Pesquisa Operacional (2015) 35(2): 213-250
© 2015 Brazilian Operations Research Society
Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope
doi: 10.1590/0101-7438.2015.035.02.0213

GRAPH PROPERTIES OF MINIMIZATION OF OPEN STACKS PROBLEMS AND A NEW INTEGER PROGRAMMING MODEL*

Isabel Cristina Lopes[1] and J.M. Valério de Carvalho[2]**

Received May 22, 2015 / Accepted June 18, 2015

ABSTRACT. The Minimization of Open Stacks Problem (MOSP) is a Pattern Sequencing Problem that often arises in industry. Besides the MOSP, there are also other related Pattern Sequencing Problems of similar relevance. In this paper, we show that each feasible solution to the MOSP results from an ordering of the vertices of a graph that defines the instance to solve, and that the MOSP can be seen as an edge completion problem that renders that graph an interval graph. We review concepts from graph theory, in particular related to interval graphs, comparability graphs and chordal graphs, to provide insight to the structural properties of the admissible solutions of Pattern Sequencing Problems. Then, using Olariu's characterization and other structural properties of interval graphs, we derive an integer programming model for the MOSP. Some computational results for the model are presented.

Keywords: integer programming, graph layout problems, Minimization of Open Stacks Problem.

1 INTRODUCTION

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 deal with the generation of a set of cutting patterns that minimizes waste. But, beyond pattern generation, there are often additional aspects to deal with in the process of planning industrial cutting operations. An important issue is to define the sequence in which the patterns are cut. Most probably the first researcher raising awareness to these aspects was Dyson [12]. The *Pattern Sequencing Problems* (PSP), also referred to as the *Pattern Allocation*

*Invited paper.
**Corresponding author.

[1]LEMA/CIEFGEI/ESEIG-IPP – Escola Superior de Estudos Industriais e de Gestão, Instituto Politécnico do Porto, Rua D. Sancho I, 981, 4480-876 Vila do Conde, Portugal. E-mail: cristinalopes@eseig.ipp.pt
[2]Departamento de Produção e Sistemas, Escola de Engenharia, Universidade do Minho, Campus de Gualtar, 4710-057 Braga, Portugal. E-mail: vc@dps.uminho.pt

Problems (PAP), consist in finding the permutation of the predetermined cutting patterns that optimizes a given objective function, related, for instance, with the number of tool changes, the average order spread, the number of discontinuities or 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 0, otherwise. Pattern sequencing problems consist of constructing a permutation of the columns of this matrix, while minimizing some given objective function. The selected permutation of the columns provides the order for processing the patterns. There are, evidently, $m!$ solutions.

Consider a cutting machine that processes just one cutting pattern at a time. The items of the same type already cut are piled in a stack 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 working area only after all items of that type 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. The Minimization of Open Stacks Problem (MOSP) is a pattern sequencing problem that was first addressed in 1991 by Yuen [42] and Richardson [43]. It arose in the Australian flat glass industry, but it can appear in other cutting industries like steel tubes, paper, wooden panels, and others.

Most papers on pattern sequencing study the MOSP, maybe because the problem itself has a complex structure and it is very rich in applications to other fields of science. Many authors use it while solving a two stage procedure: first, they solve the classic problem of finding the best patterns to cut stock sheets, and only then, in a second stage, do they deal with determining 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 [32, 33, 39], and others use the number of open stacks rather as a constraint than as the objective function [35, 26].

There are several papers that propose integer programming models for pattern sequencing problems, which are described with detail in section 2. Madsen [30] was the first to present an exact formulation based on the Travelling Salesman Problem (TSP) as a model for the Minimization of Order Spread Problem (MORP), but in fact it only solves the Minimization of Discontinuities Problem (MDP) because it does not consider the duration of the discontinuities when counting the order spread [14]. Bard [3] presented a nonlinear model for the MSTP using binary variables to assign the jobs to the n available positions in time. Tang & Denardo [35] also used time indexed binary variables to develop an IP formulation for the Minimization of Tool Switches

Problem (MTSP). In [37], Yanasse proved two propositions that make the model by Tang and Denardo suitable for modeling the MOSP. There is also another IP model based on the TSP by Laporte, González & Semet [26] for the MTSP where the scheduling uses linear ordering variables, that later was adapted to the MOSP by Pinto [33]. Yanasse and Pinto also proposed a new IP model for the MOSP [40] which aims at sequencing the completion of stacks rather than focusing on sequencing the patterns. And there is a Mixed Integer Programming formulation for the MOSP by Baptiste [2], submitted to the *2005 Constraint Modeling Challenge*.

In this paper, we present a new 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 adding arcs to this graph, converting it into an interval graph and defining an ordering of the vertices based on a sequence of cliques.

This paper is organized as follows. In Section 2, we present the MOSP and some related problems. Then, in Section 3, review concepts in graph theory that are related to the structure of the solutions of the MOSP. In Section 4, we review MOSP graphs, and, in Section 5, we derive an integer programming model. In Section 6, computational results are presented, and afterwards, some conclusions are drawn.

2 MOSP: MINIMIZATION OF THE NUMBER OF OPEN STACKS

Consider the matrix $A_{n\times m}$ with the specification of the m cutting patterns, whose element a_{ij} equals 1 if pattern j contains item i, and 0, otherwise. A sequence to process the cutting patterns is a permutation $\Pi = (\pi_1, \ldots, \pi_m)$ 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 $\Pi = (\pi_1, \ldots, \pi_m)$ of the m patterns as

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

where $|.|$ denotes the cardinality of the set.

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 from [41] with 8 cutting patterns and 6 items. The composition of the cutting patterns is described in Table 1. If the patterns are processed in the original sequence (Figure 1a), there is a time when there are 5 simultaneously open stacks. If the ordering of the patterns is as in Figure 1b), there are only 4 simultaneous open stacks at most.

Table 1 – An example of the MOSP with 8 patterns and 6 items.

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

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

(a)

Items	Patterns							
	P4	P5	P3	P1	P2	P6	P7	P8
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	0	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

(b)

Figure 1 – Two solutions for the Pattern Sequencing Problem.

MOSP has been proved to be a NP-hard problem [28]. Besides the applications referred to above, it arises in production planning (for rapidly fulfilling the customers' orders) [36], and also in other fields such as VLSI Circuit Design with the Gate Matrix Layout Problem and PLA Folding [28], and in classical problems from Graph Theory presented in Section 3.8 such as Pathwidth, Modified Cutwidth and Vertex Separation.

2.1 Other pattern sequencing problems related to MOSP

The Minimization of Order Spread Problem (MORP) arises when it is desirable to keep all the pieces cut belonging to one order as close in time as possible, as, for instance, in the glass cutting industry, where the glass pieces have to be handled and stored individually, which is very time- and storage-consuming, and hence expensive. Reducing the time elapsed between cutting all the pieces corresponding to the same order cuts down the handling and storing costs [30]. A stack is created for each customer, as happens in the MOSP. 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 [30]. The objective of the MORP is therefore to minimize the order spread, which will minimize the time that the stacks remain open.

