaining three cases where $j \prec i$, the inequality $y_{ij} \leq x_{ij}$ (4.3) forces $y_{ij} = 0$.

In fact, if $y_{ij} = 1$, meaning that i ends before j starts, then $x_{kj} = 1$ meaning that k should start before j, as shown in Figures 4.6 and 4.7, or $y_{ik} = 1$, meaning that i should end before k starts, as depicted in Figures 4.7 and 4.8.

In the example, the three vertices 6, 1 and 3 originate the inequality

$$y_{61} \leq x_{31} + y_{63}$$

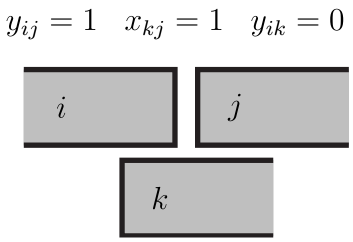


Figure 4.6: Interval i closes before interval j opens, with interval k being simultaneous to interval i and opening before j

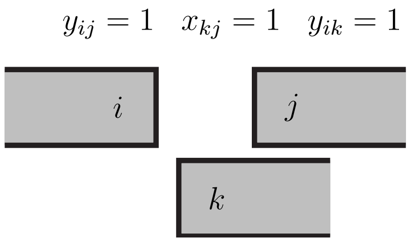


Figure 4.7: Interval i closes before intervals j and k open, with interval k opening before interval j

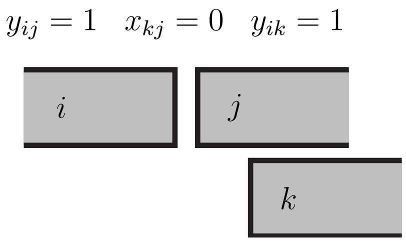
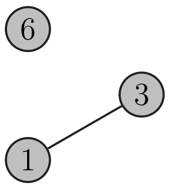


Figure 4.8: Interval i closes before intervals j and k open, with interval j opening before interval k



As in the solution the linear ordering of these vertices is $6 \prec 1 \prec 3$, if the arc [63] was added, then the arc [61] should also be added. In this case both of the arcs were not added, making these three variables equal to one.

4.6 An objective function to evaluate the MOSP

Consider K to be the maximum number of open stacks. At each node, except at the first K ones, there is an interval that begins and one that ends, so that all the cliques in the sequence of the perfect elimination order are maximal, except the first K ones. And because every appearance of a vertex in these cliques must be consecutive and there are N instants in the sequence, at each instant only one vertex changes. This change corresponds to the closing of an interval and the beginning of another.

The precedences of the opening and closing of the intervals are declared by the variables y_{ij}. If we sum up the variables y_{ij} we can count the precedences between the intervals. For every vertex j, the sum $\sum_{i=1}^{N} y_{ij}$ counts how many intervals finish before interval j starts and the sum $\sum_{j=1}^{N} y_{ij}$ counts how many intervals will start after i finishes. The vertex that finishes first is the one that has the greatest $\sum_{j=1}^{N} y_{ij}$.

If we associate each opening of an interval to an instant of time, this last sum $\sum_{j=1}^{N} y_{ij}$ says how long we must wait since the opening of i until the end of processing all intervals. The instant of closing of the interval i is $N - \sum_{j=1}^{N} y_{ij}$, *i.e.* , the number of intervals that must be started minus the number of intervals that will start after interval i closes. For example, a vertex with $\sum_{j=1}^{N} y_{ij} = 2$ means that there are two vertices that start after i finishes, each at a different instant in time. The last instant of this interval is $N - \sum_{j=1}^{N} y_{ij} = N - 2$.

If we sum up the variables x_{ij} we will find the position of each vertex in the sequence of vertices. For every vertex i, the sum $\sum_{j=1}^{N} x_{ij}$ counts how many vertices come after i in the ordering, i.e., the number of intervals that will start after i starts. Computing for every vertex i the sum $\sum_{j=1}^{N} x_{ij}$ and ordering those sums, we get the sequence in which the orders should open:

the vertex i having the highest sum $\sum_{j=1}^{N} x_{ij}$ corresponds to the first order to open, because vertex i precedes all the other vertices j. The lowest sum equals zero and corresponds to the last vertex to open.

For every vertex j, the sum $\sum_{i=1}^{N} x_{ij}$ counts how many vertices precede j, i.e., the number of intervals that start before i starts. If we want to know which interval opens at instant t we just have to find the vertex j such that $\sum_{i=1}^{N} x_{ij} = t - 1$. The beginning of an interval j happens at instant $\sum_{i=1}^{N} x_{ij} + 1$. It is the number of intervals that have started before j plus the interval j itself. So the number of intervals that are open at that instant is $\sum_{i=1}^{N} x_{ij} + 1 - \sum_{i=1}^{N} y_{ij}$ because we need to subtract the number of intervals that have already been closed before that instant.

This leads to a set of functions that can be used to evaluate the MOSP number:

$$\sum_{\substack{i=1\\ i\neq j}}^{N} x_{ij} - \sum_{\substack{i=1\\ [ij]\notin E}}^{N} y_{ij} \ + 1 \leq K \quad \forall j = 1, ..., N \tag{4.6}$$

Each function provides a lower bound for the MOSP, being K the maximum of those functions. The objective function of the model is to minimize K.

If one puts each interval in a line, as in Figure 4.9, the number of lines that we have open when an interval starts is a lower bound for the maximum number of open stacks.

In the example presented before, when interval 5 starts, the number of open stacks turns to two,

$$j = 5 : \quad \sum x_{ij} - \sum y_{ij} + 1 = 1 - 0 + 1 = 2 \ \leq 3$$

which is a lower bound for the MOSP (which is three). The same inequality, correspondent to the moment that interval 1 starts, gives a better lower bound. At that instant, there are four intervals already open (2, 5, 4 and 6) but two of those have already closed (intervals 2 and 6). Hence the inequality is:

$$j = 1 : \quad \sum x_{ij} - \sum y_{ij} + 1 = 4 - 2 + 1 = 3 \ \leq 3$$

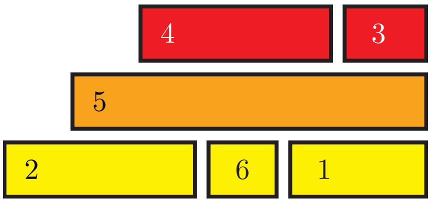


Figure 4.9: Optimal solution of the example from Table 4.1

There are 6 cliques in the sequence of the perfect elimination order, such that every appearance of a vertex in these cliques is consecutive. The sequence of cliques is:

$$\{2\}, \{2, 5\}, \{2, 5, 4\}, \{6, 5, 4\}, \{1, 5, 4\}, \{1, 5, 3\}$$

The first 2 cliques are not maximal, the remaining of them are maximal cliques with size 3, which is the optimum for this instance of the MOSP.

4.7 The integer programming formulation for MOSP

Given an instance of the minimization of open stacks problem with N items, we start by building the correspondent MOSP graph $G = (V; E)$ with $|V| = N$ and $|E| = M$. Combining all that has been said in the previous sections, our basic new mathematical formulation for the MOSP problem is:

Minimize K

Subject to:

$$\begin{aligned}
x_{ij} + x_{ji} &= 1 && \forall i,j = 1,...,N \text{ with } i \neq j && (4.7)\\
x_{ij} + x_{jk} + x_{ki} &\leq 2 && \forall i,j,k = 1,...,N \text{ with } i \neq j \neq k && (4.8)\\
y_{ij} &\leq x_{ij} && \forall i,j = 1,...,N \text{ with } i \neq j \text{ and } [ij] \notin E && (4.9)\\
y_{ij} &\leq x_{kj} && \forall i,j,k = 1,...,N \text{ with } [ij] \notin E,\ [ik] \in E && (4.10)\\
y_{ij} - y_{ik} &\leq x_{kj} && \forall i,j,k = 1,...,N \text{ with } [ij],[ik] \notin E && (4.11)\\
\sum_{\substack{i=1\\ i\neq j}}^{N} x_{ij} - \sum_{\substack{i=1\\ [ij]\notin E}}^{N} y_{ij} \ +1 &\leq K && \forall j = 1,...,N && (4.12)\\
x_{ij} &\in \{0,1\} && \forall i,j = 1,...,N \text{ with } i \neq j && (4.13)\\
y_{ij} &\in \{0,1\} && \forall i,j = 1,...,N \text{ with } i \neq j, [ij] \notin E && (4.14)\\
K &\in \mathbb{N} && && (4.15)
\end{aligned}$$

4.8 Reducing the Number of Variables in the Model

The variables x_{ij} were defined for every $i, j = 1, ..., N$ such that $i \neq j$, but it is possible to use only half of these variables, defining x_{ij} only for $i < j$, because all the other variables are defined by consequence by the equation (4.7). This will make some changes in four of the remaining inequalities, because (4.8) to (4.12) have to be rewritten using only the variables x_{ij} such that $i < j$.

The inequality (4.8) is equivalent to the six following expressions avoiding the removed variables:

$$\begin{aligned} x_{ij} + x_{jk} + x_{ki} &\leq 2 \\ \Leftrightarrow x_{ij} + x_{jk} - x_{ik} &\leq 1 \text{ if } i < j < k \qquad (4.16) \\ \Leftrightarrow x_{ik} + x_{kj} - x_{ij} &\geq 0 \text{ if } i < k < j \qquad (4.17) \\ \Leftrightarrow x_{ki} + x_{ij} - x_{kj} &\leq 1 \text{ if } k < i < j \qquad (4.18) \\ \Leftrightarrow x_{kj} + x_{ji} - x_{ki} &\geq 0 \text{ if } k < j < i \qquad (4.19) \\ \Leftrightarrow x_{ji} + x_{ik} - x_{jk} &\geq 0 \text{ if } j < i < k \qquad (4.20) \\ \Leftrightarrow x_{jk} + x_{ki} - x_{ji} &\leq 1 \text{ if } j < k < i \qquad (4.21) \end{aligned}$$

Clearly inequalities (4.16), (4.18), and (4.21) are the same (just using different index letters) and inequalities (4.17), (4.19), and (4.20) are the same too. Hence inequality (4.8) gives origin to two inequalities when considering only half the variables x_{ij}:

$$x_{ij} + x_{jk} - x_{ik} \geq 0 \ \forall i, j, k \in V \text{ with } i < j < k \qquad (4.22)$$

$$x_{ij} + x_{jk} - x_{ik} \leq 1 \ \forall i, j, k \in V \text{ with } i < j < k \qquad (4.23)$$

The linking inequality (4.9) is kept for $i < j$ but has to be different if $j < i$, so it gives rise to the two following inequalities:

$$y_{ij} \leq x_{ij} \quad \forall i, j \in V \text{ with } i < j, [ij] \notin E \qquad (4.24)$$

$$y_{ij} + x_{ji} \leq 1 \quad \forall i, j \in V \text{ with } j < i, [ij] \notin E \qquad (4.25)$$

The Olariu inequalities in the model will also have to be analyzed because they use the variables x_{ij}. The first one, (4.10) is unfolded in two inequalities as the previous one:

$$y_{ij} \leq x_{kj} \quad \forall i, j, k \in V \text{ with } k < j, [ij] \notin E, [ik] \in E \qquad (4.26)$$

$$y_{ij} \leq 1 - x_{jk} \quad \forall i, j, k \in V \text{ with } j < k, [ij] \notin E, [ik] \in E \qquad (4.27)$$

Now let us analyze the second Olariu inequality (4.11). For every $i, j, k \in V$ such that $[ij], [ik] \notin E$, because previously we had both the variables x_{jk} and x_{kj}, then we must have both the inequalities:

$$y_{ij} - y_{ik} \leq x_{kj} \ \forall i, j, k = 1, ..., N \text{ with } k < j, [ij], [ik] \notin E \qquad (4.28)$$

$$y_{ik} - y_{ij} \leq x_{jk} \ \forall i, j, k = 1, ..., N \text{ with } j < k, [ij], [ik] \notin E \qquad (4.29)$$

Each of these inequalities will be kept or rewritten according to the case $j < k$ or $k < j$. The first one is equivalent to:

$$\begin{aligned} y_{ij} - y_{ik} &\leq x_{kj} \\ \Leftrightarrow y_{ij} - y_{ik} - x_{kj} &\leq 0 \text{ if } k < j \\ \Leftrightarrow y_{ik} - y_{ij} + x_{kj} &\geq 0 \text{ if } k < j \qquad (4.30) \\ \Leftrightarrow y_{ij} - y_{ik} &\leq 1 - x_{jk} \text{ if } j < k \\ \Leftrightarrow y_{ij} - y_{ik} + x_{jk} &\leq 1 \text{ if } j < k \qquad (4.31) \end{aligned}$$

And the second one is

$$\begin{aligned} y_{ik} - y_{ij} &\leq x_{jk} \\ \Leftrightarrow y_{ik} - y_{ij} &\leq 1 - x_{kj} \text{ if } k < j \\ \Leftrightarrow y_{ik} - y_{ij} + x_{kj} &\leq 1 \text{ if } k < j \qquad (4.32) \\ \Leftrightarrow y_{ik} - y_{ij} - x_{jk} &\leq 0 \text{ if } j < k \\ \Leftrightarrow y_{ij} - y_{ik} + x_{jk} &\geq 0 \text{ if } j < k \qquad (4.33) \end{aligned}$$

There is no need for all of these, because some of them are equivalent; we can use only (4.30) and (4.32) if the indexes i, j, k chosen are such that $k < j$, or use (4.31) and (4.33) if $j < k$.

All the other inequalities in the model use only the variables y_{ij} so they can be kept. In summary, the model with reduced number of variables is

Minimize K

Subject to:

$$0 \leq x_{ij} + x_{jk} - x_{ik} \leq 1 \qquad \forall i, j, k = 1, ..., N, i < j < k \qquad (4.34)$$

$$y_{ij} - x_{ij} \leq 0 \qquad \forall i, j = 1, ..., N, i < j, [ij] \notin E \qquad (4.35)$$

$$y_{ij} + x_{ji} \leq 1 \qquad \forall i, j = 1, ..., N, j < i, [ij] \notin E \qquad (4.36)$$

$$y_{ij} - x_{kj} \leq 0 \qquad \forall i, j, k = 1, ..., N, k < j[ij] \notin E, [ik] \in E \qquad (4.37)$$

$$y_{ij} + x_{jk} \leq 1 \qquad \forall i, j, k = 1, ..., N, j < k[ij] \notin E, [ik] \in E \qquad (4.38)$$

$$0 \leq y_{ik} - y_{ij} + x_{kj} \leq 1 \qquad \forall i, j, k = 1, ..., N, k < j, [ij], [ik] \notin E \qquad (4.39)$$

$$0 \leq y_{ij} - y_{ik} + x_{jk} \leq 1 \qquad \forall i, j, k = 1, ..., N, j < k, [ij], [ik] \notin E \qquad (4.40)$$

$$\sum_{i=1}^{j-1} x_{ij} + \sum_{i=j+1}^{N} (1 - x_{ji}) - \sum_{\substack{i=1 \\ [ij] \notin E}}^{N} y_{ij} + 1 \leq K \qquad \forall j = 1, ..., N \qquad (4.41)$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j = 1, ..., N \text{ with } i < j \qquad (4.42)$$

$$y_{ij} \in \{0, 1\} \qquad \forall i, j = 1, ..., N \text{ with } i \neq j \text{ and } [ij] \notin E \qquad (4.43)$$

$$K \in \mathbb{N} \qquad (4.44)$$

For a graph $G = (V, E)$ with $|V| = N$ and $|E| = M$, the number of variables x is now $N(N-1)/2$, the number of variables y is $N(N-1)-2M$; hence the dimension of the polytope is $3N(N-1)/2 - 2M$.

4.9 Conclusions

In this chapter we forged an integer programming model to solve the minimization of open stacks problem. The model is based on the fact that an instance of the problem can be put in a MOSP graph and that the solution of the problem corresponds to a set of intervals that match the duration of the stacks. The model is basically adding more arcs to the MOSP graph to create an interval graph and deciding which intervals should start before the other intervals. The objective function minimizes the maximum number of simultaneous intervals using a lower bound derived from an analysis of the linear ordering of the intervals. This is a very basic model and additional inequalities should be added to strengthen it. In the next chapter we will analyze the inequalities presented here in the light of polyhedral theory and will test the model with commercial software on benchmark instances from literature.

Chapter 5

Polyhedral Analysis and Other Valid Inequalities

This chapter aims at consolidating the model for the MOSP that was presented in the previous chapter. We start by proving that many of the inequalities presented are facets and then explore some properties of interval graphs to derive more inequalities for the model. We end the chapter discussing computational results of the several presented versions of the model.

5.1 Facets of the Polyhedron

In this section we will analyze which inequalities of the model are facets. If the polyhedron P_{IGC} defined by inequalities (4.34) to (4.43) is full-dimensional, we can prove that an inequality is a facet by exhibiting $3N(N-1)/2-2M$ affinely independent points of P that satisfy it at equality.

Let us start by recalling the main concepts in polyhedral theory.

Definition 5.1.1. A set of points $x^1, ..., x^k \in \mathbb{R}^n$ is *affinely independent* if the unique solution of

$$\begin{cases} \sum_{i=1}^{k} \alpha_i x^i & = & 0 \\ \sum_{i=1}^{k} \alpha_i & = & 0 \end{cases} \tag{5.1}$$

is $\alpha_i = 0$ for $i = 1, ..., k$.

Linear independence implies affine independence but the converse is not true:

Proposition 5.1.2. *[48] The following statements are equivalent:*

1. *The k points $x^1, ..., x^k \in \mathbb{R}^n$ are affinely independent*
2. *The $k-1$ directions $x^2 - x^1, ..., x^k - x^1$ are linearly independent*
3. *The k vectors $(x^1, 1), ..., (x^k, 1) \in \mathbb{R}^{n+1}$ are linearly independent*

Definitions 5.1.3. A *polyhedron* $P \subseteq \mathbb{R}^n$ is the set of points that satisfy a finite number of linear inequalities, that is, $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ where (A, b) is a $m \times (n+1)$ matrix. A polyhedron is *bounded* if there exists an $\omega \in \mathbb{R}_+$ such that

$$P \subseteq \{x \in \mathbb{R}^n : -\omega \leq x_j \leq \omega \ \forall j = 1, ..., n\}$$

A *polytope* is a bounded polyhedron. A polyhedron P is of dimension k, denoted by $dim(P) = k$, if the maximum number of affinely independent points in P is $k+1$. A polyhedron $P \subseteq \mathbb{R}^n$ is *full-dimensional* if $dim(P) = n$.