The Minimization of Discontinuities Problem (MDP) consists in finding a sequence to process the cutting patterns such that the number of discontinuities is minimum. 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 is cut again later. The difference from the previous problem is that the duration of the discontinuities does not 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 [17].

Another problem is the Minimization of Tool Switches Problem (MTSP). This is a job scheduling problem that arises in flexible manufacturing machines, and has been applied, for instance, in the metal working industry and in the assembling operations of printed circuit boards. Machines in such systems are capable of different tasks, but may need a certain combination of tools. This problem considers machines that can hold a set of tools, which can be changed in order to have the adequate set of 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 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. A link between MTSP and MOSP can be established considering the jobs as cutting patterns and the tools as items to be cut.

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 & Yanasse [28] presented counterexamples to all the equivalence conjectures, except for the equivalence between the MTSP and MDP. They proved that if MTSP is fixed parameter tractable (FPT), then MDP is also FPT. Yanasse [37] showed that, although the MOSP is not equivalent to the MTSP in a general case, they are equivalent when the optimum of the MOSP (denoted by C^*) equals the capacity C of the machine in the MTSP. 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.

3 GRAPHS AND LAYOUT PROBLEMS

3.1 Basic Definitions

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 are exactly k edges $e_1, e_2, \dots, e_k$ such that $e_1 = e_2 = \cdots = e_k = [uv]$. 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 are assumed to be simple graphs.

A *directed graph* or *digraph* is a graph $G = (V, E)$ where each edge in E has 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 $outdeg(v)$. 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 $E \bigcap E^{-1} = \emptyset$. In this work, we will mostly use undirected graphs; for this reason, from this point forward, unless stated otherwise, by graph we mean an undirected graph, and we will use without distinction $[uv]$ and $[vu]$. 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. In simple graphs, as edges from a vertex to itself are not allowed, one vertex does not belong to its own neighborhood. So it is also usual to define the closed neighborhood of a vertex if we want to include it. 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 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 $N(v)$.

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. A graph is *complete* if every pair of distinct vertices is connected by one edge. The complete graph on n vertices is usually denoted by K_n. In a simple graph, the number of all possible edges is $\binom{n}{2}$.

3.2 Graph Optimization Problems

There are several problems in Graph Theory that are somehow related to the MOSP problem. The solutions to those problems provide graph measures that are illustrated in Figure 2.

Maximum Clique Number: find the set of vertices that form the largest clique in a graph. 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 is denoted by $\omega(G)$.

Minimum Clique Cover: find a set of cliques to cover all the vertices in the graph. 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 is denoted by $k(G)$.

Maximum Independent Set: find the largest set of vertices nonadjacent to each other. 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 is denoted by $\alpha(G)$.

Chromatic Number: find the minimum number of colors that are necessary to color the vertices of a graph using different colors for adjacent vertices. A *proper k-coloring* of $G = (V, E)$ is a

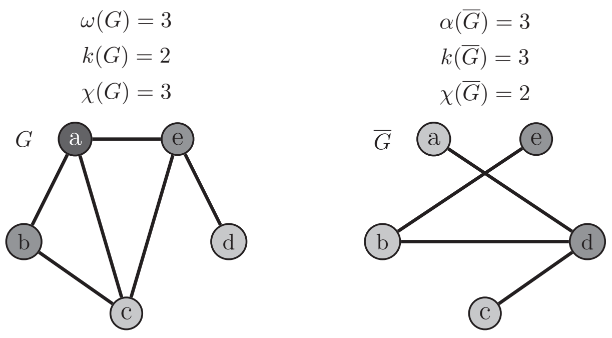


Figure 2 – Chromatic number, stability number, clique cover number and clique number of a graph and its complement graph.

partition of the vertices $V = X_1 \cup X_2 \cup \cdots \cup X_k$ such that each X_i is a stable set. In a proper k-coloring there is an assignment of integers $\{1, 2, \ldots, k\}$ (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 is denoted by $\chi(G)$.

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:

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

Since every vertex of a maximum stable set must be contained in a different partition segment in any minimum clique cover, it is valid that:

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

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 [17]. However, for a special type of graphs, that are called *perfect graphs*, linear time algorithms are known. The class of perfect graphs includes interval graphs, chordal graphs and comparability graphs that play a central role in the structure of the solutions of Pattern Sequencing Problems.

3.3 Chordal Graphs

A *path* is a sequence of vertices $[v_0, v_1, \dots, v_k]$ such that $[v_{i-1}v_i]$ is an edge for $i = 1, \dots, k$ 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. 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 $[v_0, v_1, \dots, v_k, v_0]$ 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 1. *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.

3.4 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 2. *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 [18], it is known that simplicial vertices appear in all chordal graphs:

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

Definition 4. *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 5. *A linear ordering $\sigma = [v_1, v_2, \dots, v_n]$ 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 $G_{v_i,\dots,v_n}$.*

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

Definition 6. *[4] 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 7. *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 [18]:

Theorem 2. *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 & Gross [16]. 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. This procedure was used by Lueker (1974) and Rose & Tarjan (1975) to write a linear time algorithm to recognize chordal graphs.

3.5 Comparability Graphs

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

Definition 8. *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:*

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

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*.

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

3.6 Interval Graphs

Definition 9. *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. Recognizing whether a given graph is an interval graph can be carried out in linear time.

Besides the definition, there are theorems that characterize interval graphs. Lekkerkerker & Boland (1962) characterization [27] focuses on the fact that an interval graph cannot branch into more than two directions nor circle back onto itself.

Theorem 3. *[18] 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 statement in this theorem introduces the concept of asteroidal triple: 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 states 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 [9].

The graph in Figure 3 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 $N(c) = f$.

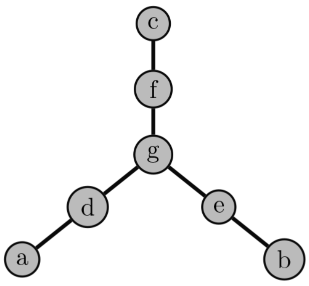


Figure 3 – Not an interval graph.

Some other characterizations of interval graphs are known, namely the following theorem (Gilmore & Hoffman, 1964) as cited in [18]:

Theorem 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.*

Note that, in this theorem, it is stated 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. An example is shown in Figure 4.

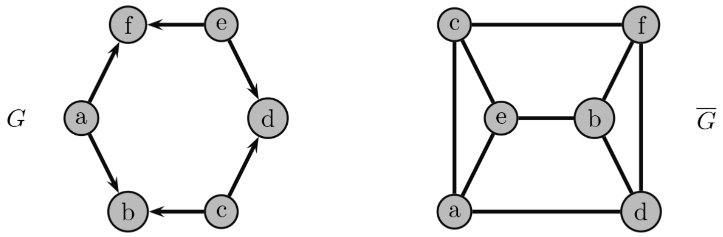


Figure 4 – G is a comparability graph but $\overline{G}$ is not an interval graph.