Full-dimensional polyhedra have a unique minimal description using a finite set of linear inequalities. This inequalities are necessary, which means that if one of those inequalities is removed, the resulting polyhedron is not the same, and sufficient, which means that every valid inequality that is not a positive multiple of one of these inequalities is redundant.

Definitions 5.1.4. The inequality $\pi x \leq \pi_0$ (or (π, π_0)) is a *valid inequality* for P if it is satisfied by all points in P. If $\pi x \leq \pi_0$ and $\mu x \leq \mu_0$ are two valid inequalities for P, we say that $\pi x \leq \pi_0$ *dominates* $\mu x \leq \mu_0$ if there exists $u > 0$ such that $\pi \geq u\mu$ and $\pi_0 \leq u\mu_0$ and $(\pi, \pi_0) \neq (u\mu, u\mu_0)$.

Note that if $\pi x \leq \pi_0$ dominates $\mu x \leq \mu_0$ then

$$\{x \in \mathbb{R}^n_+ : \pi x \leq \pi_0\} \subseteq \{x \in \mathbb{R}^n_+ : \mu x \leq \mu_0\}$$

Definition 5.1.5. If $\pi x \leq \pi_0$ is a valid inequality for P and $F = \{x \in \mathbb{R}^n_+ : \pi x = \pi_0\}$, F is called a *face* of P and we say that $\pi x \leq \pi_0$ *represents* or *defines* the face. A face F of P is called a *facet* of P if $dim(F) = dim(P) - 1$.

Proposition 5.1.6. *[59] If P is full-dimensional, a valid inequality $\pi x \leq \pi_0$ is necessary in the description of P if and only if it defines a facet of P.*

So for full-dimensional polyhedra, $\pi x \leq \pi_0$ defines a facet of P if and only if there are n affinely independent points of P satisfying it at equality.

Now let us use this definitions to find out which inequalities in our model represent facets of the polytope.

It has been proved in [30] that the 3-dicycle inequalities (4.34) and the trivial inequalities $0 \leq x_{ij} \leq 1$ are facets of the linear ordering polytope. But in our model there are other inequalities that dominate these trivial inequalities in some cases.

Proposition 5.1.7. *For $i, j \in V, i < j$ the inequality $x_{ij} \geq 0$ is not a facet of P_{IGC} if $[ij] \notin E$ or if $\exists k \in V : [kj] \notin E \wedge [ki] \in E$.*

Proof. If $[ij] \notin E$ then inequality (4.35) is valid and dominates $x_{ij} \geq 0$. If $\exists k \in V : [kj] \notin E \wedge [ki] \in E$ then inequality (4.37) can be written swapping indexes i and k as

$$y_{kj} - x_{ij} \leq 0$$

which dominates $x_{ij} \geq 0$. □

Proposition 5.1.8. *For $i, j \in V, i < j$ the inequality $x_{ij} \leq 1$ is not a facet of P_{IGC} if $[ij] \notin E$ or if $\exists k \in V : [ki] \notin E \wedge [kj] \in E$.*

Proof. If $[ij] \notin E$ the inequality (4.36) dominates $x_{ij} \geq 0$. If $\exists k \in V : [ki] \notin E \wedge [kj] \in E$, the inequality (4.38) can be written as

$$y_{ki} + x_{ij} \leq 1$$

which dominates $x_{ij} \leq 1$. □

To prove which of the valid inequalities in the model define facets, we will need to define some special points. We start by observing that the linear ordering of the vertices allows us to have points with a special structure on the variables x.

In all points defined from now on, consider that the first variables are the variables x_{ij} ordered for each j from 2 to N and for i from 1 to $j-1$ and then the variables y (ordering is irrelevant for now).

$$P = (x_{12}, x_{13}, x_{23}, x_{14}, x_{24}, x_{34}, ..., x_{1N}, ..., x_{(N-1)N}; ..., y_{ij}, ...)$$

Consider the point P^0 such that $\forall i, j \in V, i < j \; x_{ij} = y_{ij} = y_{ji} = 0$; it corresponds to a layout of the graph where all vertices are in the opposite natural order.

$$P^0 = (0, ..., 0; 0, ..., 0)$$

Points that have all variables y and the first variables x equal to zero and the remaining variables x equal to one are feasible points. Let P_{ij}^{xR} be such that $x_{ij} = 1$, $\forall a < i \; x_{aj} = 0$, $\forall a > i \; x_{aj} = 1$, $\forall b < j \; \forall a < b \; x_{ab} = 0$, $\forall b > j \; \forall a < b \; x_{ab} = 1$ and $\forall l, m : [lm] \notin E \; y_{lm} = 0$.

$$P_{ij}^{xR} = (0, ..., 0, 1, ..., 1; 0, ..., 0)$$

This is a feasible point because it corresponds to the specific ordering of the vertices in which each vertex b such that $b > j$ is at position b, vertex j is at position $j - i + 1$, and each vertex b such that $b < j$ is at position $j - b$ if $i \leq b$ or at position $j - b + 1$ if $i > b$.

Also, points that have all variables y and the last variables x equal to zero and the remaining variables x equal to one are feasible points. Let P_{ij}^{xL} be such that $x_{ij} = 1$, $\forall a < i \; x_{aj} = 1$, $\forall a > i \; x_{aj} = 0$, $\forall b < j \; \forall a < b \; x_{ab} = 1$, $\forall b > j \; \forall a < b \; x_{ab} = 0$ and $\forall l, m : [lm] \notin E \; y_{lm} = 0$.

$$P_{ij}^{xL} = (1, ..., 1, 0, ..., 0; 0, ..., 0)$$

This is a feasible point because it corresponds to the layout of the vertices in which each vertex b such that $b > j$ is at position $N + 1 - b$, vertex j is at position $N + 1 - j + i$, and each vertex b such that $b < j$ is at position $N + 1 - j + b$ if $i \geq b$ or at position $N + 1 - j + b + 1$ if $i < b$.

For N=4, these points are illustrated in Table 5.1, but there are many other configurations for the variables x that produce feasible points.

Point	x_{12}	x_{13}	x_{23}	x_{14}	x_{24}	x_{34}	y	Layout
P_{12}^{xR}	1	1	1	1	1	1	0,...,0	1234
P_{13}^{xR}	0	1	1	1	1	1	0,...,0	2134
P_{23}^{xR}	0	0	1	1	1	1	0,...,0	2314
P_{14}^{xR}	0	0	0	1	1	1	0,...,0	3214
P_{24}^{xR}	0	0	0	0	1	1	0,...,0	3241
P_{34}^{xR}	0	0	0	0	0	1	0,...,0	3421
P^0	0	0	0	0	0	0	0,...,0	4321
P_{12}^{xL}	1	0	0	0	0	0	0,...,0	4312
P_{13}^{xL}	1	1	0	0	0	0	0,...,0	4132
P_{23}^{xL}	1	1	1	0	0	0	0,...,0	4123
P_{14}^{xL}	1	1	1	1	0	0	0,...,0	1423
P_{24}^{xL}	1	1	1	1	1	0	0,...,0	1243
P_{34}^{xL}	1	1	1	1	1	1	0,...,0	1234

Table 5.1: Points P_{ij}^{xR} and P_{ij}^{xL} for $N = 4$

Define the points P^y_{ij} such that $y_{ij} = 1$ and all other variables y are zero. The variables x must be determined in order to satisfy the inequalities. Having $y_{ij} = 1$ implies that $x_{ij} = 1$ if $i < j$ (or $x_{ji} = 0$ if $j < i$) to satisfy (4.35) and (4.36). Besides, to satisfy inequalities (4.37) to (4.40), for every vertex k we must have $x_{kj} = 1$ if $k < j$ (or $x_{jk} = 0$ if $j < k$). This corresponds to a layout where vertex j is the last in the sequence. There is more than one possibility for the values of the variables x in this point, but in this case it is not important which one we choose, as long as it is feasible.

$$P^y = \left(\begin{array}{c|ccc} & 1 & \cdots & 0 \\ A & \vdots & \ddots & \vdots \\ & 0 & \cdots & 1 \end{array} \right) = \left(\begin{array}{c|c} A & I \end{array} \right)$$

Proposition 5.1.9. *The polyhedron P_{IGC} is full-dimensional.*

Proof. There are $3N(N-1)/2-2M$ variables in the polyhedron and considering the set of all points P^0, P^{xL}, P^y we have $1+N(N-1)/2+N(N-1)-2M$ affinely independent points.

$$\left(\begin{array}{ccc|ccc} 0 & \cdots & 0 & 0 & \cdots & 0 \\ \hline 1 & \cdots & 0 & & & \\ \vdots & \ddots & \vdots & & O & \\ 1 & \cdots & 1 & & & \\ \hline & & & 1 & \cdots & 0 \\ & A & & \vdots & \ddots & \vdots \\ & & & 0 & \cdots & 1 \end{array} \right)$$

□

Proposition 5.1.10. *The 3-dicycle inequalities (4.34) are facets of P_{IGC}.*

Proof. The 3-dicycle inequalities (4.34) are facets of the linear ordering polytope, which means that there is a square matrix B using only the variables x such that $\left(\begin{array}{c|c} B & O \end{array} \right)$ are $N(N-1)/2$ affinely independent points that satisfy each of the inequalities (4.34) at equality. Considering also the points P^y with suitable values for the x variables, we have $3N(N-1)/2-2M$ affinely independent points.

$$\left(\begin{array}{c|c} B & O \\ \hline A & I \end{array} \right)$$

□

Proposition 5.1.11. *The precedence inequalities (4.35) and (4.36) are facets of P_{IGC}.*

Proof. Define the points P^1_{ij} such that $x_{ij} = 0$, the remaining variables x are of the form P^{xL} or P^{xR}, and all variables y are zero. This comprehends $N(N-1)/2 - 1$ points.

$$P^1_{ij} = \left(\begin{array}{ccccccc|c} 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & \\ \vdots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots & \\ 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & \\ 0 & \cdots & 0 & 0 & 1 & \cdots & 1 & O \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & \end{array}\right)$$

Consider the point P^2_{ij} such that $y_{ij} = 1$, all other variables y are zero, $x_{ij} = 1$ if $i < j$ (or $x_{ji} = 0$ if $j < i$) and, for every other vertex k, $x_{kj} = 1$ if $k < j$ and $x_{jk} = 0$ if $j < k$. This point corresponds to a case where vertex j is the last in the linear ordering.

Define the points P^3_{ij} such that $x_{ij} = y_{ij} = 0$ and the other variables y correspond to the identity matrix. The remaining variables x are properly set to make this a feasible point. If $y_{lm} = 1$ then $x_{lm} = 1$ and, for every vertex r, $x_{rm} = 1$ if $r < m$ or $x_{mr} = 0$ if $m < r$. These are $N(N-1)-2M-1$ points where j precedes i and m is the last vertex in the ordering.

$$P^3_{ij} = \left(\begin{array}{ccc|ccc} & 0 & & 1 & \cdots & 0 \\ C & \vdots & D & \vdots & \cdots & \vdots \\ & 0 & & 0 & \cdots & 1 \end{array}\right)$$

Let the points P^4_{ji} be all the points of the form P^{xL} or P^{xR} such that $x_{ji} = 1$, and with variables y equal to zero. These are $N(N-1)/2$ linearly independent points.

$$P^4_{ji} = \left(\begin{array}{ccccccc|c} 1 & \cdots & 1 & 1 & 1 & \cdots & 1 & \\ \vdots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots & \\ 0 & \cdots & 1 & 1 & 1 & \cdots & 1 & \\ 1 & \cdots & 1 & 1 & 0 & \cdots & 0 & O \\ \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots & \\ 1 & \cdots & 1 & 1 & \cdots & 1 & 0 & \end{array}\right)$$

The points $P^0, P^1_{ij}, P^2_{ij}, P^3_{ij}$ satisfy (4.35) at equality and form a set of $3N(N-1)/2 - 2M$ affinely independent points, hence it is a facet of P_{IGC}.

Point	x			x_{ij}	x			y_{ij}	y		
P^0	0	$\cdots$	0	0	0	$\cdots$	0	0	0	$\cdots$	0
P^1_{ij}	1	$\cdots$	0	0	0	$\cdots$	0	0	O		
	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$			
	1	$\cdots$	1	0	0	$\cdots$	0	0			
	0	$\cdots$	0	0	1	$\cdots$	1	0			
	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$			
	0	$\cdots$	0	0	0	$\cdots$	1	0			
P^2_{ij}		$\cdots$		1		$\cdots$		1	0	$\cdots$	0
P^3_{ij}		C		0		D		0	1	$\cdots$	0
				$\vdots$				$\vdots$	$\vdots$	$\ddots$	$\vdots$
				0				0	0	$\cdots$	1

The points $P^4_{ji}, P^2_{ij}, P^3_{ij}$ satisfy (4.36) at equality and form a set of $3N(N-1)/2 - 2M$ affinely independent points, proving that it is a facet of P_{IGC}.

Point	x			x_{ji}	x			y_{ij}	y		
P^4_{ji}	1	$\cdots$	1	1	1	$\cdots$	1	0	O		
	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$			
	0	$\cdots$	1	1	1	$\cdots$	1	0			
	1	$\cdots$	1	1	0	$\cdots$	0	0			
	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$	$\vdots$	$\vdots$			
	1	$\cdots$	1	1	$\cdots$	1	0	0			
P^2_{ij}		$\cdots$		0		$\cdots$		1	0	$\cdots$	0
P^3_{ij}		C		1		D		0	1	$\cdots$	0
				$\vdots$				$\vdots$	$\vdots$	$\ddots$	$\vdots$
				1				0	0	$\cdots$	1

□

Proposition 5.1.12. *The Olariu inequalities (4.37) to (4.40) are facets of P_{IGC}.*

Proof. Let $P^5_{kj,ij}$ be the point such that $y_{kj} = y_{ij} = 1$, $x_{kj} = 1$, all other variables y are zero and the remaining variables x are set accordingly to satisfy all inequalities.

Consider also the points $P^6_{kj,ij}$ such that $y_{kj} = y_{ij} = 0$, $x_{kj} = 0$, all other variables y are set as in P^y and the remaining variables x are set accordingly to satisfy all inequalities.

Define the points $P^7_{jk,ij}$ such that $y_{jk} = y_{ij} = 0$, $x_{jk} = 1$, all other variables y are set as in P^y and the remaining variables x are set accordingly to satisfy all inequalities.

Let $P^8_{ij,ik}$ be the point such that $y_{ij} = y_{ik} = 0$, $x_{kj} = 0$, all other variables y are zero and the remaining variables x are set accordingly to satisfy all inequalities.

Define the points $P^9_{ij,ik}$ such that $y_{ij} = y_{ik} = 0$, $x_{kj} = 0$, all other variables y are set as in P^y and the remaining variables x are set accordingly to satisfy all inequalities.

Assume that P^{10}_{ikj} is the point such that $y_{ij} = y_{ik} = 1$, $x_{kj} = 1$, all other variables y are zero and the remaining variables x are set accordingly to satisfy all inequalities.

Let P^{11}_{ijk} be the point such that $y_{ik} = 1$, $x_{kj} = 0$, all other variables y are zero and the remaining variables x are set accordingly to satisfy all inequalities.

Define the points P^{12}_{ikj} such that $y_{ij} = y_{ik} = 0$, $x_{kj} = 1$, all other variables y are set as in P^y and the remaining variables x are set accordingly to satisfy all inequalities.

Inequality (4.37) is a facet because $P^1_{kj}, P^2_{ij}, P^5_{kj,ij}, P^6_{kj,ij}$ are $3N(N-1)/2 - 2M$ affinely independent points that satisfy it at equality.

Point	x	x_{kj}	x	y_{ij}	y_{kj}	y
P^0	$0 \cdots 0$	0	$0 \cdots 0$	0	0	$0 \cdots 0$
P^1_{kj}	$1 \cdots 0$	0	$0 \cdots 0$	0	0	O
	$\vdots \ddots \vdots$	$\vdots$	$\vdots \cdots \vdots$	$\vdots$	$\vdots$	
	$1 \cdots 1$	0	$0 \cdots 0$	0	0	
	$0 \cdots 0$	0	$1 \cdots 1$	0	0	
	$\vdots \ddots \vdots$	$\vdots$	$\vdots \ddots \vdots$	$\vdots$	$\vdots$	
	$0 \cdots 0$	0	$0 \cdots 1$	0	0	
P^2_{ij}	$\cdots$	1	$\cdots$	1	0	$0 \cdots 0$
$P^5_{kj,ij}$	$\cdots$	1	$\cdots$	1	1	$0 \cdots 0$
$P^6_{kj,ij}$	A	0	B	0	0	$1 \cdots 0$
		$\vdots$		$\vdots$	$\vdots$	$\vdots \ddots \vdots$
		0		0	0	$0 \cdots 1$

Inequality (4.38) is a facet because $P^4_{jk}, P^2_{ij}, P^2_{jk}, P^7_{jk,ij}$ are $3N(N-1)/2 - 2M$ affinely independent points that satisfy it at equality.

Point		x		x_{jk}		x		y_{ij}	y_{jk}		y	
P^4_{jk}	1	$\cdots$	1	1	1	$\cdots$	1	0	0		O	
	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$			
	0	$\cdots$	1	1	1	$\cdots$	1	0	0			
	1	$\cdots$	1	1	0	$\cdots$	0	0	0			
	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$			
	1	$\cdots$	1	1	$\cdots$	1	0	0	0			
P^2_{ij}		$\cdots$		0		$\cdots$		1	0	0	$\cdots$	0
P^2_{jk}		$\cdots$		1		$\cdots$		0	1	0	$\cdots$	0
$P^7_{jk,ij}$		A		1		B		0	0	1	$\cdots$	0
				$\vdots$				$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
				1				0	0	0	$\cdots$	1

The first part of (4.39) is a facet because $P^0, P^1_{kj}, P^2_{ij}, P^8_{ij,ik}, P^9_{ij,ik}$ are $3N(N-1)/2 - 2M$ affinely independent points that satisfy it at equality. The second part of (4.40) is equivalent to this, it merely uses x_{jk} instead of x_{kj}; hence it is also a facet.

Point		x		x_{kj}		x		y_{ij}	y_{ik}		y	
P^0	0	$\cdots$	0	0	0	$\cdots$	0	0	0	0	$\cdots$	0
P^1_{kj}	1	$\cdots$	0	0	0	$\cdots$	0	0	0		O	
	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$			
	1	$\cdots$	1	0	0	$\cdots$	0	0	0			
	0	$\cdots$	0	0	1	$\cdots$	1	0	0			
	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$			
	0	$\cdots$	0	0	0	$\cdots$	1	0	0			
P^2_{ij}		$\cdots$		1		$\cdots$		1	0	0	$\cdots$	0
$P^8_{ij,ik}$		$\cdots$		0		$\cdots$		1	1	0	$\cdots$	0
$P^9_{kj,ij}$		A		0		B		0	0	1	$\cdots$	0
				$\vdots$				$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
				0				0	0	0	$\cdots$	1

The second part of (4.39) is a facet because $P^4_{kj}, P^{10}_{ikj}, P^{11}_{ijk}, P^{12}_{ikj}$ are $3N(N-1)/2 - 2M$ affinely independent points that satisfy it at equality. The first part of (4.40) is equivalent to this, it just uses x_{jk} instead of x_{kj}; hence it is also a facet.

Point	x	x_{kj}	x	y_{ij}	y_{ik}	y
P^4_{kj}	$1 \cdots 1$	1	$1 \cdots 1$	0	0	O
	$\vdots \ddots \vdots$	$\vdots$	$\vdots \cdots \vdots$	$\vdots$	$\vdots$	
	$0 \cdots 1$	1	$1 \cdots 1$	0	0	
	$1 \cdots 1$	1	$0 \cdots 0$	0	0	
	$\vdots \ddots \vdots$	$\vdots$	$\ddots \ddots \vdots$	$\vdots$	$\vdots$	
	$1 \cdots 1$	1	$\cdots 1 \; 0$	0	0	
P^{10}_{ikj}	$\cdots$	1	$\cdots$	1	1	$0 \cdots 0$
P^{11}_{ijk}	$\cdots$	0	$\cdots$	0	1	$0 \cdots 0$
P^{12}_{jkj}	A	1 $\vdots$ 1	B	0 $\vdots$ 0	0 $\vdots$ 0	$1 \cdots 0$ $\vdots \ddots \vdots$ $0 \cdots 1$

□

5.2 Other Valid Inequalities

The inequalities that we have seen so far are sufficient to have a valid model and to guarantee that the solution will be an interval graph. But we can strengthen our model by adding some constraints related to additional properties of interval graphs.

5.2.1 Neighbor of Successor Inequalities

The MOSP model can be strengthened with the following three inequalities that can be proved simply by observing that in an interval graph both variables on the left are not allowed to be simultaneously equal to one without contradicting Theorem 3.4.9.

$$y_{ij} + y_{ki} \leq 1 \quad \forall i, j, k \in V \text{ with } [ij], [ik] \notin E, [jk] \in E \tag{5.2}$$

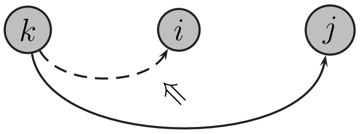
$$y_{ij} + y_{jk} \leq 1 \quad \forall i, j, k \in V \text{ with } [ij], [jk] \notin E, [ik] \in E \tag{5.3}$$

$$y_{ij} + y_{lk} \leq 1 \quad \forall i, j, k, l \in V \text{ with } [ij], [kl] \notin E, [jl], [ik] \in E \tag{5.4}$$

The inequality (5.2) says that a neighbor of the successor of vertex i, which is vertex k, cannot end before vertex i opens.

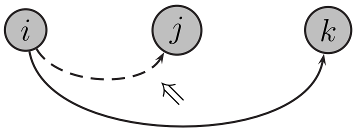
If both the variables on the left hand side of the inequality (5.2) were $y_{ij} = y_{ki} = 1$, then the three vertices were linearly order as in $k \prec i \prec j$.

As $[jk] \in E$, Theorem 3.4.9 would force to have the arc $[ki] \in F$, asserted by $y_{ki} = 0$, which contradicts the initial assumption.



The inequality (5.3) states that if vertex k is a neighbor of vertex i, it cannot open after the closing of a successor of vertex i, which is represented by vertex j.

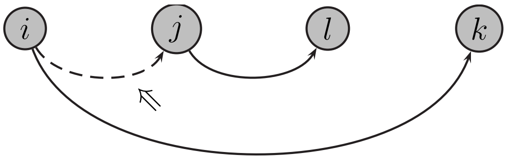
If both the variables on the left hand side of the inequality (5.3) were $y_{ij} = y_{jk} = 1$, then the linear order of the three vertices would be $i \prec j \prec k$. As $[ik] \in E$, Theorem 3.4.9 would force adding the arc $[ij] \in F$, making $y_{ij} = 0$, which is absurd.



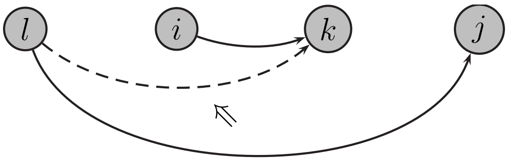
Finally, the inequality (5.4) declares that a neighbor l of the successor j of vertex i cannot close before the neighbor k of vertex i opens.

If the two variables are considered 1 as in $y_{ij} = y_{lk} = 1$, then the linear order of the four vertices should satisfy $i \prec j$ and $l \prec k$. Now there are two possible cases.

If $j \prec k$, as $[ik] \in E$, by Theorem 3.4.9 then $[ij] \in F$, making $y_{ij} = 0$, which is absurd.



If $k \prec j$, then the linear ordering would be $l \prec k \prec j$ and the existence of the arc $[jl] \in E$ would make the arc $[lk] \in F$, stated by $y_{lk} = 0$ which is absurd.



5.2.2 Co-comparability Graph

For the solution graph $H = (V, E \cup F)$ to be an interval graph, its complement $\overline{H}$ must be a comparability graph. The ordering of the vertices must respect transitivity in the complement graph and must not have direct cycles. If the arcs $[ij]$ and $[jk]$ exist in the complement graph, with an orientation $i \prec j$ and $j \prec k$, then if the arc $[ik]$ exists, it must be oriented as in $i \prec k$.

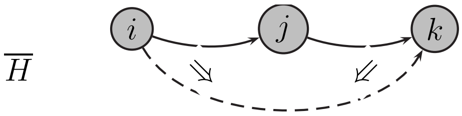


Figure 5.1: $\overline{H}$ must be transitively orientable

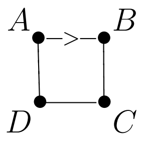
The transitivity of the relation between the variables y comes from the comparability graph property and forces an ordering of the vertices. If a direction is defined in an arc of a graph, that will determine the flow of all the other ones. The variables y define the complement graph, because y_{ij} equals 1 when the arc $[ij] \notin F$, hence it exists in the complement graph $\overline{H}$ and the orientation of the vertices is $i \prec j$. The transitivity in $\overline{H}$ is expressed by

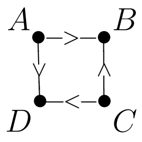
$$y_{ij} = y_{jk} = 1 \Rightarrow y_{ik} = 1$$

This can be assured by the following statement for every $i \neq j \neq k$ such as the arcs $[ij], [ik], [jk]$ did not exist in the initial MOSP graph:

$$y_{ij} + y_{jk} - 1 \leq y_{ik} \quad \forall i, j, k \in V, [ij], [jk], [ik] \notin E \tag{5.5}$$

For example, consider the following graph. If we define an ordering of the vertices from A to B, then B must come after C because otherwise having A to B and B to C by transitivity we should also have the arc A to C, which does not exist. By similar reasons, the other arcs have the following orientations: C to D and A to D.





Let us analyze another example, an instance of the MOSP problem with five different items and eight patterns taken from [18, p.322].

Patterns:	P1	P2	P3	P4	P5	P6	P7	P8
Item 1	X	X				X	X	
Item 2				X	X			
Item 3			X				X	
Item 4		X		X				X
Item 5			X		X	X		

Table 5.2: An instance for the MOSP with 5 items and 8 patterns

This instance originates a MOSP graph with 5 vertices, one corresponding to each item, and with edges between the vertices (items) that belong to the same pattern.

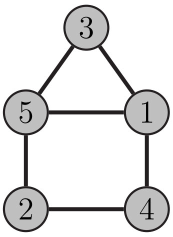
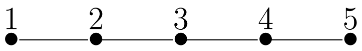


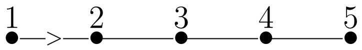
Figure 5.2: MOSP Graph of the instance in Table 5.2

The graph correspondent to this instance is not yet an interval graph. We need to add more arcs so it will become chordal and its complement graph will become a comparability graph.

Initially, the complement graph of the MOSP graph in Figure 5.2 is:



If we set an orientation on the first arc of the complement graph, for example, from vertex 1 to vertex 2, it will mean that $y_{12} = 1$.



Because $\overline{H}$ must be a comparability graph, one of the following must happen:

$$y_{32} = 1 \wedge y_{23} = 0 \qquad \text{or} \qquad y_{32} = y_{23} = 0$$

meaning that either an arc exists in $\overline{H}$ linking vertices 2 and 3 with orientation from vertex 3 to 2, or that arc does not even exist. This will correspond to the inequality

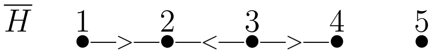
$$y_{12} + y_{23} \leq 1$$

and to another inequality to be an interval graph

$$y_{21} - y_{23} \leq x_{31}$$

This last inequality means that, for example, if $y_{21} = 1$ and $x_{13} = 1$ then $y_{23} = 1$. In terms of the intervals, this says that if interval 2 closes before interval 1 opens and interval 1 opens before interval 3 opens, then interval 2 closes before interval 3 opens.

A possible solution for this instance of the problem corresponds to, in the end of the process of finding a solution, having the arc [45] removed from the complement graph and the linear ordering as in the following picture:



The complement of this graph is the interval graph H that corresponds to the solution of the problem:

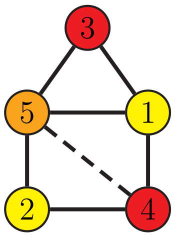
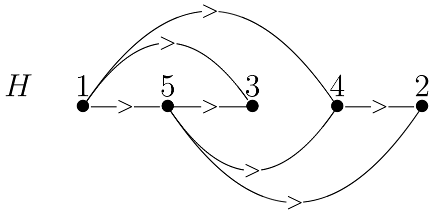


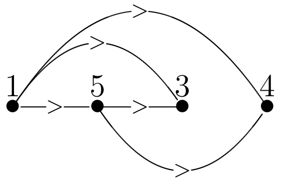
Figure 5.3: Interval graph corresponding to the solution of the instance in Table 5.2

An ordering defined for the vertices of the complement graph $\overline{H}$ will correspond to the same ordering for the vertices of the interval graph H.

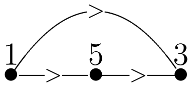
The interval graph of this instance can then be sketched with the vertices in a straight line sorted by their linear ordering.



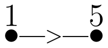
In this graph, the perfect elimination scheme would first eliminate vertex 2 and its associated arcs.



Then vertex 4 should be eliminated, resulting in:



After that, one would eliminate vertex 3, followed by vertex 5 and finally vertex 1.



The reverse order of the perfect elimination scheme sets the order of the beginning of the intervals: 1-5-3-4-2.

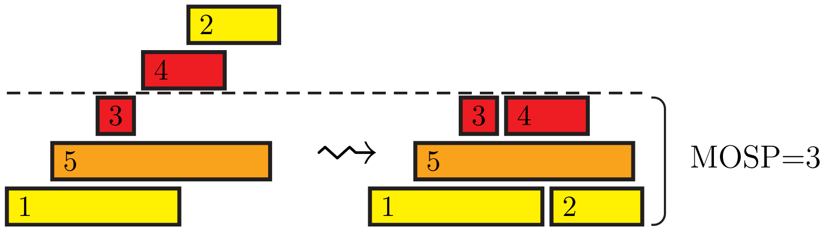


Figure 5.4: Optimal solution of the example in Table 5.2

This sequence of vertices corresponds to the sequence of patterns P1-P6-P3-P7-P8-P2-P4-P5.

As there are some stacks that are not simultaneous at any time, like 3 and 4, or 1 and 2, those stacks can use the same stack space, hence this sequence of patterns gives a maximum of three simultaneously open stacks, that is the optimum for this instance, as can be seen in Figure 5.4.

5.2.3 Chords in k-cycles

Another way to reinforce the model is to reduce the number of arcs that are added to the original MOSP graph. If the graph G is completed to become an interval graph, it has to be chordal, so in every k-cycle for $k \geq 4$ sufficient chords must be added. In a 4-cycle defined by the ordered vertices $\{ijkl\}$, we need to add at least one of the arcs in the diagonal.

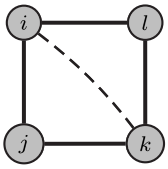


Figure 5.5: A 4-cycle must have a chord

In the complement graph there will remain the other diagonal arc, or none, whose existence is flagged by the variables y. The need to add one of the arcs $[ik]$, $[ki]$, $[jl]$ or $[lj]$ can be expressed by the restriction:

$$y_{ik} + y_{ki} + y_{jl} + y_{lj} \leq 1 \quad \forall [ik],[jl] \notin E, [ij],[jk],[kl],[li] \in E \tag{5.6}$$

Lemma 5.2.1. *Constraints (5.6) can be derived from the Olariu's constraints (4.10).*

Proof. Consider the following Olariu's inequalities:

$$y_{ik} \leq x_{lk}, y_{jl} \leq x_{kl}$$

$$y_{ik} \leq x_{jk}, y_{lj} \leq x_{kj}$$

$$y_{ki} \leq x_{li}, y_{jl} \leq x_{il}$$

$$y_{ki} \leq x_{ji}, y_{lj} \leq x_{ij}$$

Therefore,

$$y_{ik} + y_{jl} \leq 1$$

$$y_{ik} + y_{lj} \leq 1$$

$$y_{ki} + y_{jl} \leq 1$$

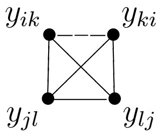
$$y_{ki} + y_{lj} \leq 1$$

Furthermore,

$$y_{ik} + y_{ki} \leq 1$$
$$y_{lj} + y_{jl} \leq 1$$

The six constraints combined enforce that, when a binary variable y_{pq}, with $pq \in \{lj, jl, ik, ki\}$ takes the value 1, all the other variables in the set take the value 0. Therefore, the constraint (5.6) for the non-chordal 4-cycle follows.

In fact, the constraint is a *clique constraint* derived from a 4 vertex incompatibility graph: each vertex corresponds to a y variable and each of the six edges corresponds to a constraint that states that we can only select one of the vertices at the ends of the edge. The graph is a clique, and only one of the vertices can be selected.



□

A 5-cycle needs at least two chords, but not any chord will do, because the two chords must share a vertex, as in Figure 5.6(a).

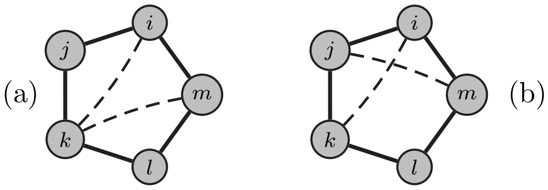


Figure 5.6: 5-cycles with two chords: (a) is chordal, (b) is not chordal

In a 5-cycle defined by the ordered vertices $\{ijklm\}$, the inequality should be:

$$y_{il} + y_{li} + y_{ik} + y_{ki} + y_{jl} + y_{lj} + y_{jm} + y_{mj} + y_{mk} + y_{km} \leq 3 \qquad (5.7)$$
$$\forall [ik], [il], [jl], [jm], [km] \notin E, [ij], [jk], [kl], [lm], [mi] \in E$$

The number 3 comes from the fact that we need at least 2 chords from the possible 5 chords, so there must be at most 3 chords left in the complement graph, causing the defined variables y to sum at most 3.