The last statement in Theorem 4 is related to another characterization of interval graphs by Fulkerson & Gross (1965), which refers to the clique matrix of a graph. 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.

Definition 10. *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 5. *[16] 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 [5].

Theorem 6. *$G = (V, E)$ is an interval graph if and only if there exists a linear ordering of G such that the associated adjacency matrix A verifies:*

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

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

There is an alternative characterization of interval graphs, due to Olariu [9], that uses the linear ordering of the vertices, and is illustrated in Figure 5:

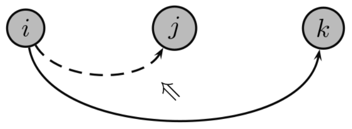


Figure 5 – Olariu's characterization of interval graphs.

Theorem 7. *A graph $G = (V, E)$ is an interval graph if and only if there exists a linear ordering $\varphi : V \to \{1, \dots, N\}$ such that $\forall i, j, k \in V : \varphi(i) < \varphi(j) < \varphi(k)$ we have $[ik] \in E \Rightarrow [ij] \in E$.*

We will use this characterization to develop an integer programming model for the MOSP. Starting from a graph that represents the MOSP instance, new edges are added to the MOSP graph to assure that the graph of the solution is an interval graph, as described in detail in Section 4. Issues related to adding edges (edge completion) are addressed in Section 3.10.

The ordering of the maximal cliques in an interval graph referred to in Theorem 4 will allow us to set an interesting ordering for the vertices, using the following theorem from Biedl [4].

Theorem 8. *An interval graph G has an interval representation such that all endpoints of intervals are distinct integers.*

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 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: Let $v_1, \dots, v_n$ be a 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. Note that if a graph G has a perfect elimination order, then G is chordal.

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 [4].

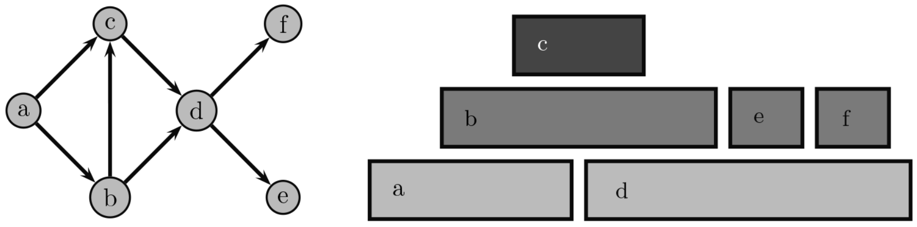


Figure 6 – The p.e.o. a, b, c, d, e, f gives the linear ordering of the left endpoints of the intervals.

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 7. 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.

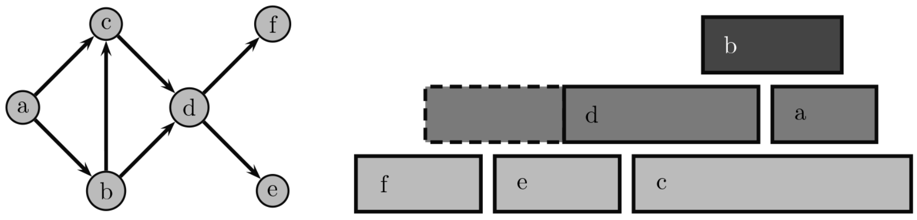


Figure 7 – The p.e.s. f, e, d, c, b, a is not adequate for ordering the left endpoints of the intervals.

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 [31], 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), but have a p.e.o. because they are chordal graphs.

3.7 Perfect Graphs

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

Definition 11. *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 9. ***(Perfect Graph Theorem)*** *[18] 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 1. ***(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 proved by Chudnovsky, Seymour, Robertson and Thomas in 2003 [8].

Theorem 10. *[18] 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.

There is a polynomial time algorithm to test whether a graph is perfect, as shown by Cornuéjols, Li & Vušković [10]. Biedl showed in [4] that many problems which are generally NP-hard, such as the Clique, Coloring and Maximum Independent Set problems, can be solved in polynomial time when circumscribed to perfect graphs.

3.8 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 measures that naturally can be taken and used to describe geometric properties of the graph.

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 12. *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.

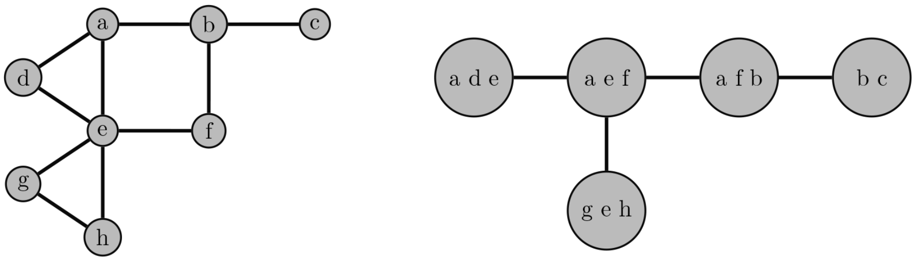


Figure 8 – A graph with a tree decomposition of width 2 [6].

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.

Definition 13. *Given a tree decomposition $T = (I, F)$, the* width of the tree decomposition *is* $\max_{i \in I} |X_i| - 1$.

Definition 14. *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 1. *[6] If G is chordal then G has a tree decomposition of width* $\omega(G) - 1$.

The minimum degree $\delta(G)$ of the vertices of a graph G and the *degeneracy* of a graph $\delta D(G)$, defined by $\delta D(G) := \max_{H \subseteq G} \delta(H)$, are lower bounds for the treewidth:

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

Computing the treewidth is a NP-hard problem [1, 7].

Pathwidth

A particular case occurs when we require the decomposition to be a path, which is called a *path decomposition of width k*.

Definition 15. *A graph G has* pathwidth $PW(G)$ *bounded by k if G has a tree decomposition T of width k such that T is a path.*

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 $TW(G) \leq PW(G)$.

In [28], Yanasse and Linhares pointed out that the MOSP, the Gate Matrix Layout Problem (GMLP) and the Pathwidth are equivalent problems that have been studied independently in the literature. In fact, Kinnersley proved in [24] the equivalence between the Pathwidth and the Vertex Separation problem and showed that the GMLP cost of a graph equals its pathwidth plus one. Hence the pathwidth is equivalent to the MOSP. In fact, the pathwidth problem consists of finding an interval supergraph with the smallest clique number [15].