Lemma 5.2.2. *Constraints (5.7) can be derived from the Olariu's constraints (4.10).*

Proof. Consider the following Olariu's inequalities in a 5-cycle $\{ijklm\}$:

$$\begin{array}{lll}
y_{mk} \leq x_{ik}, & y_{mk} \leq x_{lk}, & y_{mk} \leq x_{mk} \\
y_{li} \leq x_{ki}, & y_{li} \leq x_{mi}, & y_{li} \leq x_{li} \\
y_{ki} \leq x_{li}, & y_{ki} \leq x_{ji}, & y_{ki} \leq x_{ki} \\
y_{jl} \leq x_{il}, & y_{jl} \leq x_{kl}, & y_{jl} \leq x_{jl} \\
y_{il} \leq x_{jl}, & y_{il} \leq x_{ml}, & y_{il} \leq x_{il} \\
y_{mj} \leq x_{lj}, & y_{mj} \leq x_{ij}, & y_{mj} \leq x_{mj} \\
y_{km} \leq x_{jm}, & y_{km} \leq x_{lm}, & y_{km} \leq x_{km} \\
y_{lj} \leq x_{mj}, & y_{lj} \leq x_{kj}, & y_{lj} \leq x_{lj} \\
y_{jm} \leq x_{km}, & y_{jm} \leq x_{im}, & y_{jm} \leq x_{jm} \\
y_{ik} \leq x_{mk}, & y_{ik} \leq x_{jk}, & y_{ik} \leq x_{ik}
\end{array}$$

Therefore, summing up in groups of two the inequalities from the first column,

$$y_{mk} + y_{li} \leq 1$$

$$y_{ki} + y_{jl} \leq 1$$

$$y_{il} + y_{mj} \leq 1$$

$$y_{km} + y_{lj} \leq 1$$

$$y_{jm} + y_{ik} \leq 1$$

Furthermore, summing up the inequalities in the second column that refer to symmetric variables x, we have:

$$y_{jm} + y_{li} \leq 1$$

$$y_{mj} + y_{ki} \leq 1$$

$$y_{ik} + y_{lj} \leq 1$$

$$y_{jl} + y_{mk} \leq 1$$

$$y_{km} + y_{il} \leq 1$$

Combining in pairs the inequalities in the third column we get:

$$y_{il} + y_{li} \leq 1$$

$$y_{ik} + y_{ki} \leq 1$$

$$y_{jl} + y_{lj} \leq 1$$

$$y_{jm} + y_{mj} \leq 1$$

$$y_{km} + y_{mk} \leq 1$$

And finally crossing the inequalities in the first column with the correspondent inequalities in the third column we obtain ten more inequalities:

$$y_{il} + y_{lj} \leq 1$$

$$y_{li} + y_{ik} \leq 1$$

$$y_{ik} + y_{km} \leq 1$$

$$y_{ki} + y_{il} \leq 1$$

$$y_{jl} + y_{li} \leq 1$$

$$y_{lj} + y_{jm} \leq 1$$

$$y_{jm} + y_{mk} \leq 1$$

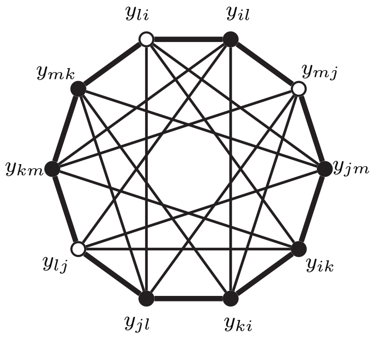
$$y_{mj} + y_{jl} \leq 1$$

$$y_{km} + y_{mj} \leq 1$$

$$y_{mk} + y_{ki} \leq 1$$

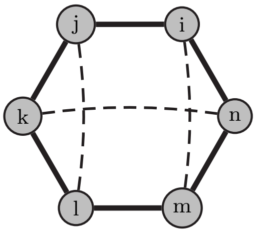
The 25 constraints combined enforce that, when a binary variable y_{pq}, with $pq \in \{ik, ki, il, li, jl, lj, jm, mj, km, mk\}$ takes the value 1, only two more of the other variables in the set can take the value 1. Therefore, the constraint (5.7) for the non-chordal 5-cycle follows.

In fact, the constraint can be derived from a 10 vertex incompatibility graph: each vertex corresponds to a y variable and each of the 25 edges corresponds to a constraint that states that we can only select one of the vertices at the ends of the edge. In this graph there is a central symmetry, each vertex has degree 5, and we can only select at most three vertices.

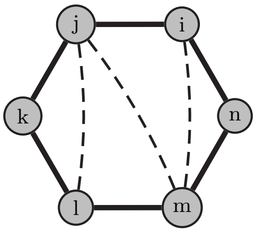


□

Similar constraints can be made to k-cycles of greater order that can be found in the initial graph. For the resulting graph to be chordal, we will need to add to each k-cycle only the appropriate $k-3$ chords. Not every choice of $k-3$ chords works, as we can see in the following picture.

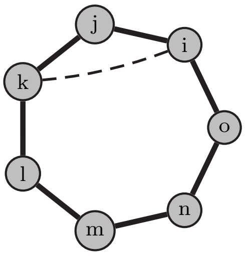


A 6-cycle with two chords that is not chordal



A 6-cycle with two chords that is chordal

This result is easily proved by induction. For any 4-cycle we just need to add one directed arc between two of the non-adjacent vertices. Let us assume that in any k-cycle, for $k \geq 4$, we just need to add $k-3$ arcs to make it chordal. Given a $(k+1)$-cycle, we start by adding an arc between one of the vertices and, not the next, but the second following vertex in the cycle.



A 7-cycle with a chord becomes a 6-cycle and a 3-cycle

We get a 3-cycle and a remaining k-cycle, so by induction this remaining k-cycle needs $k-3$ arcs to become chordal, giving a total of $k-2$ arcs to the $(k+1)$-cycle to become chordal.

If we had chosen to add a different chord to the $(k+1)$-cycle, we would obtain a i-cycle and a $(k+3-i)$-cycle, for $3 \leq i \leq k$. In the i-cycle we would need to add $i-3$ arcs and in the $(k+3-i)$-cycle $k-i$ arcs, giving a total number of arcs:

$$1 + (i-3) + (k-i) = k+2 = (k+1) - 3$$

proving the result.

Hence for a k-cycle to be chordal, we just need to add $k-3$ chords. But we may need to have more than these $k-3$ chords, if we also want it to be an interval graph. We need the extra arcs to have the sequence of cliques. So the number or chords we add in a k-cycle must be greater than or equal to $k-3$.

The sum of the variables y defined must be at most the number of arcs in a clique of size k minus the lateral arcs that form the cycle, minus the $k-3$ necessary chords, performing a total of:

$$\frac{k(k-1)}{2} - k - (k-3) = \frac{(k-3)(k-2)}{2}$$

Hence the inequality for a general k-cycle is:

$$\sum_{\substack{i,j \in k\text{-cycle} \\ [ij] \notin E}} y_{ij} \leq \frac{(k-3)(k-2)}{2}$$

5.2.4 Coloring the Vertices of the Interval Graph

The value of the optimum of the MOSP is equal to the size of the biggest clique in the solution graph $\omega(H)$ and, because interval graphs are perfect graphs, it is equal to the chromatic number of the graph $\chi(H)$, which is the number of colors needed to assign to the vertices of the graph such that there are no two adjacent vertices of the same color.

To explain the relations between the intervals horizontally, we will add an extra set of variables z, based on the asymmetric representatives formulation for the vertex coloring problem by Campêlo et al. [12].

Definition 5.2.3. Given a graph $G = (V, E)$, a linear ordering of V, and a coloring of V such that adjacent vertices have different colors, we say that a vertex $i \in V$ is a *representative* if i precedes all other vertices with the same color of i. We say that vertex $i \in V$ *represents* vertex $j \in V$ if i and j have the same color and if i is a representative.

For example, in the graph correspondent to Figure 5.4, vertex 3 would represent vertex 4.

If we assign colors to the vertices of the desired interval graph, such that no two adjacent vertices have the same color, we can count the maximum number of simultaneously open stacks by counting the minimum number of different colors needed, because simultaneously open stacks will get different colors, and stacks that do not overlap can have the same color.

The variables that we will use are:

$$z_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ represents vertex } j \\ 0 & \text{otherwise} \end{cases} \qquad \forall i, j \in V : [ij] \notin E$$

Note that if $i \in V$ is a representative then $z_{ii} = 1$.

Therefore the number of different colors is counted by the number of representatives vertices:

$$\sum_{i=1}^{N} z_{ii} = K \tag{5.8}$$

There is also a constraint to set that all N vertices must have representatives

$$\sum_{\substack{i=1 \\ [ij]\notin E}}^{N} \sum_{\substack{j=1 \\ [ij]\notin E}}^{N} z_{ij} = N \tag{5.9}$$

but each vertex has only one representative:

$$\sum_{\substack{i=1 \\ [ij]\notin E}}^{N} z_{ij} = 1 \quad \forall j = 1, ..., N \tag{5.10}$$

A vertex i represents a vertex j ($z_{ij} = 1$) only if i is a representative vertex

$$z_{ij} \leq z_{ii} \quad \forall i, j = 1, ..., N \text{ with } [ij] \notin E \tag{5.11}$$

and only if i and j share the same stack ($y_{ij} = 1$):

$$z_{ij} \leq y_{ij} \quad \forall i, j = 1, ..., N \text{ with } [ij] \notin E \tag{5.12}$$

In the previous example (see Figure 5.7), vertex 3 can only represent vertices 2 or 4, which is stated by inequalities (5.12), because variables y_{32} and y_{34} are the only non-null variables of type y_{3j} in the solution, and only if $z_{33} = 1$, which means that interval 3 is the “leader”of a new stack.

If there is a vertex that represents more than one vertex, then all those vertices must share the same stack space. That possibility happens only when variables y become 1.

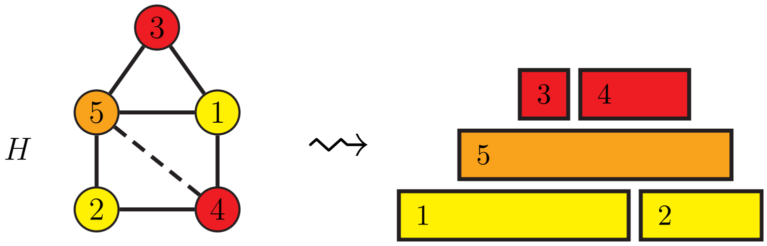


Figure 5.7: Colored interval graph and interval representation of the solution of the example in Table 5.2

Consider three non adjacent vertices $i, j, k \in V$ such that in the solution i is a representative vertex of j and k. In that case, $[jk] \notin F$ and all three vertices must share the same stack space. Inequality (5.12) is not enough, because it only forces $y_{ij} = 1$ and $y_{ik} = 1$ saying that i and j are not simultaneous intervals, neither i and k. But j and k must not be simultaneous too, because they share the same color. So either j finishes before k starts (making $y_{jk} = 1$) or k finishes before j starts ($y_{kj} = 1$).

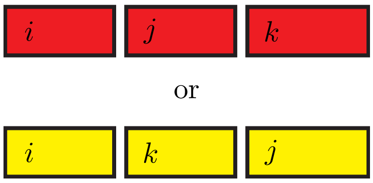


Figure 5.8: Possible interval representations for an interval graph where vertex i is a representative of vertices j and k

Conversely, if $[jk] \in F$ then $y_{jk} = y_{kj} = 0$, making impossible for vertex i to represent both the vertices j and k. The corresponding inequality is then:

$$z_{ij} + z_{ik} \leq y_{jk} + y_{kj} + 1 \qquad \forall i, j, k \in V : [ij], [ik], [jk] \notin E \tag{5.13}$$

If there are more than three vertices in this situation, the inequalities (5.12) force the representative vertex to close before every one of its represented vertices opens, and inequalities (5.13) allow each pair of the represented vertices to share the same stack by forcing one of the variables y in the pair to be 1. The transitivity of the linear ordering extends this sharing of the stack for all the represented vertices.

A vertex in the anti-neighborhood of each clique can represent only one vertex of the clique. For a vertex i in the anti-neighborhood of a 2-clique $\{jk\}$ we have the inequality

$$z_{ij} + z_{ik} \leq z_{ii} \qquad \forall i, j, k \in V : [ij], [ik] \notin E, [jk] \in E \tag{5.14}$$

There is a similar inequality for a 3-clique $\{jkl\}$

$$z_{ij} + z_{ik} + z_{il} \leq z_{ii} \qquad \forall i, j, k, l \in V : [ij], [ik], [il] \notin E, [jk], [kl], [lj] \in E \tag{5.15}$$

and for a 4-clique $\{jklm\}$

$$z_{ij} + z_{ik} + z_{il} + z_{im} \leq z_{ii} \qquad \forall i, j, k, l, m \in V : [ij], [ik], [il], [im] \notin E, [jk], [jl], [jm], [kl], [km], [lm] \in E \tag{5.16}$$

In our example, we can see in Figure 5.7 that there is a 2-clique $\{2, 4\}$ which originates the inequality

$$z_{32} + z_{34} \leq z_{33}$$

which means that vertex 3, which is in the anti-neighborhood of that clique, can only represent vertex 2 or vertex 4 and only if 3 is a representative, i.e., interval 3 can use the same stack space as interval 2 or interval 4 but not both, and only if interval 3 is the “leader”of that stack (see Figure 5.4).

We can also derive an inequality for coloring the induced cycles of the graph. In a 5-cycle we need to add at least 2 chords, maintaining the minimum number of colors 3, as depicted in Figure 5.9.

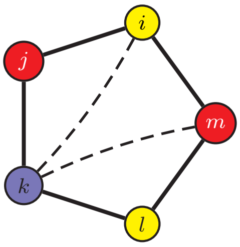


Figure 5.9: A minimum colored 5-cycle after adding 2 chords

In result, there is a maximum of two pairs of vertices with the same color; therefore we can have at most 2 vertices that represent other vertices of that 5-cycle, making the sum of the variables z involved in this cycle at most 2.

$$z_{il} + z_{li} + z_{ik} + z_{ki} + z_{jl} + z_{lj} + z_{jm} + z_{mj} + z_{mk} + z_{km} \leq 2 \\ \forall i,j,k,l,m \in V : [ij],[jk],[kl],[lm],[mi] \in E \text{ and } [ik],[il],[jl],[jm],[km] \notin E \tag{5.17}$$

5.3 The Revised Formulation of the Model for MOSP

Given an instance of a MOSP problem, let the graph $G = (V; E)$ be the associated graph and $|V| = N$ and $|E| = M$. The mathematical formulation that was presented in Chapter 4 can now be improved considering the variables and the inequalities explained in the previous sections of this Chapter.

Therefore, our revised integer programming formulation for the MOSP combines the basic inequalities of the model studied in the previous chapter, either in their first form (4.7) to (4.15), or using the reduced number of variables (4.34) to (4.44), with the new inequalities (5.2) to (5.17) developed in this chapter.

Minimize K

Subject to:

$$0 \leq x_{ij} + x_{jk} - x_{ik} \leq 1 \qquad \forall i,j,k = 1,...,N, i < j < k \tag{5.18}$$

$$y_{ij} - x_{ij} \leq 0 \qquad \forall i,j = 1,...,N, i < j, [ij] \notin E \tag{5.19}$$

$$y_{ij} + x_{ji} \leq 1 \qquad \forall i,j = 1,...,N, j < i, [ij] \notin E \tag{5.20}$$

$$y_{ij} - x_{kj} \leq 0 \qquad \forall i,j,k = 1,...,N, k < j, [ij] \notin E, [ik] \in E \tag{5.21}$$

$$y_{ij} + x_{jk} \leq 1 \qquad \forall i,j,k = 1,...,N, j < k, [ij] \notin E, [ik] \in E \tag{5.22}$$

$$0 \leq y_{ik} - y_{ij} + x_{kj} \leq 1 \qquad \forall i,j,k = 1,...,N, k < j, [ij],[ik] \notin E \tag{5.23}$$

$$0 \leq y_{ij} - y_{ik} + x_{jk} \leq 1 \qquad \forall i,j,k = 1,...,N, j < k, [ij],[ik] \notin E \tag{5.24}$$

$$\sum_{i=1}^{j-1} x_{ij} + \sum_{i=j+1}^{N} (1 - x_{ji}) - \sum_{\substack{i=1 \\ [ij]\notin E}}^{N} y_{ij} + 1 \leq K \qquad \forall j = 1,...,N \tag{5.25}$$

$$y_{ij} + y_{ki} \leq 1 \qquad \forall i,j,k = 1,...,N \text{ with } [ij],[ik] \notin E, [jk] \in E \tag{5.26}$$

$$y_{ij} + y_{jk} \leq 1 \qquad \forall i,j,k = 1,...,N \text{ with } [ij],[jk] \notin E, [ik] \in E \tag{5.27}$$

$$y_{ij} + y_{lk} \leq 1 \qquad \forall i,j,k,l = 1,...,N \text{ with } [ij],[kl] \notin E, [jl],[ik] \in E \tag{5.28}$$

$$y_{ij} + y_{jk} - y_{ik} \leq 1 \qquad \forall i,j,k = 1,...,N \text{ with } [ij],[jk],[ik] \notin E \tag{5.29}$$

$$y_{ik} + y_{ki} + y_{jl} + y_{lj} \leq 1 \qquad \begin{array}{l} \forall i,j,k,l = 1,...,N \text{ with } i \neq j \neq k \neq l, \\ [ik],[jl] \notin E, [ij],[jk],[kl],[li] \in E \end{array} \tag{5.30}$$

$$\begin{array}{r} y_{il} + y_{li} + y_{ik} + y_{ki} + y_{jl} + \\ + y_{lj} + y_{jm} + y_{mj} + y_{mk} + y_{km} \end{array} \leq 3 \qquad \begin{array}{l} \forall i,j,k,l,m = 1,...,N \text{ with } i \neq j \neq k \neq l \neq m, \\ [ik],[il],[jl],[jm],[km] \notin E, [ij],[jk],[kl],[lm],[mi] \in E \end{array} \tag{5.31}$$

$$\sum_{i=1}^{N} z_{ii} = K \tag{5.32}$$

$$\sum_{\substack{i=1 \\ [ij]\notin E}}^{N} \sum_{\substack{j=1 \\ [ij]\notin E}}^{N} z_{ij} = N \tag{5.33}$$

$$\sum_{\substack{i=1 \\ [ij]\notin E}}^{N} z_{ij} = 1 \qquad \forall j = 1, ..., N \tag{5.34}$$

$$z_{ij} \leq y_{ij} \qquad \forall i, j = 1, ..., N \text{ with } [ij] \notin E \tag{5.35}$$

$$z_{ij} + z_{ik} - y_{jk} - y_{kj} \leq 1 \qquad \forall i, j, k = 1, ..., N \text{ with } [ij], [ik], [jk] \notin E \tag{5.36}$$

$$z_{ij} \leq z_{ii} \qquad \forall i, j = 1, ..., N \text{ with } [ij] \notin E \tag{5.37}$$

$$z_{ij} + z_{ik} \leq z_{ii} \qquad \forall i, j, k = 1, ..., N \text{ with } j < k, [ij], [ik] \notin E, [jk] \in E \tag{5.38}$$

$$z_{ij} + z_{ik} + z_{il} \leq z_{ii} \qquad \begin{array}{l} \forall i, j, k, l = 1, ..., N \text{ with } j < k < l, \\ [ij], [ik], [il] \notin E, [jk], [kl], [lj] \in E \end{array} \tag{5.39}$$

$$z_{ij} + z_{ik} + z_{il} + z_{im} \leq z_{ii} \qquad \begin{array}{l} \forall i, j, k, l, m = 1, ..., N \text{ with } j < k, j < l, k < m, \\ [ij], [ik], [il], [im] \notin E, [jk], [jl], [jm], [kl], [km], [lm] \in E \end{array} \tag{5.40}$$

$$\begin{array}{r} z_{il} + z_{li} + z_{ik} + z_{ki} + z_{jl} + \\ + z_{lj} + z_{jm} + z_{mj} + z_{mk} + z_{km} \quad \leq 2 \end{array} \qquad \begin{array}{l} \forall i, j, k, l, m = 1, ..., N \text{ with } i \neq j \neq k \neq l \neq m, \\ [ik], [il], [jl], [jm], [km] \notin E, [ij], [jk], [kl], [lm], [mi] \in E \end{array} \tag{5.41}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j = 1, ..., N \text{ with } i < j \tag{5.42}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i, j = 1, ..., N \text{ with } i \neq j, [ij] \notin E \tag{5.43}$$

$$z_{ij} \in \{0, 1\} \qquad \forall i, j = 1, ..., N \text{ with } [ij] \notin E \tag{5.44}$$

$$K \in \mathbb{N} \tag{5.45}$$

In this formulation, the optimum value for the MOSP is represented by K, and expression (5.25) assures it has a lower bound. Inequalities (5.18) choose the linear ordering of the vertices.

The statements (5.19) to (5.24) relate the variables x and y. The implication between closing precedences and opening precedences is settled by (5.19) and (5.20). The following, (5.21) to (5.24) have to do with the characterization of an interval graph by Olariu, which is reinforced by inequalities (5.26) to (5.28).

Inequality (5.29) declares that the complement graph has to be a comparability graph. Equations (5.30) and (5.31) add the necessary number of chords in 4-cycles and 5-cycles, respectively.

The coloring of the stacks is expressed by inequalities (5.32) to (5.41). The first of these inequalities identifies the number of colors as the number of simultaneously open stacks. The next two inequalities (5.33) and (5.34) say that every vertex has one and only one representative.

The inequalities (5.35) and (5.36) relate the y variables with the z variables. The first one states that if a vertex represents another vertex then they cannot be adjacent in the interval graph, and the second inequality says that if two vertices are represented by a same third vertex, they have the same color and thus there must not be any arc between them.

The next four inequalities (5.37) to (5.40) are concerned with the number of representatives that can exist in the anti-neighborhood of a clique, and (5.41) restrict the number of representatives that can exist in a 5-cycle.

Last, (5.42), (5.43) and (5.44) state that both the variables x, y and z are binary integers, y, z are defined only for items that are not present in the same pattern, and (5.45) defines the integer variable that counts the number of simultaneous open stacks.

5.4 Computational Tests

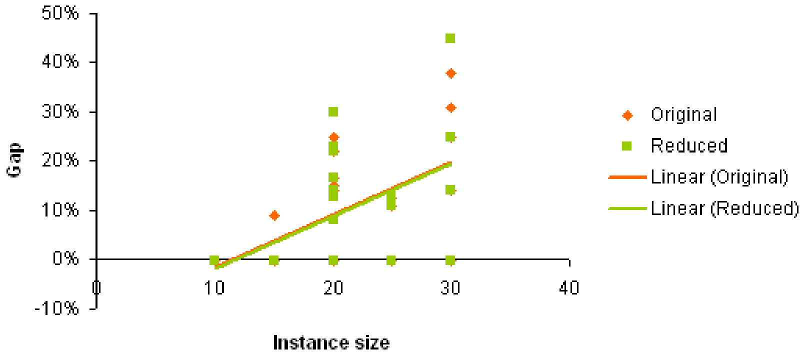
The original integer programming model with all the inequalities discussed in this chapter and the version with reduced number of variables were tested on the instances of the Constraint Modeling Challenge 2005, available at:

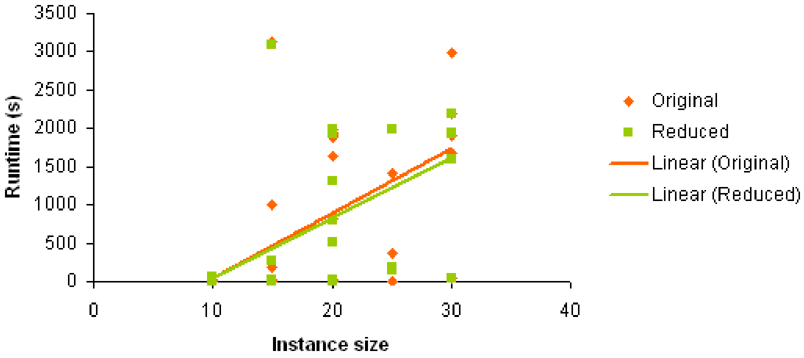
`http://www.cs.st-andrews.ac.uk/∼ipg/challenge/instances.html`

The instances were provided by the participants in the challenge and present different kinds of difficulty, such as size, sparseness and symmetry. Computational tests were performed with ILOG OPL Development Studio 5.5 on an Intel®Core2 Duo T7200@2.00GHz 0.99GB RAM. For each instance, the best objective value found by the model, the best lower bound, the gap, the number of nodes of the search tree and the runtime were recorded.

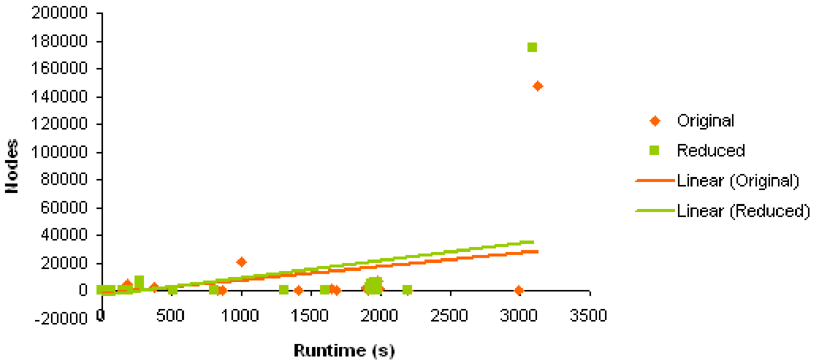
In small instances we found the optimal solution in just a few seconds. In larger instances we found the optimal solution in a few seconds as well, but it takes too long to prove that it is optimal, specially in instances with many symmetries. In really large instances the models could not be started because there was not enough memory to handle so many variables and inequalities.

The model with reduced number of variables improved the gap and the runtime in some of the larger instances, but not in all of them. The cells in green in Table 5.3 represent the cases where the reduced model improved the gap, the runtime or the number of nodes.



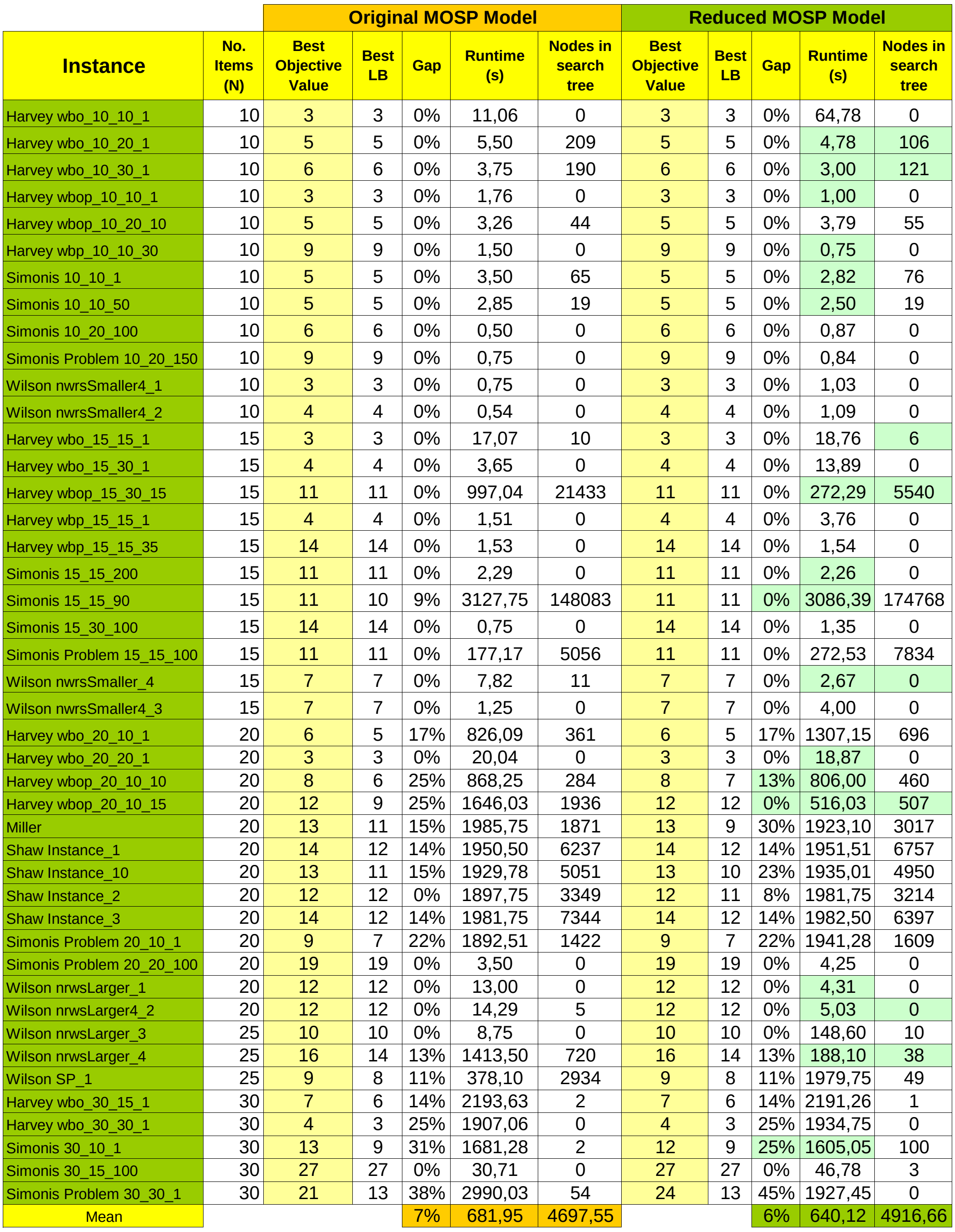


According to the charts, the reduced model performed slightly better in terms of runtime and number of nodes in the branching, but the difference between the models is not significant.



5.5 Conclusions

In this Chapter we proved that the basic model proposed for the minimization of open stacks is a full-dimensional polytope with most of the valid inequalities in it representing facets. We have developed more inequalities and have added them to the model to strengthen the formulation. Finally we have presented computational results for the two versions of the model discussed for the minimization of open stacks.



		Original MOSP Model					Reduced MOSP Model				
Instance	No. Items (N)	Best Objective Value	Best LB	Gap	Runtime (s)	Nodes in search tree	Best Objective Value	Best LB	Gap	Runtime (s)	Nodes in search tree
Harvey wbo_10_10_1	10	3	3	0%	11,06	0	3	3	0%	64,78	0
Harvey wbo_10_20_1	10	5	5	0%	5,50	209	5	5	0%	4,78	106
Harvey wbo_10_30_1	10	6	6	0%	3,75	190	6	6	0%	3,00	121
Harvey wbop_10_10_1	10	3	3	0%	1,76	0	3	3	0%	1,00	0
Harvey wbop_10_20_10	10	5	5	0%	3,26	44	5	5	0%	3,79	55
Harvey wbp_10_10_30	10	9	9	0%	1,50	0	9	9	0%	0,75	0
Simonis 10_10_1	10	5	5	0%	3,50	65	5	5	0%	2,82	76
Simonis 10_10_50	10	5	5	0%	2,85	19	5	5	0%	2,50	19
Simonis 10_20_100	10	6	6	0%	0,50	0	6	6	0%	0,87	0
Simonis Problem 10_20_150	10	9	9	0%	0,75	0	9	9	0%	0,84	0
Wilson nwrsSmaller4_1	10	3	3	0%	0,75	0	3	3	0%	1,03	0
Wilson nwrsSmaller4_2	10	4	4	0%	0,54	0	4	4	0%	1,09	0
Harvey wbo_15_15_1	15	3	3	0%	17,07	10	3	3	0%	18,76	6
Harvey wbo_15_30_1	15	4	4	0%	3,65	0	4	4	0%	13,89	0
Harvey wbop_15_30_15	15	11	11	0%	997,04	21433	11	11	0%	272,29	5540
Harvey wbp_15_15_1	15	4	4	0%	1,51	0	4	4	0%	3,76	0
Harvey wbp_15_15_35	15	14	14	0%	1,53	0	14	14	0%	1,54	0
Simonis 15_15_200	15	11	11	0%	2,29	0	11	11	0%	2,26	0
Simonis 15_15_90	15	11	10	9%	3127,75	148083	11	11	0%	3086,39	174768
Simonis 15_30_100	15	14	14	0%	0,75	0	14	14	0%	1,35	0
Simonis Problem 15_15_100	15	11	11	0%	177,17	5056	11	11	0%	272,53	7834
Wilson nwrsSmaller_4	15	7	7	0%	7,82	11	7	7	0%	2,67	0
Wilson nwrsSmaller4_3	15	7	7	0%	1,25	0	7	7	0%	4,00	0
Harvey wbo_20_10_1	20	6	5	17%	826,09	361	6	5	17%	1307,15	696
Harvey wbo_20_20_1	20	3	3	0%	20,04	0	3	3	0%	18,87	0
Harvey wbop_20_10_10	20	8	6	25%	868,25	284	8	7	13%	806,00	460
Harvey wbop_20_10_15	20	12	9	25%	1646,03	1936	12	12	0%	516,03	507
Miller	20	13	11	15%	1985,75	1871	13	9	30%	1923,10	3017
Shaw Instance_1	20	14	12	14%	1950,50	6237	14	12	14%	1951,51	6757
Shaw Instance_10	20	13	11	15%	1929,78	5051	13	10	23%	1935,01	4950
Shaw Instance_2	20	12	12	0%	1897,75	3349	12	11	8%	1981,75	3214
Shaw Instance_3	20	14	12	14%	1981,75	7344	14	12	14%	1982,50	6397
Simonis Problem 20_10_1	20	9	7	22%	1892,51	1422	9	7	22%	1941,28	1609
Simonis Problem 20_20_100	20	19	19	0%	3,50	0	19	19	0%	4,25	0
Wilson nrwsLarger_1	20	12	12	0%	13,00	0	12	12	0%	4,31	0
Wilson nrwsLarger4_2	20	12	12	0%	14,29	5	12	12	0%	5,03	0
Wilson nrwsLarger_3	25	10	10	0%	8,75	0	10	10	0%	148,60	10
Wilson nrwsLarger_4	25	16	14	13%	1413,50	720	16	14	13%	188,10	38
Wilson SP_1	25	9	8	11%	378,10	2934	9	8	11%	1979,75	49
Harvey wbo_30_15_1	30	7	6	14%	2193,63	2	7	6	14%	2191,26	1
Harvey wbo_30_30_1	30	4	3	25%	1907,06	0	4	3	25%	1934,75	0
Simonis 30_10_1	30	13	9	31%	1681,28	2	12	9	25%	1605,05	100
Simonis 30_15_100	30	27	27	0%	30,71	0	27	27	0%	46,78	3
Simonis Problem 30_30_1	30	21	13	38%	2990,03	54	24	13	45%	1927,45	0
Mean				7%	681,95	4697,55			6%	640,12	4916,66

Table 5.3: Computational results for the MOSP models

Chapter 6

Variants of the MOSP Model

Having developed a fully functional integer programming model for the minimization of open stacks problem, we now explore some variants of the model discussed in the previous Chapters.

The main idea behind the integer programming model presented is the completion of the MOSP graph with suitable fill edges, with the purpose of constructing an interval graph. There are several edge completion problems documented in Section 3.8. Here we address the Minimum Interval Graph Completion, which searches for the minimum number of fill edges that should be added to a graph to obtain an interval graph. With small changes in the objective function and using some of the previous constraints, we can build an integer programming model for this problem in Graph Theory.

Noticing that in most instances there are alternative optimal solutions for the MOSP, we tried to take the problem further and added a second step with a new objective function: the minimization of the order spread. This pattern sequencing problem similar to the MOSP was presented in section 2.1.3 and it is also related with the minimum interval graph completion problem.

There is also another pattern sequencing problem called the Minimization of Tool Switches which is addressed in this Chapter, using the similarities between this problem and the MOSP.

At the end of each section we will present some computational results and short conclusions.

6.1 An Integer Programming Model for the Minimum Interval Graph Completion Problem

Several edge completion problems have been studied in literature, consisting of, given a graph $G = (V, E)$, finding a supergraph $H = (V, E \cup F)$ with the same set of vertices of G and belonging to some predefined class of graphs $\mathscr{C}$.

The most popular of these problems is called the *minimum fill-in* problem (or minimum triangulation), where we want to obtain a chordal graph H adding the least number of edges $|F|$. As this problem is NP-hard, it is frequent to address the *minimal fill-in* problem that is a polynomially computable related problem, where it is required only that H is such that no proper subgraph is chordal.

If H is required to be an interval graph, we can define analogously the *minimum* and the *minimal interval graph completion* problems.

The minimum interval graph completion problem (IGC) has applications in Cutting Stock Industry, in Archaeology, in DNA physical mapping and in Numerical Analysis. Two other graph problems equivalent to this have been studied independently in literature: the minimum sum cut problem and the profile minimization problem.

In this section we present an integer programming formulation for solving the minimum interval graph completion problem recurring to a characterization of interval graphs that produces a linear ordering of the maximal cliques of the solution graph.

6.1.1 Introduction

The minimum interval graph completion problem and similar problems have been studied widely in terms of complexity, addressed with heuristics and approximation algorithms, and treated for special classes of input graphs. However, there have been few integer programming approaches for solving these type of problems.

In this thesis we propose an integer programming formulation to solve the minimum interval graph completion problem, based on inequalities from a known IP model for the linear ordering problem, and new inequalities inspired on a characterization of interval graphs that uses a linear ordering of the vertices.