3.9 Linear ordering

Many graph layout problems involve finding a linear ordering of the vertices to optimize a given objective function. One example is the *Linear Ordering Problem* (LOP). It 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. It belongs to the class of NP-hard combinatorial optimization problems, and its integer programming formulation and its polytope were studied in [13, 19, 34]. The decision variables are defined as:

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

We will not present the LOP, but just focus on the constraints that define the linear ordering of the vertices:

$$x_{ij} + x_{ji} = 1 \qquad \forall i, j \in V, i < j \tag{1}$$

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

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V, i < j \tag{3}$$

The first set of contraints (1) means that, in the linear ordering, either vertex i is before j or vice versa, reducing this to a minimal equation system. The inequalities (2) are called the 3-dicycle inequalities and along with the first equations guarantee that the graph is free from cycles. The last inequalities (3) are called hypercube constraints.

The inequalities (2) and (3) define facets of the linear ordering polytope [19]. For $n \leq 5$, the inequalities (1)-(3) are sufficient to describe the linear ordering polytope, but for $n > 6$ more facet defining constraints are needed [34].

3.10 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 $E \cap F = \varnothing$ for distinguishing the fill edges in F from the original ones in E.

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 addressed. 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 is 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 [25], split graphs [21], proper interval graphs [23] and comparability graphs [22] have also been studied.

Minimum vs. Minimal Edge Completion Problems

There are also variants depending on the selected cost function. 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 [6]. By Lemma 1, the *treewidth* of a graph G coincides with $\min_H \omega(H) - 1$ for all triangulations H of G. 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 $\omega(H) - 1$.

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* [20].

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.

4 MOSP GRAPH

A MOSP problem with at most two different items per pattern can be represented through a graph that associates vertices to orders and arcs to patterns [38]. By making each item correspond to a vertex, and considering two vertices to be adjacent if and only if the corresponding items are simultaneously present in a pattern, we obtain a *MOSP graph*.

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) [38]. 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 [38], Yanasse proves that the maximum number of stacks in both problems is the same.

As an example, consider the MOSP instance with seven patterns and six different items presented in Table 2. This instance originates a MOSP graph with 6 vertices, one for each item, and with edges between the vertices (items) that belong to the same pattern. Notice that, for example, 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.

Table 2 – An instance of the MOSP with 7 cutting patterns and 6 items.

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		

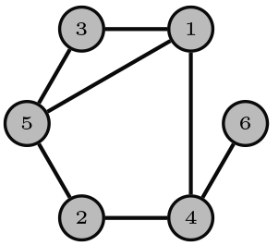


Figure 9 – MOSP Graph of the instance in Table 2.

Furthermore, Yanasse showed that it is possible to take a solution for the ordering of the vertices of the MOSP graph and construct a sequence for the corresponding cutting patterns [38]. 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.

Some simplifications are possible. 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 [37] 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. It happens, for instance, with patterns P6 or P7 in Table 1. Each pattern should be sequenced just before the first of the patterns containing that item, and the number of simultaneously open stacks will not increase [37]. Therefore, this instance can be reduced to only five relevant patterns (P1, P2, P3, P4 and P5) generating the same graph.

There are other situations in which patterns can be removed from the original problem before solving it, and then 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.

For the example in Table 1, if the patterns are sequenced by the ordering $P_1 P_2 P_3 P_4 P_5 P_6 P_7$, 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 $P_3 P_7 P_2 P_5 P_6 P_4 P_1$. 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, which is the optimum for this instance, as can be observed in Figure 10.

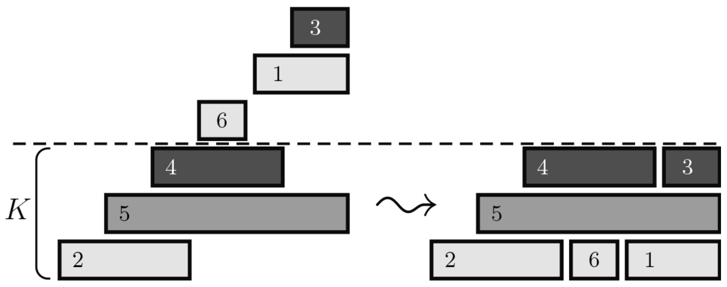


Figure 10 – Non simultaneous items can share stack space.

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 saw 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 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 in Figure 9, the interval graph corresponding to the solution displayed in Figure 10 has the same vertices and edges of the MOSP graph and two additional edges, as depicted in Figure 11. This is an interval graph completion (as explained in Section 3.10) 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.

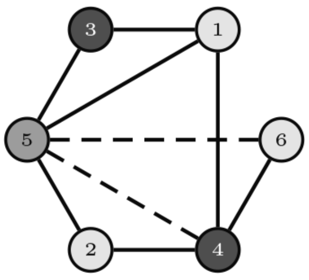


Figure 11 – Interval Graph of the instance in Table 2.

As discussed in Section 3.6, 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.

5 AN INTEGER PROGRAMMING MODEL FOR THE MOSP

Given an instance of the problem, we first build a MOSP 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. Each arc of the future interval graph means that, for a period of time, the two stacks (the 2 vertices that are endpoints 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 remaining 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. We will use the characterization in Theorem 7 to guarantee that the graph obtained in the solution of the problem is an interval graph.

5.1 Decision Variables

We set an ordering for opening the stacks by assigning a number to each item cut, with a bijective function $\varphi : V \to \{1, \dots, N\}$. 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 set an orientation into the arcs, to keep track of the sequence of the items. If $x_{ij} = 1$ then item i starts being cut before the item j, even though the corresponding stacks may overlap or not, i.e., in spite of having an arc between the two vertices or not.

Other decision variables will be used to identify the arcs that are added 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. 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:

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

Note that y_{ij} is 1 when the arc $[ij]$ is NOT added. 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 12.

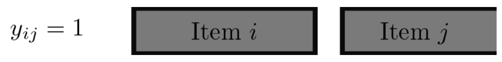


Figure 12 – Interval i opens and closes before j starts.

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

$$y_{ij} \leq x_{ij} \quad \forall i, j \in V : i \neq j, [ij] \notin E \tag{4}$$

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.

5.2 Edge Completion to Obtain 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 7, to express relations between the binary variables y_{ij} and x_{ij}. Recall that Olariu's 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 $[ij] \in E$, as shown in Figure 5.

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 $[ij]$ have to be added. Let us separate the analysis in two cases: (i) arc $[ik] \in E$ and (ii) 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 5.

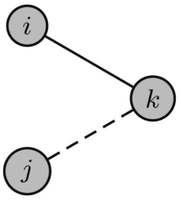
This inequality can be strengthened as follows. Combine the inequality with the inequalities $y_{ij} \leq x_{ij}$ and $x_{jk} \leq 1$ to obtain:

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

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

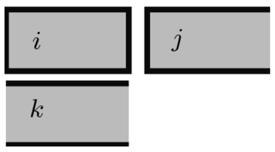
$$y_{ij} \leq x_{kj} \quad \forall i, j, k \in V, [ij] \notin E, [ik] \in E \tag{5}$$

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) 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.

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

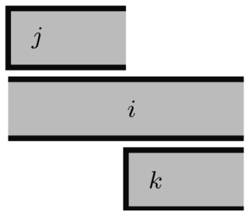


In the example presented in Section 4, the vertices 2, 6 and 4 form a set in these conditions, because $[24] \in E$ but $[2, 6] \notin E$. Hence, in the model for this example there is the inequality $y_{26} \leq x_{46}$. In the solution, as can be observed in Figure 10, interval 2 opens and closes before 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 11.

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

On the other hand, the arc $[ik]$ may not be originally in the set of arcs E, but it may be added, 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 $y_{ij} + (x_{ik} - y_{ik}) + x_{jk} \leq 2$. 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 $y_{ij} + x_{jk} - y_{ik} \leq 1$. This inequality is equivalent to a constraint in binary variables equivalent to the logical implication $y_{ij} \Rightarrow x_{kj} \vee y_{ik}$:

$$y_{ij} \leq x_{kj} + y_{ik} \qquad \forall i, j, k \in V, [ij], [ik] \notin E \tag{6}$$

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 3.

Table 3 – Possible cases when arc $[ik] \in F$.

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

In the second, fifth and sixth cases ($k \prec j$), adding 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) 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 13 and 14, or $y_{ik} = 1$, meaning that i should end before k starts, as depicted in Figures 14 and 15.

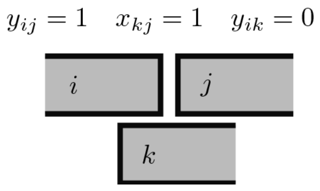


Figure 13 – Interval i closes before interval j opens, with interval k being simultaneous to interval i and opening before j.

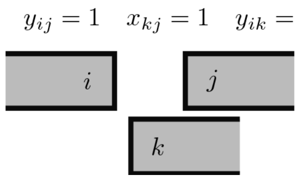


Figure 14 – Interval i closes before intervals j and k open, with interval k opening before interval j.

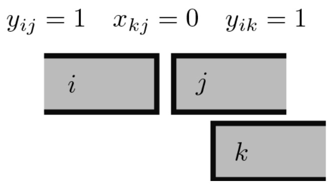
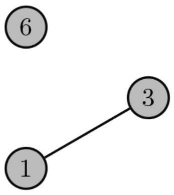


Figure 15 – Interval i closes before intervals j and k open, with interval j opening before interval k.

In the example, the three vertices 6, 1 and 3 originate the inequality $y_{61} \leq x_{31} + y_{63}$.



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 arcs were not added, making these three variables equal to one.

5.3 Objective function

The position of vertex j in the linear ordering is found counting the number of vertices that precede it. 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. On the other hand, variable $y_{ij} = 1$ means that vertex i closes before vertex j opens. For every vertex j, the sum $\sum_{i=1}^{N} y_{ij}$ counts how many intervals finish before interval j starts.

So, when vertex j opens, the number of intervals that are open at that instant is $\sum_{i=1}^{N} x_{ij} - \sum_{i=1}^{N} y_{ij} + 1$, where the constant 1 accounts for the vertex j itself. This leads to a set of functions that can be used to evaluate the MOSP number:

$$\sum_{\substack{i=1 \\ i \neq j}}^{N} x_{ij} - \sum_{\substack{i=1 \\ [ij] \notin E}}^{N} y_{ij} + 1 \leq K \qquad \forall j = 1, \ldots, N \tag{7}$$

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 16, the number of lines open when an interval starts is a lower bound for the maximum number of open stacks.

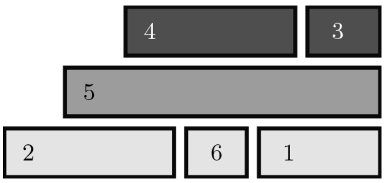


Figure 16 – Optimal solution of the example from Table 2.

In the example presented before, when interval 5 starts, the number of open stacks turns to two, i.e., for $j = 5$:

$$\sum x_{ij} - \sum y_{ij} + 1 = 1 - 0 + 1 = 2 \leq 3,$$

which is a lower bound for the MOSP (which is three). The same inequality, corresponding 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, for $j = 1$:

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

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, but all the others are maximal with size 3, which is the optimum for this instance of the MOSP.

The value of the optimum of the MOSP is equal to the size of the largest 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.

Our basic new mathematical formulation for the MOSP problem is:

Minimize K

Subject to:

$$x_{ij} + x_{ji} = 1 \qquad \forall i, j = 1, \ldots, N \text{ with } i \neq j \tag{8}$$

$$x_{ij} + x_{jk} + x_{ki} \leq 2 \qquad \forall i, j, k = 1, \ldots, N \text{ with } i \neq j \neq k \tag{9}$$

$$y_{ij} \leq x_{ij} \qquad \forall i, j = 1, \ldots, N \text{ with } i \neq j \text{ and } [ij] \notin E \tag{10}$$

$$y_{ij} \leq x_{kj} \qquad \forall i, j, k = 1, \ldots, N \text{ with } [ij] \notin E, [ik] \in E \tag{11}$$

$$y_{ij} - y_{ik} \leq x_{kj} \qquad \forall i, j, k = 1, \ldots, N \text{ with } [ij], [ik] \notin E \tag{12}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{N} x_{ij} - \sum_{\substack{i=1 \\ [ij] \notin E}}^{N} y_{ij} + 1 \leq K \qquad \forall j = 1, \ldots, N \tag{13}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j = 1, \ldots, N \text{ with } i \neq j \tag{14}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i, j = 1, \ldots, N \text{ with } i \neq j, [ij] \notin E \tag{15}$$

$$K \in \mathbb{N} \tag{16}$$

Recall that constraints (8) and (9) are the linear ordering constraints presented in Section 3.9.

The variables x_{ij} were defined for every $i, j = 1, \ldots, N$ 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 equations (8). Later, we will denote the model that only uses variables x_{ij}, for $i < j$, as the reduced model.

5.4 Other Valid Inequalities

The model presented in the last section is based on Olariu's characterization of interval graphs and has an objective function that seeks the interval graph with the best MOSP number. In this section, we present other valid inequalities derived from properties of interval graphs. These inequalities provide insight into the structure of the solutions of the model. In particular, the 4-cycle inequalities proved to be able to strengthen the model.

Neighbor of Successor Inequalities

In an interval graph both variables on the left are not allowed to be simultaneously equal to one without contradicting Theorem 7.