The decision variables x_{ij} defined as

$$x_{ij} = \begin{cases} 1 & \text{if } \varphi(i) < \varphi(j) \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j \in V$$

set a linear ordering $\varphi : V \to \{1, ..., N\}$ for the vertices and the variables y_{ij} defined only for the possible fill edges $[ij] \notin E$

$$y_{ij} = \begin{cases} 1 & \text{if } [ij] \notin E \text{ and } [ij] \notin F \text{ and } \varphi(i) < \varphi(j) \\ 0 & \text{if } [ij] \notin E \text{ and } [ij] \in F \text{ or } \varphi(i) \geq \varphi(j) \end{cases}$$

decide which intervals will overlap in the desired interval graph H.

We will not need the variables z_{ij} because the number of stacks is irrelevant in the minimum interval graph completion problem. The objective is simply completing the graph with the smallest number of edges to obtain an interval graph.

The sum of all variables y gives the number of edges that are not added to the graph G when completing it to an interval graph H. By maximizing this sum, we get a minimum number of added edges.

The characterization of interval graphs by Olariu and the linear ordering inequalities will be the main part of the model. We can use just the basic inequalities from section (4.7), or we can benefit from the improvements made to the model, such as the reduced number of variables presented in section (4.8) and the additional inequalities that we derived in section (5.2), namely the ones that originated from properties of interval graphs.

6.1.2 The Formulation of the Model

Given a graph $G = (V; E)$ with $|V| = N$ and $|E| = M$, we compose the following new mathematical formulation for the minimum interval graph completion problem using the original variables x and y and the additional inequalities that strengthen the model.

$$\text{Maximize} \quad \sum_{[ij]\notin E} y_{ij}$$

$$\begin{array}{rlr}
\text{Subject to:} \quad x_{ij} + x_{ji} = 1 & \forall i,j \in V & (6.1) \\
x_{ij} + x_{jk} + x_{ki} \leq 2 & \forall i,j,k \in V & (6.2) \\
y_{ij} \leq x_{ij} & \forall i,j \in V \text{ with } [ij] \notin E & (6.3) \\
y_{ij} \leq x_{kj} & \forall i,j,k \in V \text{ with } [ij] \notin E,\ [ik] \in E & (6.4) \\
y_{ij} - y_{ik} \leq x_{kj} & \forall i,j,k \in V \text{ with } [ij],[ik] \notin E & (6.5) \\
y_{ij} + y_{ki} \leq 1 & \forall i,j,k \in V \text{ with } [ij],[ik] \notin E, [jk] \in E & (6.6) \\
y_{ij} + y_{jk} \leq 1 & \forall i,j,k \in V \text{ with } [ij],[jk] \notin E, [ik] \in E & (6.7) \\
y_{ij} + y_{lk} \leq 1 & \forall i,j,k,l \in V \text{ with } [ij],[kl] \notin E, [jl],[ik] \in E & (6.8) \\
y_{ij} + y_{jk} - 1 \leq y_{ik} & \forall i,j,k \in V \text{ with } [ij],[jk],[ik] \notin E & (6.9) \\
y_{ik} + y_{ki} + y_{jl} + y_{lj} \leq 1 & \forall i,j,k,l \in V \text{ with } [ik],[jl] \notin E, [ij],[jk],[kl],[li] \in E & (6.10) \\
y_{il} + y_{li} + y_{ik} + y_{ki} + y_{jl} + y_{lj} + & \forall i,j,k,l,m \in V \text{ with } [ij],[jk],[kl],[lm],[mi] \in E & \\
+ y_{jm} + y_{mj} + y_{mk} + y_{km} \quad \leq 3 & \text{and } [ik],[il],[jl],[jm],[km] \notin E & (6.11) \\
x_{ij} \in \{0,1\} & \forall i,j \in V \text{ with } i \neq j & (6.12) \\
y_{ij} \in \{0,1\} & \forall i,j \in V \text{ with } [ij] \notin E & (6.13)
\end{array}$$

We use the linear ordering polytope inequality (6.1) to state that either $\varphi(i) < \varphi(j)$ or $\varphi(j) < \varphi(i)$, and (6.2) to force the transitivity of the linear ordering:

$$\varphi(i) < \varphi(j) \wedge \varphi(j) < \varphi(k) \Rightarrow \varphi(i) < \varphi(k).$$

The logical implication between variables y and x is stated by (6.3). Olariu's characterization of interval graphs is in (6.4) and (6.5), strengthened by (6.6)-(6.8). Inequality (6.9) guarantees that the complement graph of H is a comparability graph and inequalities (6.10) and (6.11) add the necessary number of chords to chordless cycles of sizes 4 and 5.

This model has $N(N-1)$ binary variables x_{ij} and $N(N-1)-2M$ binary variables y_{ij} (because for each one of the M edges $[ij] \in E$ the corresponding variables y_{ij} and y_{ji} do not exist), therefore the total number of variables in the model is $2N(N-1) - 2M$.

If we choose to use the reduced number of variables and the correspondent inequalities, we would have to change the first five inequalities in this model:

$$\begin{array}{lll}
\text{Maximize} & \displaystyle\sum_{[ij]\notin E} y_{ij} & \\
\text{Subject to:} & (5.18) \text{ to } (5.24) & \\
 & (6.6) \text{ to } (6.11) & \\
 & x_{ij} \in \{0,1\} & \forall i,j = 1,...,N \text{ with } i < j \\
 & y_{ij} \in \{0,1\} & \forall i,j = 1,...,N \text{ with } i \neq j, [ij] \notin E
\end{array}$$

In this case, the total number of variables is $3N(N-1)/2 - 2M$.

6.1.3 Computational Results for IGC

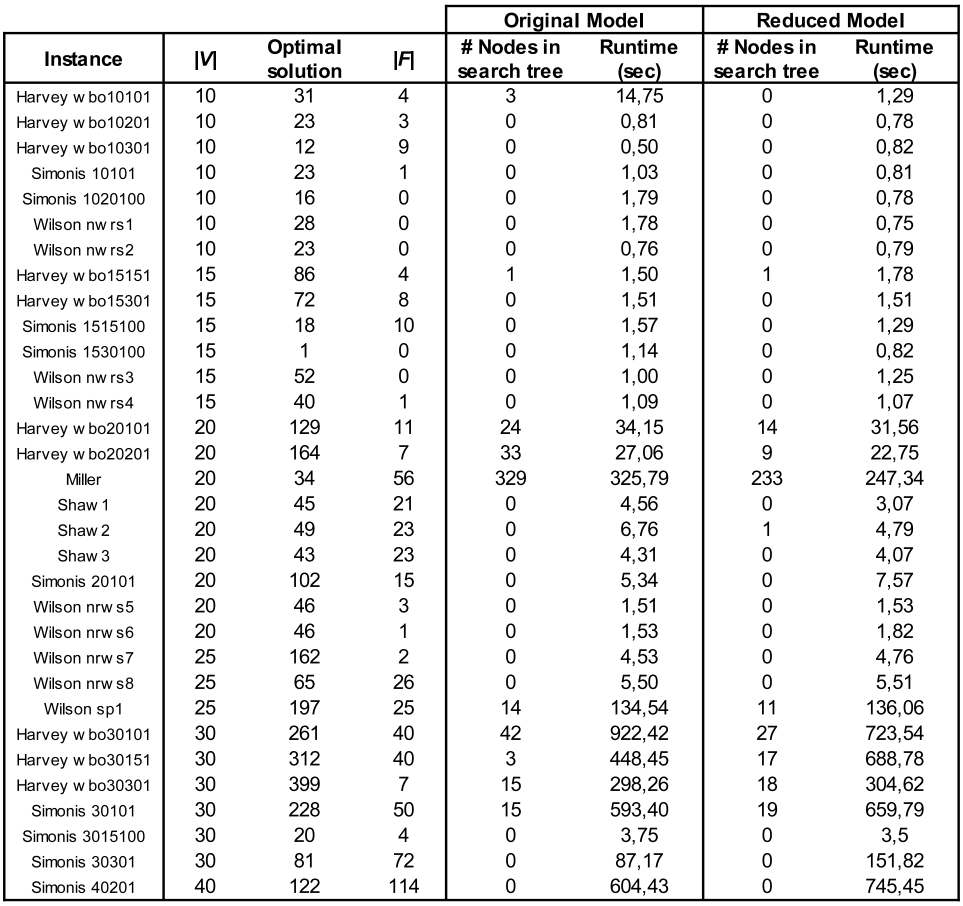
The model was tested on graphs appropriately built from the instances of the Constraint Modeling Challenge 2005, available at

`http://www.cs.st-andrews.ac.uk/∼ipg/challenge/instances.html`

Computational tests were performed with ILOG OPL Development Studio 5.5 on an Intel®Core2 Duo T7200@2.00GHz 0.99GB RAM. We tested each instance with both the original model and the reduced version of the model: the optimal solution, the number of nodes of the search tree and the runtime are recorded in Table 6.1.

As the model maximizes the sum of all variables y, which is the number of arcs that do not exist in H, the number of fill edges $|F|$ in the solution, that is the minimum number of edges for an interval completion H of G, was also registered. Clearly, the sum of the optimal solution with $|E|$ and $|F|$ is equal to the number of edges in a complete graph.

In all of the instances tested, the optimal solution was reached and proved optimal, using both of the models presented in this paper. In small and medium instances the optimal solution was found in just a few seconds or minutes. Unfortunately, for larger instances there was not enough memory to handle the model. Comparing the results of the original formulation with the reduced model, we can see that the runtime and the number of nodes in search tree are similar for most of the instances. In some instances the reduced model was able to decrease the runtime, or the number of nodes. This did not happen with all of the instances, because in IP it is really more important to strengthen the lower bound from the linear programming relaxation rather than to reduce the number of variables.



				Original Model		Reduced Model	
Instance	|V|	Optimal solution	|F|	# Nodes in search tree	Runtime (sec)	# Nodes in search tree	Runtime (sec)
Harvey w bo10101	10	31	4	3	14,75	0	1,29
Harvey w bo10201	10	23	3	0	0,81	0	0,78
Harvey w bo10301	10	12	9	0	0,50	0	0,82
Simonis 10101	10	23	1	0	1,03	0	0,81
Simonis 1020100	10	16	0	0	1,79	0	0,78
Wilson nw rs1	10	28	0	0	1,78	0	0,75
Wilson nw rs2	10	23	0	0	0,76	0	0,79
Harvey w bo15151	15	86	4	1	1,50	1	1,78
Harvey w bo15301	15	72	8	0	1,51	0	1,51
Simonis 1515100	15	18	10	0	1,57	0	1,29
Simonis 1530100	15	1	0	0	1,14	0	0,82
Wilson nw rs3	15	52	0	0	1,00	0	1,25
Wilson nw rs4	15	40	1	0	1,09	0	1,07
Harvey w bo20101	20	129	11	24	34,15	14	31,56
Harvey w bo20201	20	164	7	33	27,06	9	22,75
Miller	20	34	56	329	325,79	233	247,34
Shaw 1	20	45	21	0	4,56	0	3,07
Shaw 2	20	49	23	0	6,76	1	4,79
Shaw 3	20	43	23	0	4,31	0	4,07
Simonis 20101	20	102	15	0	5,34	0	7,57
Wilson nrw s5	20	46	3	0	1,51	0	1,53
Wilson nrw s6	20	46	1	0	1,53	0	1,82
Wilson nrw s7	25	162	2	0	4,53	0	4,76
Wilson nrw s8	25	65	26	0	5,50	0	5,51
Wilson sp1	25	197	25	14	134,54	11	136,06
Harvey w bo30101	30	261	40	42	922,42	27	723,54
Harvey w bo30151	30	312	40	3	448,45	17	688,78
Harvey w bo30301	30	399	7	15	298,26	18	304,62
Simonis 30101	30	228	50	15	593,40	19	659,79
Simonis 3015100	30	20	4	0	3,75	0	3,5
Simonis 30301	30	81	72	0	87,17	0	151,82
Simonis 40201	40	122	114	0	604,43	0	745,45

Table 6.1: Computational results of the IP model for the minimum interval graph completion problem

6.2 Minimization of the Stack Occupation

The model we have developed for the minimization of open stacks can be used in another pattern sequencing problem, where the objective is to find an optimal sequence to process the cutting patterns in order to minimize the occupation of the stacks.

6.2.1 Introduction

The problem we address here is similar to minimizing the flow time of the orders: besides having the minimum number of open stacks, we also want to minimize the sum of the time that the stacks remain open within the system.

The sequence in which preset cutting patterns are processed can affect the flow and total completion time, so it is desirable to optimize the occupation of the stacks to eliminate unnecessary dispersion.

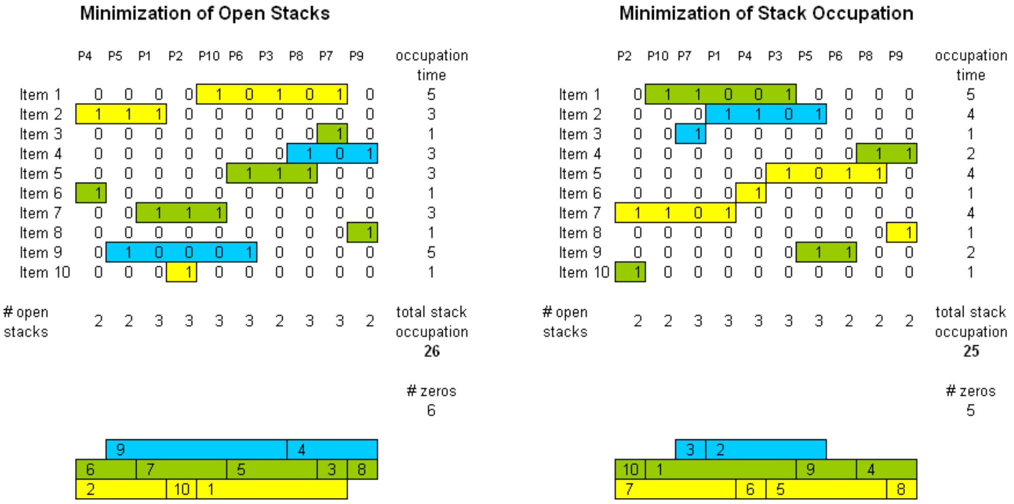
A solution can also be modeled by an interval graph exhibiting a set of intervals that match the duration of stacks. Hence we can benefit from the IP model discussed in the previous chapters and, with small modifications, use it to reduce the occupation of the stacks. This goal will be achieved by adding the least number of edges to the interval graph.

Recall the minimization of order spread problem discussed in section 2.1.3. Most of the authors define the MORP as the minimization of the average stack spread, which is the time between the opening and the closing of each stack, measured in number of patterns.

The minimization of the stack occupation comprehends both the objectives from the minimization of the order spread and the minimization of the number of open stacks.

When considering the MOSP, it is usual to find more than one optimal solution, in the sense that there is more than one sequence of the cutting patterns that achieves the same maximum number of open stacks. We may be interested in choosing between these optimal solutions of the MOSP according to a different criterion. A natural choice is the minimization of the order spread.

In Figure 6.1 we can see two optimal solutions concerning the minimization of open stacks problem that have different total stack occupation.



Minimization of Open Stacks

	P4	P5	P1	P2	P10	P6	P3	P8	P7	P9	occupation time
Item 1	0	0	0	0	1	0	1	0	1	0	5
Item 2	1	1	1	0	0	0	0	0	0	0	3
Item 3	0	0	0	0	0	0	0	0	1	0	1
Item 4	0	0	0	0	0	0	0	1	0	1	3
Item 5	0	0	0	0	0	1	1	1	0	0	3
Item 6	1	0	0	0	0	0	0	0	0	0	1
Item 7	0	0	1	1	1	0	0	0	0	0	3
Item 8	0	0	0	0	0	0	0	0	0	1	1
Item 9	0	1	0	0	0	1	0	0	0	0	5
Item 10	0	0	0	1	0	0	0	0	0	0	1
# open stacks	2	2	3	3	3	3	2	3	3	2	total stack occupation **26**

zeros
6

9
4
6
7
5
3
8
2
10
1

Minimization of Stack Occupation

	P2	P10	P7	P1	P4	P3	P5	P6	P8	P9	occupation time
Item 1	0	1	1	0	0	1	0	0	0	0	5
Item 2	0	0	0	1	1	0	1	0	0	0	4
Item 3	0	0	1	0	0	0	0	0	0	0	1
Item 4	0	0	0	0	0	0	0	0	1	1	2
Item 5	0	0	0	0	0	1	0	1	1	0	4
Item 6	0	0	0	0	1	0	0	0	0	0	1
Item 7	1	1	0	1	0	0	0	0	0	0	4
Item 8	0	0	0	0	0	0	0	0	0	1	1
Item 9	0	0	0	0	0	0	1	1	0	0	2
Item 10	1	0	0	0	0	0	0	0	0	0	1
# open stacks	2	2	3	3	3	3	3	2	2	2	total stack occupation **25**

zeros
5

Figure 6.1: Number of open stacks and total stack occupation for the instance wbo10101

Our model consists in finding out which arcs should be added to the original MOSP graph $G = (V, E)$ in order to get an interval graph $H = (V, E \cup F)$ that minimizes the stack occupation while keeping the minimum number of simultaneously open stacks.

The model we present is divided in two steps. In a first step, the minimum number of open stacks is determined, and then in a second step, we search for a new sequence of the patterns that improves the total stack spread while using the optimal number of open stacks.

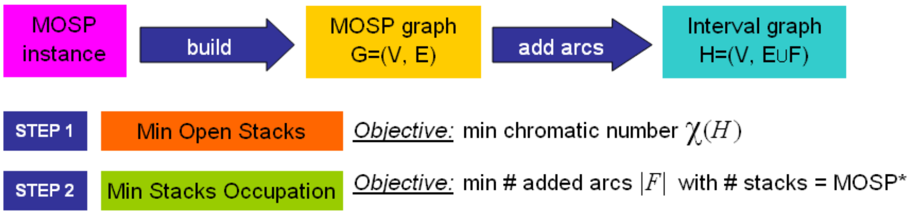

MOSP instance
build
MOSP graph G=(V, E)
add arcs
Interval graph H=(V, E∪F)
STEP 1
Min Open Stacks
Objective: min chromatic number $\chi(H)$
STEP 2
Min Stacks Occupation
Objective: min # added arcs $|F|$ with # stacks = MOSP*

Figure 6.2: A two step model to minimize the occupation of stacks

6.2.2 The formulation of the model

In the first step the formulation is the same as in section 5.3, with the objective to minimize the maximum number of open stacks.

$$\begin{array}{lll} \text{STEP1: Minimize} & & K \\ & \text{Subject to:} & (5.18) \text{ to } (5.45) \end{array}$$

Let us denote the optimal number of open stacks found by $MOSP^*$. Then, in the second step, the objective becomes the minimization of the stack spread.

To minimize the average order spread is equivalent to minimizing the total stack spread. This is also equivalent to minimizing the number of fill-in zeros obtained in the matrix of the description of the cutting patterns after the columns have been rearranged to match the sequence in which the patterns will be processed.

This is done by minimizing the number of arcs that are added to the MOSP graph in order to obtain an interval graph. As the variables y_{ij} are 1 when an arc is not added to the graph, we can minimize the number of added arcs by maximizing the sum of the variables y_{ij}. Therefore the objective function in step 2 is:

$$\max \sum_{[ij]\notin E} y_{ij} \tag{6.14}$$

To guarantee that the optimal number of open stacks does not increase from step 1 to step 2, some of the inequalities have to be modified. The inequality that carried the main lower bound for the MOSP in step 1, (5.25), must now force the number of open stacks at each instant to be less than or equal to the optimal number of open stacks found in step 1, $MOSP^*$.

$$\sum_{\substack{i=1\\i\neq j}}^{N} x_{ij} - \sum_{\substack{i=1\\ [ij]\notin E}}^{N} y_{ij} \ + 1 \leq MOSP^* \qquad \forall j \in V \tag{6.15}$$

The number of colors should also be the optimal number of stacks found previously in step 1, thus the inequality that counts the number of colors (5.32) must be altered to:

$$\sum_{i=1}^{N} s_{ii} \ = MOSP^* \tag{6.16}$$

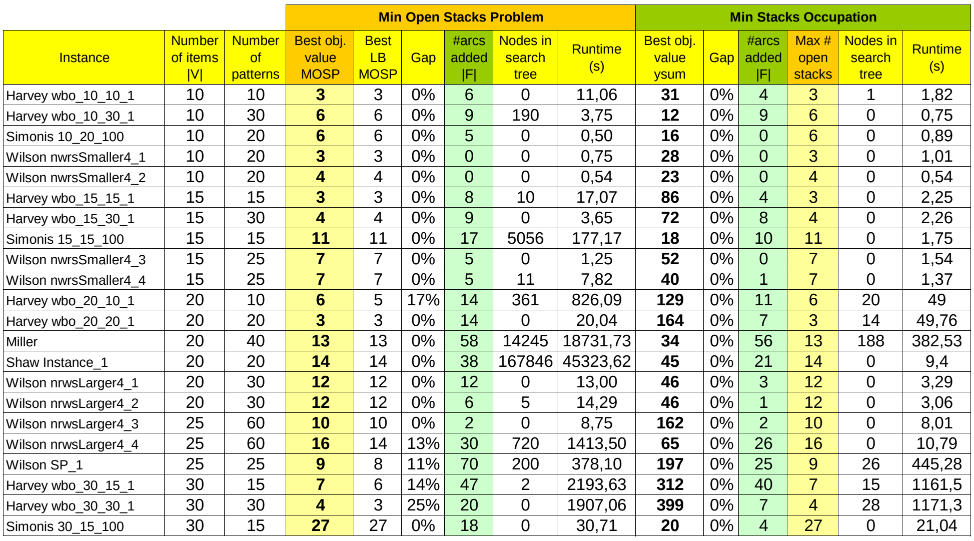
Statement (5.45) is removed and the remaining inequalities in the model are maintained.

$$\begin{array}{rl} \text{STEP2: Maximize} & \displaystyle\sum_{[ij]\notin E} y_{ij} \\ \text{Subject to:} & (5.18) \text{ to } (5.24) \\ & (6.15) \\ & (5.26) \text{ to } (5.31) \\ & (6.16) \\ & (5.33) \text{ to } (5.44) \end{array}$$

6.2.3 Computational results for the minimization of the occupation of the stacks

The model for the minimization of the occupation of the stacks was tested on the instances of the Constraint Modeling Challenge and the results are presented in Table 6.2.

In the second step we were able to obtain the optimal solution in every instances tested. This second step allowed to reduce the order spread in almost every instance, while maintaining the same optimal number of open



			Min Open Stacks Problem						Min Stacks Occupation					
Instance	Number of items |V|	Number of patterns	Best obj. value MOSP	Best LB MOSP	Gap	#arcs added |F|	Nodes in search tree	Runtime (s)	Best obj. value ysum	Gap	#arcs added |F|	Max # open stacks	Nodes in search tree	Runtime (s)
Harvey wbo_10_10_1	10	10	**3**	3	0%	6	0	11,06	**31**	0%	4	3	1	1,82
Harvey wbo_10_30_1	10	30	**6**	6	0%	9	190	3,75	**12**	0%	9	6	0	0,75
Simonis 10_20_100	10	20	**6**	6	0%	5	0	0,50	**16**	0%	0	6	0	0,89
Wilson nwrsSmaller4_1	10	20	**3**	3	0%	0	0	0,75	**28**	0%	0	3	0	1,01
Wilson nwrsSmaller4_2	10	20	**4**	4	0%	0	0	0,54	**23**	0%	0	4	0	0,54
Harvey wbo_15_15_1	15	15	**3**	3	0%	8	10	17,07	**86**	0%	4	3	0	2,25
Harvey wbo_15_30_1	15	30	**4**	4	0%	9	0	3,65	**72**	0%	8	4	0	2,26
Simonis 15_15_100	15	15	**11**	11	0%	17	5056	177,17	**18**	0%	10	11	0	1,75
Wilson nwrsSmaller4_3	15	25	**7**	7	0%	5	0	1,25	**52**	0%	0	7	0	1,54
Wilson nwrsSmaller4_4	15	25	**7**	7	0%	5	11	7,82	**40**	0%	1	7	0	1,37
Harvey wbo_20_10_1	20	10	**6**	5	17%	14	361	826,09	**129**	0%	11	6	20	49
Harvey wbo_20_20_1	20	20	**3**	3	0%	14	0	20,04	**164**	0%	7	3	14	49,76
Miller	20	40	**13**	13	0%	58	14245	18731,73	**34**	0%	56	13	188	382,53
Shaw Instance_1	20	20	**14**	14	0%	38	167846	45323,62	**45**	0%	21	14	0	9,4
Wilson nrwsLarger4_1	20	30	**12**	12	0%	12	0	13,00	**46**	0%	3	12	0	3,29
Wilson nrwsLarger4_2	20	30	**12**	12	0%	6	5	14,29	**46**	0%	1	12	0	3,06
Wilson nrwsLarger4_3	25	60	**10**	10	0%	2	0	8,75	**162**	0%	2	10	0	8,01
Wilson nrwsLarger4_4	25	60	**16**	14	13%	30	720	1413,50	**65**	0%	26	16	0	10,79
Wilson SP_1	25	25	**9**	8	11%	70	200	378,10	**197**	0%	25	9	26	445,28
Harvey wbo_30_15_1	30	15	**7**	6	14%	47	2	2193,63	**312**	0%	40	7	15	1161,5
Harvey wbo_30_30_1	30	30	**4**	3	25%	20	0	1907,06	**399**	0%	7	4	28	1171,3
Simonis 30_15_100	30	15	**27**	27	0%	18	0	30,71	**20**	0%	4	27	0	21,04

Table 6.2: Results for the two step model for minimizing the stack occupation

stacks. This reduction was very significant in many cases, decreasing around 75% of the number of added edges.

6.3 Minimization of Tool Switches

The model developed in this thesis for the MOSP can be compared with the models from Tang and Denardo [57] and Laporte and Semet [41] that were originally developed for the MTSP. In [41] it is mentioned that the model of Laporte seriously improved the lower bound of Tang's model on a particular instance. We adapted our model to the MTSP to compare the value of the linear relaxation of our model with the values described in that paper.

Recall that there are similarities between the minimization of open stacks and the minimization of tool switches. The jobs can act as cutting patterns that need to be sequenced and the tools can act as piece types. The intervals in the solution will correspond to the intervals of time during each tool is loaded in the machine.

It was discussed previously in Section 2.1.6 that the MOSP and the MTSP are not equivalent problems, unless the capacity of the machine is equal to the optimal solution of the MOSP. So, to adapt our model to be able to solve the MTSP we have to change the objective function and add a constraint to say that the maximum number of stacks is equal to the capacity of the machine C.

$$K = C \tag{6.17}$$

A tool switch occurs each time that an interval i ends and another interval j starts, while i and j share the same stack. This corresponds to two vertices in the graph with the same color and the first vertex i being a representative of the second vertex j, if the first interval i is the leader of the stack. If it is not, then there is a third vertex k that is the leader of that stack and that represents vertex j. Actually, in that case there are two tool switches from k to i and from i to j. The first tool switch is defined by i being represented by k, and the second tool switch is declared by j being represented by k. Each tool switch is counted by that vertex being represented by a different vertex, thus the number of tool switches can be counted by the number of pairs of representative-represented vertices.

In our model this is captured by the variables z_{ij} that say when a vertex represents the color of another. The sum of all variables z_{ij} such that $i \neq j$ counts the number of tool switches. The objective function will then be:

$$\min \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} z_{ij} \tag{6.18}$$

However, our model will have a limited use in this problem. If the capacity of the tool magazine is smaller than the minimum number of open stacks, the MOSP can only be used to determine a lower bound for the number of tool switches:

Proposition 6.3.1. *[63] Let M be the number of tools, C the capacity of the tool magazine, v the minimum number of tool switches and C^* the optimal solution value of the corresponding MOSP. If $C < M$ then $LB_4 := M - C$ is a lower bound on the number of tool switches. The optimal value of the MTSP is strictly greater than the lower bound LB_4 when $C < C^*$, that is, $v > LB_4$ if $C < C^*$.*

Since $C*$ is the minimum number of stacks produced by any sequence of the jobs, if we have less tool slots, $C < C^*$, at least one of the stacks must be interrupted to make one tool switch and resumed again later when that tool is brought back to the tool magazine.

This is what happens in the following example.

The instance addressed in [57] has 10 jobs and 9 tools (which corresponds to an instance of the MOSP with 10 patterns and 9 items) but, due to the dominance of patterns, the instance can be reduced to six patterns as presented in Table 6.3.

Jobs	J1	J2	J3	J4	J5	J6
Tool 1	1	1	0	1	0	1
Tool 2	0	0	1	0	0	1
Tool 3	0	1	0	0	1	0
Tool 4	1	0	0	0	0	1
Tool 5	0	1	0	1	1	0
Tool 6	0	0	1	0	0	0
Tool 7	0	0	1	1	0	0
Tool 8	1	0	1	0	1	0
Tool 9	1	0	0	1	0	0

Table 6.3: Instance of the MTSP with 6 jobs and 9 tools

The optimal solution of this instance is 7. The value of the linear relaxation in Tang and Denardo's model is zero. Laporte and Semet found a value of the linear relaxation equal to 1.9 by using their first model. With additional constraints they were able to increase it to 2.91, and using a lifted objective function they obtained a value of 6.0 for the linear relaxation.

With this instance, the MOSP has a minimum value of 5 simultaneous open stacks, but the capacity of the machine is $C = 4$; thus a solution of the MTSP with only 4 open stacks is infeasible in our model, because constraint 6.17 would always be violated.

In fact, the MOSP graph has a clique of size 5, which is formed by tools $\{1, 5, 7, 8, 9\}$, (in a lighter color in Figure 6.3), hence it is impossible to find a sequence with only 4 open stacks.

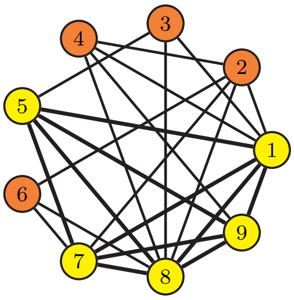


Figure 6.3: The MOSP graph of Tang and Denardo's instance

In this case, our model can only be used to find a lower bound for the minimum number of tool switches. It found that the optimal value of the MOSP problem, $C^* = 5$, is less than the capacity of the tool magazine, $C = 4$; thus by Proposition 6.3.1 we have $LB_4 = M - C = 9 - 4 = 5$ and, because $C < C^*$, it must be $v > LB_4 = 5$, and therefore we have $v \geq 6$. The lower bound obtained in this case for the number of tool switches is 6, which is the same lower bound obtained by Laporte et Semet.

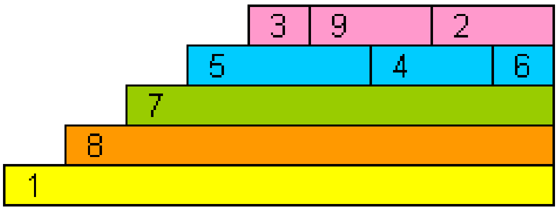


Figure 6.4: Solution of the MOSP for Tang and Denardo's instance

6.4 Conclusions

In this chapter we have explored the model developed in the previous chapters in using it to similar but different problems. With slight changes in the formulation and in the objective function, the model could be run for the Minimum Interval Graph Completion Problem and for the Minimization of the Stacks Occupation, and could be used to find lower bounds for the Minimization of the Tool Switches Problem.

Chapter 7

Conclusions and Future Work

The main integer programming models available in the literature concerning pattern sequencing in cutting stock problems were studied and compared. The focus was set on the integer programming models, but we also referred to other approaches using heuristics, genetic algorithms and dynamic programming. This research evidenced some relationships between the minimization of open stacks problem and graph layout problems. It led to the study of the graph theory related with perfect graphs and linear ordering models that could be used to derive a model for the minimization of open stacks problem.

A new integer programming formulation was developed for the MOSP, based on the edge completion of a MOSP graph and on a characterization of interval graphs that uses a perfect elimination ordering of the vertices. It was proved that most of the basic inequalities of the model are facets of the polytope. The model was strengthened with more inequalities based on properties of interval graphs. The computational results obtained with ILOG OPL Studio software for these alternative formulations of the model were discussed. Finally, some variations of the model and applications to other problems in pattern sequencing and in graph layout were explored.

As further work, there is still room for improvements in the model. One of the topics that should be addressed is the reduction of symmetry in the sequence of the patterns, which is a crucial factor for improving the runtime. Even in instances with a small number of vertices, the existence of symmetry makes the finding of the optimal solution and the proof of optimality more difficult and time consuming. We have experienced that issue with the Miller instance in Figure 7.1. One way to reduce symmetry is to choose an opening order for the neighbors of the first vertex to close and to include that in the model, no matter what vertex closes first. Clearly, all the neighbors of the first vertex to close must open beforehand, and, for instances with a large value of the MOSP, there are many permutations of the opening order.

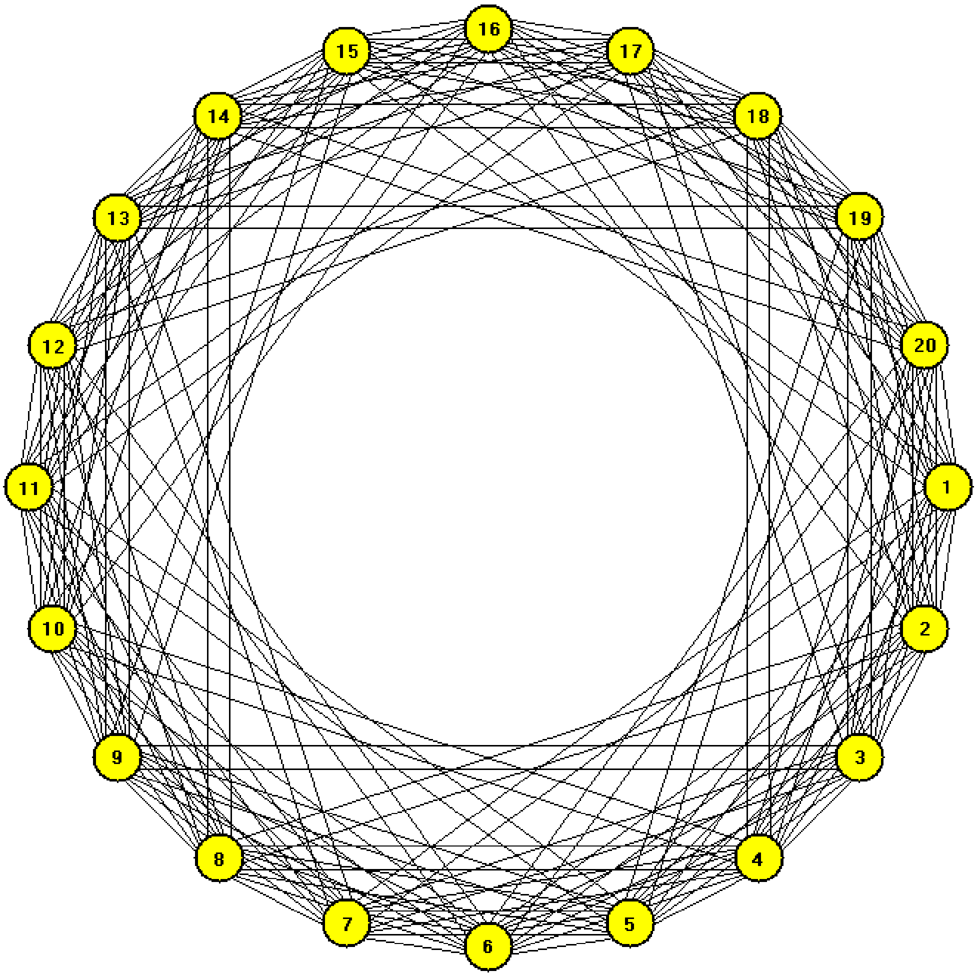


Figure 7.1: MOSP graph of the Miller instance

This work focused mainly on the formulation of a good integer programming model. The tests to validate the model were made only with standard commercial software. A natural direction from here is to derive stronger models applying decomposition techniques for solving integer programming problems, such as column generation with Dantzig-Wolfe decomposition, or Lagrangian relaxation.