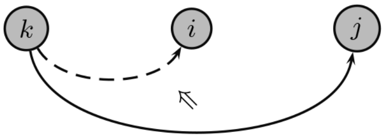
$$y_{ij} + y_{ki} \leq 1 \quad \forall i, j, k \in V \text{ with } [ij], [ik] \notin E, [jk] \in E \tag{17}$$

$$y_{ij} + y_{jk} \leq 1 \quad \forall i, j, k \in V \text{ with } [ij], [jk] \notin E, [ik] \in E \tag{18}$$

$$y_{ij} + y_{lk} \leq 1 \quad \forall i, j, k, l \in V \text{ with } [ij], [kl] \notin E, [jl], [ik] \in E \tag{19}$$

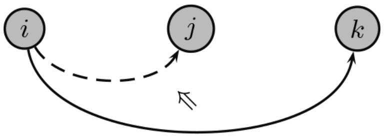
The inequality (17) says that a neighbor of the successor of vertex i, which is vertex k, cannot end before vertex i opens.

If both variables on the left hand side of the inequality (17) 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 7 would force to have the arc $[ki] \in F$, asserted by $y_{ki} = 0$, which contradicts the initial assumption.



The inequality (18) 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.

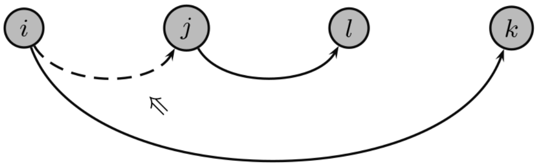
If both variables on the left hand side of the inequality (18) were $y_{ij} = y_{jk} = 1$, then the linear order of the three vertices would be $i \prec j \prec k$. As $[ik] \in E$, Theorem 7 would force adding the arc $[ij] \in F$, making $y_{ij} = 0$, which is absurd.



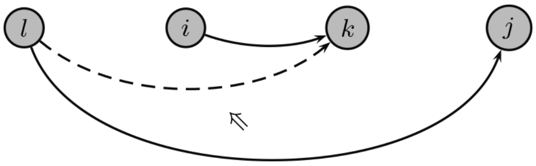
Finally, the inequality (19) declares that a neighbor l of the successor j of vertex i cannot close before the neighbor k of vertex i opens.

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 7 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.



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 $i \prec k$.

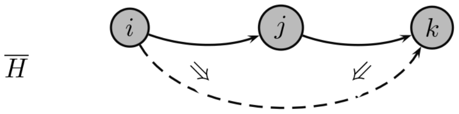


Figure 17 – $\overline{H}$ must be transitively orientable.

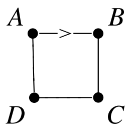
The transitivity of the relation between the variables y comes from the comparability graph property and forces an ordering of the vertices. If a direction is defined in an arc of a graph, that will determine the flow of all the other ones. The variables y define the complement graph, because y_{ij} 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

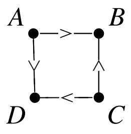
$$y_{ij} = y_{jk} = 1 \Rightarrow y_{ik} = 1$$

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:

$$y_{ij} + y_{jk} - 1 \leq y_{ik} \quad \forall i, j, k \in V, [ij], [jk], [ik] \notin E \tag{20}$$

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 [11, p.322].

Table 4 – An instance for the MOSP with 5 items and 8 patterns.

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		

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.

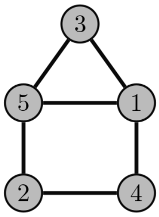
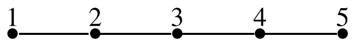
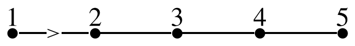


Figure 18 – MOSP Graph of the instance in Table 4.

The graph corresponding 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 18 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

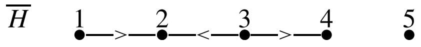
$$y_{12} + y_{23} \leq 1$$

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 feasible solution for this instance corresponds to 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:

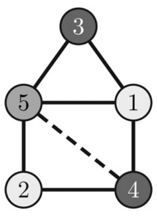
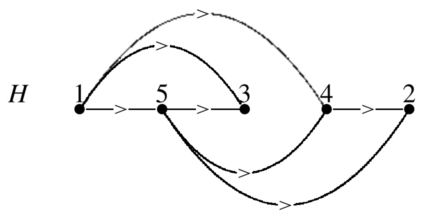
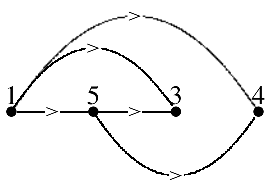


Figure 19 – Interval graph corresponding to the solution of the instance in Table 4.

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:

$$\begin{tikzcd}
1 \arrow[r] \arrow[rr, bend left] & 5 \arrow[r] & 3
\end{tikzcd}$$

After that, one would eliminate vertex 3, followed by vertex 5 and finally vertex 1.

$$\begin{tikzcd}
1 \arrow[r] & 5
\end{tikzcd}$$

$$\begin{tikzcd}
1
\end{tikzcd}$$

The reverse order of the perfect elimination scheme sets the order of the beginning of the intervals: 1-5-3-4-2.

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 20.

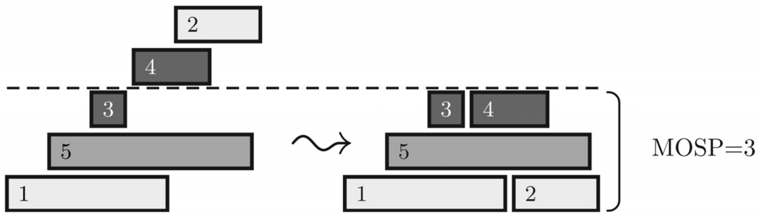


Figure 20 – Optimal solution of the example in Table 4.

Chords in k-cycles

For the graph G 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, as shown in Figure 21.

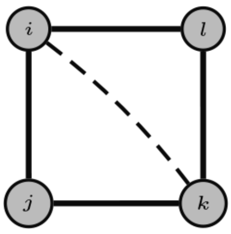


Figure 21 – A 4-cycle must have a chord.

The need to add one of the arcs $[ik]$, $[ki]$, $[jl]$ or $[lj]$ can be expressed by the constraint:

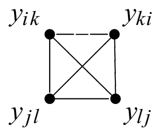
$$y_{ik} + y_{ki} + y_{jl} + y_{lj} \leq 1, \forall [ik], [jl] \notin E, [ij], [jk], [kl], [li] \in E \tag{21}$$

Proposition 1. *Constraints (21) can be derived from the constraints of the MOSP model (11).*

Proof. Consider the following inequalities:

$$y_{ik} \le x_{lk}, y_{jl} \le x_{kl}, \quad y_{ik} \le x_{jk}, y_{lj} \le x_{kj}, \quad y_{ki} \le x_{li}, y_{jl} \le x_{il}, \quad \text{and} \quad y_{ki} \le x_{ji}, y_{lj} \le x_{ij}.$$

Combining the constraints that have the variables x_{pq} and x_{qp} on the right-hand side, with $pq \in \{kl, jk, il, ij\}$ we obtain: $y_{ik} + y_{jl} \le 1$, $y_{ik} + y_{lj} \le 1$, $y_{ki} + y_{jl} \le 1$, and $y_{ki} + y_{lj} \le 1$. Furthermore, $y_{ik} + y_{ki} \le 1$ and $y_{lj} + y_{jl} \le 1$. The six constraints are *clique constraint* that form 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, which means 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, and the constraints (21) for the non-chordal 4-cycle follows.