We did a polyhedral analysis of the main constraints and showed them to be facets. However, there are more valid inequalities as, for instance, the 4-cycle and 5-cycle chordal constraints discussed in section 5.2.3 and the neighbor of successor inequalities presented in section 5.2.1, that should be further analyzed to check if they are facets or dominated faces.

The pattern sequencing problems addressed in this work are treated using the usual academic conventions of this type of problems, in order to obtain an all-purpose model that may be applied in many industry settings. In the real industry there are often many other important limitations, such as the saw time of the cutting process, or the existence of a maximum number of items that can be piled in a stack. Those constraints were not considered here and may have impact on the desired solution; thus it would be interesting to adapt this model to a real case study, with all the implications it would bring.

A future objective is the development of the main model presented in this thesis to tackle the cutting stock problem and the pattern sequencing problem in one integrated procedure, similarly to what has been done by Pinto and Yanasse in [65], or Pilleggi, Morabito and Arenales in [53].

Other possibilities of further work are to explore graph theory problems similar to the minimum interval graph completion, namely the minimum fill-in problem and the colored interval sandwich problem.

Bibliography

[1] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987. [cited at p. 58]

[2] F. Ashikaga and N. Soma. A heuristic for the minimization of open stacks problem. *Pesquisa Operacional*, 29(2):439 – 450, 2009. [cited at p. 34, 35]

[3] M. G. Banda and P. J. Stuckey. Dynamic programming to minimize the maximum number of open stacks. *Informs Journal On Computing*, 19:607–617, 2007. [cited at p. 35]

[4] P. Baptiste. Simple mip formulations to minimize the maximum number of open stacks. In *Constraint Modelling Challenge*, pages 9–13, Edinburgh, Scotland, July 31 2005. IJCAI 2005. [cited at p. 24, 35]

[5] J. F. Bard. A heuristic for minimizing the number of tool switches on a flexible machine. *IIE Transactions*, 20(4):382–391, December 1988. [cited at p. 13, 16]

[6] J. C. Becceneri, H. H. Yanasse, and N. Y. Soma. A method for solving the minimization of the maximum number of open stacks problem within a cutting process. *Computers & Operations Research*, 31(14):2315–2332, 2004. [cited at p. 29, 32]

[7] T. Biedl. *CS 762: Graph-theoretic algorithms – Lecture notes of a graduate course.* University of Waterloo, September 2005. [cited at p. 43, 49, 50, 52]

[8] A. Billionnet. On interval graphs and matrice profiles. *RAIRO. Recherche opérationnelle*, 20(3):245–256, 1986. [cited at p. 48, 63]

[9] H. L. Bodlaender and A. M. Koster. Treewidth computations I. Upper bounds. *Information and Computation*, 208(3):259 – 275, 2010. [cited at p. 58, 61]

[10] H. L. Bodlaender and A. M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008. [cited at p. 58]

[11] H. Cambazard and N. Jussien. Techniques rétrospectives pour résoudre le minimum open stacks problem. In *Actes JFPC 2006*, pages 89–98, 2006. [cited at p. 35]

[12] M. Campêlo, V. A. Campos, and R. C. Corrêa. On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics*, 156(7):1097 – 1111, 2008. GRACO 2005 - 2nd Brazilian Symposium on Graphs, Algorithms and Combinatorics. [cited at p. 111]

[13] G. Chu and P. Stuckey. Minimizing the maximum number of open stacks by customer search. In I. Gent, editor, *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*, volume 5732 of *LNCS*, pages 242–257. Springer-Verlag, 2009. [cited at p. 35]

[14] M. Chudnovsky, N. Robertson, P. D. Seymour, and R. Thomas. Progress on perfect graphs. *Mathematical Programming*, 97(1):405–422, July 2003. [cited at p. 52]

[15] D. G. Corneil, S. Olariu, and L. Stewart. The ultimate interval graph recognition algorithm? (extended abstract). In *Symposium on Discrete Algorithms*, pages 175–180, 1998. [cited at p. 47, 48]

[16] Y. Crama. Combinatorial optimization models for production scheduling in automated manufacturing systems. *European Journal of Operational Research*, 99(1):136 – 153, 1997. [cited at p. 13]

[17] J. Díaz, A. Gibbons, M. Paterson, and J. Toran. The minsumcut problem. In F. Dehne, J. Sack, and N. Santoro, editors, *Algorithms and Data structures*, volume 519 of *Lecture Notes in Computer Science*, pages 65–79, 1991. 2nd Workshop on algorithms and data structures (WADS91), Ottawa, Canada, Aug 14-16, 1991. [cited at p. 62]

[18] J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Computing Surveys*, 34(3):313–356, September 2002. [cited at p. 54, 56, 66, 67, 103]

[19] R. G. Dyson and A. S. Gregory. The cutting stock problem in the flat glass industry. *Operational Research Quarterly*, 25(1):41–53, Mar 1974. [cited at p. 6, 30]

[20] E. Faggioli and C. A. Bentivoglio. Heuristic and exact methods for the cutting sequencing problem. *European Journal of Operational Research*, 110:564–575, 1998. [cited at p. 1, 35]

[21] A. Fink and S. Voss. Applications of modern heuristic search methods to pattern sequencing problems. *Computers & Operations Research*, 26(1):17–34, 1999. [cited at p. 12, 34]

[22] S. Fiorini. 0, 1/2-cuts and the linear ordering problem: Surfaces that define facets. *SIAM Journal on Discrete Mathematics*, 20(4):893–912, 2006. [cited at p. 59, 77]

[23] H. Foerster and G. Wascher. Simulated annealing for order spread minimization in sequencing cutting patterns. *European Journal of Operational Research*, 110(2):272–281, OCT 16 1998. [cited at p. 11, 15, 34]

[24] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15:835–855, 1965. [cited at p. 44, 48, 65]

[25] M. Garey, R. Graham, D. Johnson, and D. Knuth. Complexity results for bandwidth minimization. *SIAM Journal on Applied Mathematics*, 34(3):477–495, 1978. [cited at p. 12]

[26] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979. [cited at p. 12, 41]

[27] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961. [cited at p. 1]

[28] M. C. Golumbic. *Algorithmic graph theory and perfect graphs*. Academic Press, New York, 1980. [cited at p. 43, 44, 45, 46, 47, 52]

[29] M. C. Golumbic, H. Kaplan, and R. Shamir. On the complexity of DNA physical mapping. *Advances in Applied Mathematics*, 1994. [cited at p. 63, 64]

[30] M. Grötschel, M. Jünger, and G. Reinelt. Facets of the linear ordering polytope. *Mathematical Programming*, 33(1):43–60, Sept. 1985. [cited at p. 59, 60, 93]

[31] P. Heggernes. Minimal triangulations of graphs: A survey. *Discrete Mathematics*, 306(3):297 – 317, 2006. [cited at p. 62]

[32] P. Heggernes and F. Mancini. Minimal split completions. *Discrete Applied Mathematics*, 157(12):2659 – 2669, jun 2009. Second Workshop on Graph Classes, Optimization, and Width Parameters. [cited at p. 61]

[33] P. Heggernes, F. Mancini, and C. Papadopoulos. Minimal comparability completions of arbitrary graphs. *Discrete Applied Mathematics*, 156(5):705 – 718, 2008. [cited at p. 61]

[34] H. Kaplan, R. Shamir, and R. Tarjan. Tractability of parameterized completion problems on chordal and interval graphs: minimum fill-in and physical mapping. In *35th Annual Symposium on Foundations of Computer Science Proceedings*, pages 780–791, Nov 1994. [cited at p. 64]

[35] H. Kaplan, R. Shamir, and R. E. Tarjan. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *Siam Journal On Computing*, 28(5):1906–1922, May 1999. [cited at p. 61, 62]

[36] D. Kendall. Incidence matrices, interval graphs and seriation in archeology. *Pac. J. Math.*, 28:565–570, 1969. [cited at p. 63]

[37] D. G. Kendall. Some problems and methods in statistical archaeology. *World Archaeology*, 1(1):68–76, Jun. 1969. [cited at p. 65]

[38] D. G. Kendall. Abundance matrices and seriation in archaeology. *Probability Theory and Related Fields*, 17(2):104–112, June 1971. [cited at p. 65]

[39] T. Kloks, D. Kratsch, and J. Spinrad. On treewidth and minimum fill-in of asteroidal triple-free graphs. *Theoretical Computer Science*, 175(2):309 – 335, 1997. [cited at p. 61]

[40] Y.-L. Lai and K. Williams. A survey of solved problems and applications on bandwidth, edgesum, and profile of graphs. *Journal of Graph Theory*, 31(2):75–94, 1999. [cited at p. 55, 62, 63]

[41] G. Laporte, J. J. S. González, and F. Semet. Exact algorithms for the job sequencing and tool switching problem. *IIE Transactions*, 36:37–45, 2004. [cited at p. 19, 21, 22, 35, 133]

[42] C. Lekkerkerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51:45–64, 1962. [cited at p. 46]

[43] Y. Lin and J. Yuan. Profile minimization problem for matrices and graphs. *Acta Mathematicae Applicatae Sinica (English Series)*, 10(1):107–112, Jan 1994. [cited at p. 62]

[44] A. Linhares and H. H. Yanasse. Connections between cutting-pattern sequencing, VLSI design, and flexible machines. *Computers & Operations Research*, 29(12):1759–1772, 2002. [cited at p. 12, 13, 29, 35, 63, 67, 70]

[45] O. B. Madsen. An application of travelling-salesman routines to solve pattern-allocation problems in the glass industry. *The Journal of the Operational Research Society*, 39(3):249–256, March 1988. [cited at p. 11, 14]

[46] R. Morabito and L. Belluzzo. Optimising the cutting of wood fibre plates in the hardboard industry. *European Journal of Operational Research*, 183(3):1405 – 1420, 2007. [cited at p. 8]

[47] A. Natanzon, R. Shamir, and R. Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109 – 128, 2001. [cited at p. 62]

[48] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc, New York, 1999. [cited at p. 92]

[49] A. Oliveira and L. Lorena. Sequencing cutting patterns and VLSI gates by population training algorithms. 2003. [cited at p. 35]

[50] A. C. Oliveira and L. A. Lorena. *2-Opt Population Training for Minimization of Open Stack Problem*, volume 2507 of *Lecture Notes in Artificial Intelligence*, chapter Advances in Artificial Intelligence, pages 313–323. Springer, 2002. [cited at p. 35]

[51] B. Panda and S. Das. A linear time recognition algorithm for proper interval graphs. *Information Processing Letters*, 87(3):153–161, August 2003. [cited at p. 51]

[52] S.-L. Peng and C.-K. Chen. On the interval completion of chordal graphs. *Discrete Applied Mathematics*, 154(6):1003 – 1010, 2006. [cited at p. 62]

[53] G. Pileggi, R. Morabito, and M. Arenales. Abordagens para otimização integrada dos problemas de geração e sequenciamento de padrões de corte: caso unidimensional. *Pesquisa Operacional*, 25(3):417–447, 2005. [cited at p. 139]

[54] M. J. Pinto. *Algumas contribuições à resolução do problema de corte integrado ao problema de sequenciamento dos padrões*. PhD thesis, Instituto Nacional de Pesquisas Espaciais, São José dos Campos, Brasil, Junho 2004. [cited at p. 14, 22, 35]

[55] G. Reinelt. A note on small linear-ordering polytopes. *Discrete and Computational Geometry*, 10(1):67–78, Dec. 1993. [cited at p. 59, 60, 77]

[56] B. Smith and I. Gent. Constraint modelling challenge 2005. In *The Fifth Workshop on Modelling and Solving Problems with Constraints, IJCAI 05*, Edinburgh, Scotland, July 2005. [cited at p. 34]

[57] S. C. Tang and E. V. Denardo. Models arising from a flexible manufacturing machine, part I: Minimization of the number of tool switches. *Operations Research*, 36(5):767–777, September-October 1988. [cited at p. 16, 17, 20, 35, 133, 134]

[58] Y. P. Tsao and G. J. Chang. Profile minimization on compositions of graphs. *Journal of Combinatorial Optimization*, 14(2-3):177–190, Oct. 2007. [cited at p. 62]

[59] L. Wolsey. *Integer Programming.* Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc, New York, 1998. [cited at p. 92]

[60] G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109 – 1130, 2007. [cited at p. 5, 6]

[61] H. H. Yanasse. Minimization of open orders - polynomial algorithms for some special cases. *Pesquisa Operacional*, 16(1):1–26, June 1996. [cited at p. 10, 28, 70, 72]

[62] H. H. Yanasse. An exact algorithm for the tree case of the minimization of open orders problem. In *XXIX Simpósio Brasileiro de Pesquisa Operacional*, page 100, Salvador, Brazil, 22-24 October 1997. [cited at p. 29]

[63] H. H. Yanasse. On a pattern sequencing problem to minimize the maximum number of open stacks. *European Journal of Operational Research*, 100:454–463, 1997. [cited at p. 13, 14, 20, 35, 71, 134]

[64] H. H. Yanasse. A transformation for solving a pattern sequencing problem in the wood cut industry. *Pesquisa Operacional*, 17(1):57–70, 1997. [cited at p. 28, 71]

[65] H. H. Yanasse and M. J. P. Lamosa. An integrated cutting stock and sequencing problem. *European Journal of Operational Research*, 183(3):1353–1370, 2007. [cited at p. 139]

[66] H. H. Yanasse and M. J. Pinto. Uma nova formulação para um problema de sequenciamento de padrões em ambientes de corte. In INPE, editor, *XXXV SBPO*, pages 1516–1524, Natal, Brazil, November 4th to 7th 2003. [cited at p. 23, 27]

[67] H. H. Yanasse and E. L. F. Senne. The minimization of open stacks problem: A review of some properties and their use in pre-processing operations. *European Journal of Operational Research*, 203(3):559 – 567, 2010. [cited at p. 8, 29]

[68] B. J. Yuen. Heuristics for sequencing cutting patterns. *European Journal of Operations Research*, 55(2):183–190, November 1991. [cited at p. 7]

[69] B. J. Yuen. Improved heuristics for sequencing cutting patterns. *European Journal of Operational Research*, 87(1):57 – 64, 1995. [cited at p. 30]

[70] B. J. Yuen and K. V. Richardson. Establishing the optimality of sequencing heuristics for cutting stock problems. *European Journal of Operations Research*, 84:590–598, 1995. [cited at p. 7, 35]

Index

adjacency matrix, 55
adjacent vertices, 38
affinely independent, 91
anti-neighborhood, 38
arc, 37
asteroidal triple, 46
AT-free, 46

banded matrix, 66
bandwidth of a graph, 57
bandwidth of a matrix, 55
Bin Packing Problem, 5
branch-and-bound, 3

chord, 42
chordal graph, 42
chordless cycle, 42, 106
chromatic number, 41
clique, 39
clique matrix, 47
clique number, 40
closed neighborhood, 38
co-comparability graph, 45, 102
coloring, 41, 111
comparability graph, 45
complement graph, 39
complete graph, 39
consecutive 1's property, 48
Consecutive Blocks Minimization, 12
cutting pattern, 6
Cutting Stock Problem, 5
cutwidth, 56
cycle, 42

degree, 39
digraph, 38
directed graph, 38
discontinuity, 12
DNA physical mapping, 64
dominance of patterns, 29

edge, 37
edge completion problem, 60
edge cut, 55
edge length, 56
envelope of the adjacency matrix, 55
equivalence of nodes in MOSP graph, 29

face, 92
facet, 92
fill edge, 60, 76
full envelope, 55
full-dimensional, 92

gate, 66
Gate Matrix Layout Problem, 67
graph, 37

Hamiltonian path, 19

indegree, 38
independent set, 40
Integer Programming, 2
interval graph, 46, 78
interval graph completion, 61
Interval Graph Sandwich Problem, 64
interval representation, 46
IP formulation for Minimum IGC, 126
IP formulation for MOSP, 87, 89, 116

linear ordering, 43, 75

Linear Ordering Problem, 59
lower bound, 85

Möbius-ladder, 60
maximal clique, 40
Minimal fill-in, 62
Minimal Interval Graph Completion, 62
Minimization of Discontinuities, 12
Minimization of Open Orders, 10
Minimization of Open Stacks, 7, 69
Minimization of Order Spread, 11, 129
Minimization of Stack Occupation, 128
Minimization of Tool Switches, 12, 133
Minimum fill-in, 61
Minimum Interval Graph Completion, 61, 124
Minimum Sum Cut Problem, 54
Minimum triangulation, 61
Mixed Integer Programming, 24
modified cutwidth, 56
modified edge cut, 56
MOSP graph, 28, 70
MOSP number, 8, 69, 85

neighborhood, 38
net, 67
node, 37
NP-hard, 2

odd anti-hole, 52
odd hole, 52
open stack, 8
order spread, 11
oriented graph, 38
outdegree, 38

path, 42
path decomposition, 58
pathwidth, 58
Pattern Allocation Problem, 6
Pattern Sequencing Problem, 6
perfect elimination order, 43
perfect elimination scheme, 43
perfect graph, 51
polyhedron, 92
polytope, 92
predecessor, 43
Profile Minimization Problem, 54
profile of a graph, 54
profile of a matrix, 55
pseudo-equivalence, 29

representative vertex, 111

sandwich graph, 64
sequence dating problem, 65
seriation problem, 65
simple k-fence, 60
simplicial vertex, 43
split graph, 44
stability number, 40
stable set, 40
successor, 43
sum cut, 54

Tool Replacement Problem, 19
tool switch, 12
tournament, 59
track, 67
Travelling Salesman Problem, 14
tree, 42
tree decomposition, 57
treewidth, 58
triangulated graph, 42
triangulation, 61

umbrella, 48
undirected graph, 38

valid inequality, 92
vertex, 37
vertex cut, 54
vertex separation, 54
vertex separator, 44
VLSI Layout Problem, 66