All the chordless 4-cycles have to be identified prior to the solution of the model, and their number is of order $O(n^4)$. An analysis of constraints for k-cycles, with $k \ge 5$, is presented in [29].

6 COMPUTATIONAL TESTS

The original integer programming model 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.

Only the inequalities discussed in the previous section were added to the plain model. No further actions were taken to improve the efficiency of the solution procedure, as, for instance, including in the model lowers bounds previously published in the literature or using specially tailored heuristics for the MOSP.

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.

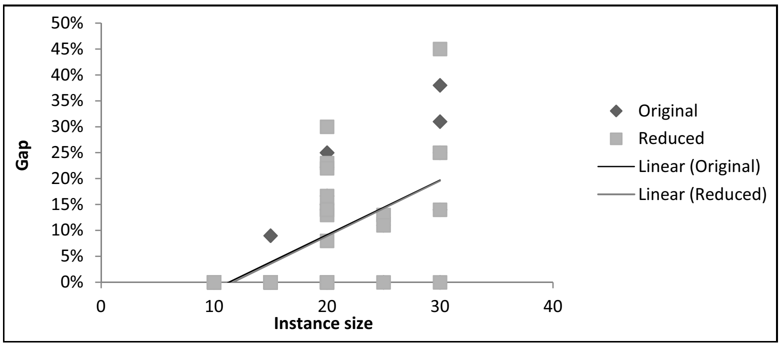
In small instances, the optimal solution is found in just a few seconds. In larger instances, the optimal solution is often found in just a few seconds as well, but it takes too long to prove that it is optimal, specially in instances with many symmetries. In really large instances, the models could not be started because there was not enough memory to handle so many variables and inequalities.

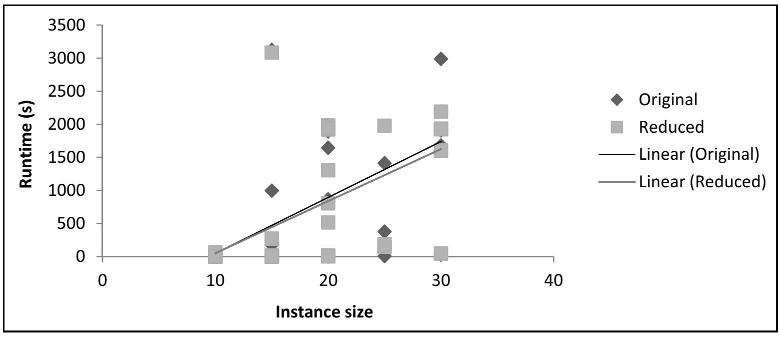
The model with reduced number of variables improved the gap and the runtime in some of the larger instances, but not in all of them. The cells in gray in Table 5 represent the cases where the reduced model improved the gap, the runtime or the number of nodes.

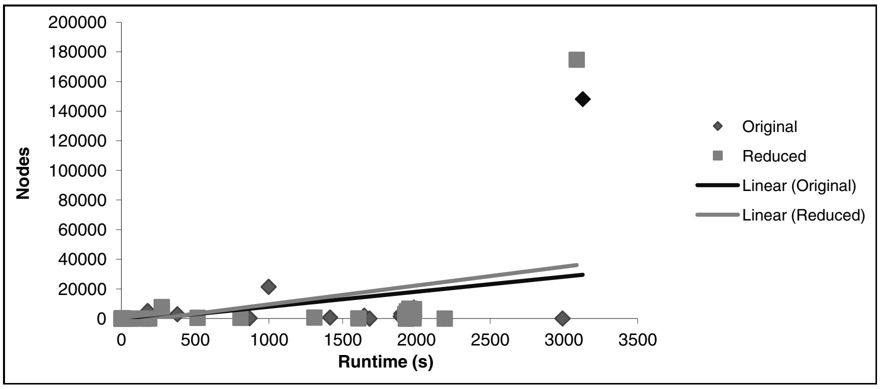
Table 5 – Computational results for the MOSP models.

Instance	No. Items (N)	Original MOSP Model					Reduced MOSP Model				
		Best Objective Value	Best LB	Gap	Runtime (s)	Nodes search tree	Best Objective Value	Best LB	Gap	Runtime (s)	Nodes 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

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.







7 CONCLUSIONS

In this paper, we presented a new integer programming model for the Minimization of Open Stacks Problem (MOSP). It is based on the edge completion of a MOSP graph (to obtain an interval graph) and on a characterization of interval graphs that uses a perfect elimination ordering of the vertices. MOSP solutions have a structure that is similar to the solutions of other pattern sequencing problems, and so the same concepts from graph theory may also apply.

Even though approaches based on dynamic programming may be more efficient to solve the MOSP, the search for MOSP models that may be combined with cutting stock models to tackle the integrated cutting stock and pattern sequencing problem is still of interest.

Clearly, there is room for improvement. The reduction of symmetry in the sequence of the patterns is a crucial factor for improving the runtime. Even in instances with a small number of vertices, the existence of symmetry makes proving the optimality of a solution very difficult and time consuming. It happened, for instance, in the Miller instance in Figure 22. One way to reduce symmetry is to choose an opening order for the neighbors of the first vertex to close and to include it 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.

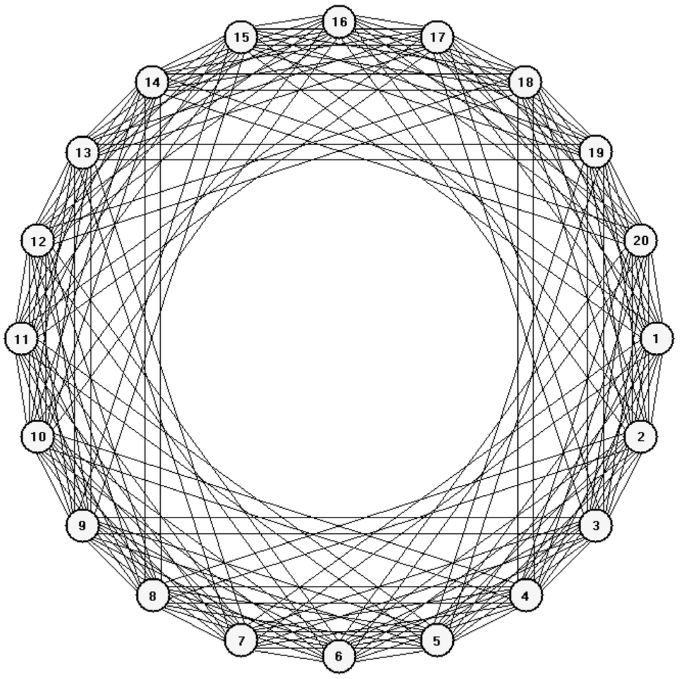


Figure 22 – MOSP graph of the Miller instance.

REFERENCES

[1] Arnborg S, Corneil DG & Proskurowski A. 1987. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, **8**(2): 277–284.

[2] Baptiste P. 2005. Simple mip formulations to minimize the maximum number of open stacks. In *Constraint Modelling Challenge*, pages 9–13, Edinburgh, Scotland, July 31. IJCAI.

[3] Bard JF. 1988. A heuristic for minimizing the number of tool switches on a flexible machine. *IIE Transactions*, **20**(4): 382–391, December.

[4] Biedl T. 2005. *CS 762: Graph-theoretic algorithms – Lecture notes of a graduate course*. University of Waterloo, September.

[5] BILLIONNET A. 1986. On interval graphs and matrice profiles. *RAIRO. Recherche opérationnelle*, **20**(3): 245–256.

[6] BODLAENDER HL & KOSTER AM. 2010. Treewidth computations I. Upper bounds. *Information and Computation*, **208**(3): 259–275.

[7] BODLAENDER HL & KOSTER AMCA. 2008. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, **51**(3): 255–269.

[8] CHUDNOVSKY M, ROBERTSON N, SEYMOUR PD & THOMAS R. 2003. Progress on perfect graphs. *Mathematical Programming*, **97**(1): 405–422, July.

[9] CORNEIL DG, OLARIU S & STEWART L. 1998. The ultimate interval graph recognition algorithm? (extended abstract). In *Symposium on Discrete Algorithms*, pages 175–180.

[10] CORNUÉJOLS G, LIU X & VUŠKOVIĆ K. 2003. A polynomial algorithm for recognizing perfect graphs. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, **0**: 20.

[11] DÍAZ J, PETIT J & SERNA M. 2002. A survey of graph layout problems. *ACM Computing Surveys*, **34**(3): 313–356, September.

[12] DYSON RG & GREGORY AS. 1974. The cutting stock problem in the flat glass industry. *Operational Research Quarterly*, **25**(1): 41–53, Mar.

[13] FIORINI S. 2006. 0, 1/2-cuts and the linear ordering problem: Surfaces that define facets. *SIAM Journal on Discrete Mathematics*, **20**(4): 893–912.

[14] FOERSTER H & WAESCHER G. 1998. Simulated annealing for order spread minimization in sequencing cutting patterns. *European Journal of Operational Research*, **110**(2): 272–281, Oct. 16.

[15] FOMIN FV & GOLOVACH PA. 1998. Interval completion with the smallest max-degree. In o. S. J. Hromkovic, editor, *LNCS*, number 1517, pages 359–371. WG'98.

[16] FULKERSON DR & GROSS OA. 1965. Incidence matrices and interval graphs. *Pacific J. Math.*, **15**: 835–855.

[17] GAREY MR & JOHNSON DS. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

[18] GOLUMBIC MC. 1980. *Algorithmic graph theory and perfect graphs*. Academic Press, New York.

[19] GRÖTSCHEL M, JÜNGER M & REINELT G. 1985. Facets of the linear ordering polytope. *Mathematical Programming*, **33**(1): 43–60, Sept.

[20] HEGGERNES P. 2006. Minimal triangulations of graphs: A survey. *Discrete Mathematics*, **306**(3): 297–317.

[21] HEGGERNES P & MANCINI F. 2009. Minimal split completions. *Discrete Applied Mathematics*, **157**(12): 2659–2669, jun. Second Workshop on Graph Classes, Optimization, and Width Parameters.

[22] HEGGERNES P, MANCINI F & PAPADOPOULOS C. 2008. Minimal comparability completions of arbitrary graphs. *Discrete Applied Mathematics*, **156**(5): 705–718.

[23] KAPLAN H, SHAMIR R & TARJAN RE. 1999. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *Siam Journal On Computing*, **28**(5): 1906–1922, May.

[24] KINNERSLEY NG. 1992. The vertex separation number of a graph equals its path-width. *Information Processing Letters*, **42**(6): 345–350.

[25] KLOKS T, KRATSCH D & SPINRAD J. 1997. On treewidth and minimum fill-in of asteroidal triple-free graphs. *Theoretical Computer Science*, **175**(2): 309–335.

[26] LAPORTE G, GONZÁLEZ JJS & SEMET F. 2004. Exact algorithms for the job sequencing and tool switching problem. *IIE Transactions*, **36**: 37–45.

[27] LEKKERKERKER C & BOLAND J. 1962. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, **51**: 45–64.

[28] LINHARES A & YANASSE HH. 2002. Connections between cutting-pattern sequencing, VLSI design, and flexible machines. *Computers & Operations Research*, **29**(12): 1759–1772.

[29] LOPES IC. 2011. *Pattern sequencing models in cutting stock problems*. PhD thesis, Universidade do Minho, Portugal.

[30] MADSEN OB. 1988. 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.

[31] PANDA B & DAS S. 2003. A linear time recognition algorithm for proper interval graphs. *Information Processing Letters*, **87**(3): 153–161, August.

[32] PILEGGI G, MORABITO R & ARENALES M. 2005. Abordagens para otimização integrada dos problemas de geração e sequenciamento de padrões de corte: caso unidimensional. *Pesquisa Operacional*, **25**(3): 417–447.

[33] PINTO MJ. 2004. *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.

[34] REINELT G. 1993. A note on small linear-ordering polytopes. *Discrete and Computational Geometry*, **10**(1): 67–78, Dec.

[35] TANG SC & DENARDO EV. 1998. Models arising from a flexible manufacturing machine, part I: Minimization of the number of tool switches. *Operations Research*, **36**(5): 767–777, September-October.

[36] YANASSE HH. 1996. Minimization of open orders – polynomial algorithms for some special cases. *Pesquisa Operacional*, **16**(1): 1–26, June.

[37] YANASSE HH. 1997. On a pattern sequencing problem to minimize the maximum number of open stacks. *European Journal of Operational Research*, **100**: 454–463.

[38] YANASSE HH. 1997. A transformation for solving a pattern sequencing problem in the wood cut industry. *Pesquisa Operacional*, **17**(1): 57–70.

[39] YANASSE HH & LAMOSA MJP. 2007. An integrated cutting stock and sequencing problem. *European Journal of Operational Research*, **183**(3): 1353–1370.

[40] YANASSE HH & PINTO MJ. 2003. 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.

[41] YANASSE HH & SENNE ELF. 2010. 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.

[42] YUEN BJ. 1991. Heuristics for sequencing cutting patterns. *European Journal of Operations Research*, **55**(2): 183–190, November.

[43] YUEN BJ & RICHARDSON KV. 1995. Establishing the optimality of sequencing heuristics for cutting stock problems. *European Journal of Operations Research*, **84**: 590–598.