Minimization of Open Orders Using Interval Graphs

Isabel Cristina Lopes* J.M. Valério de Carvalho†

Abstract—**In this paper we address an order processing optimization problem known as the Minimization of Open Stacks Problem (MOSP). This problem consists in finding the best sequence for manufacturing the different products required by costumers, in a setting where only one product can be made at a time. The objective is to minimize the maximum number of incomplete orders from costumers that are being processed simultaneously. We present an integer programming model, based on the existence of a perfect elimination order in interval graphs, which finds an optimal sequence for the costumers orders. Among other economic advantages, manufacturing the products in this optimal sequence reduces the amount of space needed to store incomplete orders.**

Keywords: ***Integer programming, Interval graphs, Open orders minimization, MOSP, Pathwidth.***

1 Introduction

In this paper we address a problem in production planning, where the objective is to rapidly fulfill the costumers' orders. Each order may require several different products, but the manufacturer can only produce one product at a time. The problem is to know in which sequence the different products should be made, to minimize the maximum number of simultaneously open orders. This is also known in literature as the Minimization of Open Stacks Problem or MOSP.

Previous research on the MOSP demonstrates that any instance of the problem can be put in a graph with a vertex for each costumer [15] and a solution can be obtained by traversing the arcs of that graph, giving an ordering of the vertices and consequently of the products [3]. In our work, we also give an ordering of the vertices of the costumers' graph, not as a consequence of any arc traversing heuristics, but as a consequence of a modification of the costumers' graph by adding edges to obtain an interval graph that fits to the set of intervals of time in which the costumers' orders will be processed.

*Supported by FCT grant SFRH/BD/32151/2006 and IPP grant SFRH/BD/49914/2009. ESEIG - Polytechnic Institute of Porto: Rua D. Sancho I - 981, 4480-876 Vila do Conde, Portugal. Email: cristinalopes@eu.ipp.pt

†Department of Production and Systems - University of Minho, Portugal. Email: vc@dps.uminho.pt

Section 2 gives more explanations on the order processing optimization problem considered in this paper, along with a summary of previous results. Section 3 recalls some properties of interval graphs and the characterization that will be used. Section 4 presents an IP model for this problem based on interval graph completion and some computational results are discussed in section 5. This paper is an extended and revised work of the corresponding conference paper that appeared in [11].

2 Minimization of the maximum number of open orders

Consider the case of a manufacturer who has a number of orders from costumers to fulfill. Each order requires the making of different products, but only one product can be made at a time. A virtual stack is opened for each costumer when the first product of that order is processed. The stack is closed as soon as all products of that order have been manufactured, so the order is completed and ready to be sent to the costumer.

The setting costs usually prevent from switching between manufacturing different products, and transportation costs dissent from partial order deliveries to costumers. Also it is not advisable to have delays on deliveries, because, besides having resources tied up to stock and requiring more storage space for the incomplete orders, it will cause delays on the costumers' payments. The objective is to find an optimal sequence to manufacture the products in order to minimize the maximum number of simultaneously open orders.

2.1 Previous research

The MOSP has been proved by Linhares and Yanasse to be a NP-hard problem [10] that is equivalent to problems arising in cutting industries (like steel tubes, paper, flat glass, wooden panels), also in other fields such as VLSI Circuit Design (Gate Matrix Layout Problem and PLA Folding), and in classical problems from Graph Theory such as Pathwidth, Modified Cutwidth and Vertex Separation.

There have been several approaches to this problem. Yuen [19, 20] introduced the minimization of open stacks

problem and presented six heuristics for it. Because of their good practical behavior in processing time and simplicity, these heuristics were for a long time considered a classic reference.

Other meta-heuristic methods for the MOSP include tabu search and simulated annealing by Fink and Voss [7] and a 2-opt constructive genetic algorithm by Oliveira and Lorena [13].

As we are interested in using integer programming models, we may refer the reader to the work of Yanasse [18], who is the researcher that has published more papers concerning this problem. Using a graph representation of the problem, he has proved that there are polynomial algorithms that solve some special cases of the MOSP, he has presented some very good heuristics based on the properties of the corresponding graphs and furthermore he has developed integer programming models.

The polynomial algorithms that solve the problem work for the cases of the MOSP where the items being produced belong to at most two different costumers and the corresponding graph is a tree, a 1-tree, or a 0-1 common vertex polygon, which is a graph with no limitation on the number of polygons it contains but with the condition that any two polygons in the graph have at most one vertex in common [15].

Becceneri, Yanasse and Soma [3] have presented a combined method of heuristics and branch-and-bound for the MOSP. First, an heuristic of minimal cost node finds an order for the products by traversing the arcs of the graph using the least quantity of arcs to close a node. In this heuristic, all vertices are first ordered by non-decreasing order of the number of arcs needed to close that node and a search is made for adjacent nodes that have arcs not yet traversed. This heuristic generates a sequence of arcs and a maximum number of open stacks. Then an arc contraction heuristic is applied, which finds nodes with the same neighborhood and performs a contraction of the arcs with vertices which have the smallest indexes, calculating a lower bound for MOSP. Finally, an exact method executes a branch-and-bound approach to reach for the optimal solution.

In [16], Yanasse demonstrates two propositions that make the IP models originally proposed by Tang and Denardo [14] for a similar problem, known as Minimization of Tool Switches Problem (MTSP), suitable for modeling the MOSP. Although these two problems are not equivalent in the general case, they are equivalent when the number of tool slots in the machine is equal to the optimal solution value of MOSP. Later, Yanasse and Lamosa [17] adapted for the MOSP another MTSP integer model formulated previously by Laporte [9].

Baptiste [2] submitted to the Constraint Modeling Challenge in 2005 a MIP formulation for MOSP that uses binary variables to assign each product to a sequence position and to indicate if each costumer's order starts before, ends after, or is in process at a given position in the sequence. The constraints of the model refer to the start and end variables, while linking the products and the costumers.

Among other methods that came out from the Constraint Modeling Challenge, we highlight a very successful approach to the MOSP using dynamic programming by Banda and Stuckey [1] that was the winning paper.

2.2 The MOSP in a graph

An instance of this problem can be put in a graph where each costumer's order is represented by a vertex, and two vertices are adjacent iff the corresponding orders have common products.

A product that is required by k different costumers will correspond to a clique of size k in the MOSP graph, because the k vertices must be connected to each other.

As an example, we will see an instance of the MOSP with five costumers and eight products taken from [6, p.322].

Table 1: An instance of the MOSP

Products:	1	2	3	4	5	6	7	8
Costumer 1	X	X				X	X	
Costumer 2				X	X			
Costumer 3			X				X	
Costumer 4		X		X				X
Costumer 5			X		X	X		

This instance originates a graph with 5 vertices, one corresponding to each costumer, and with edges between the vertices/costumers that have ordered the same product.

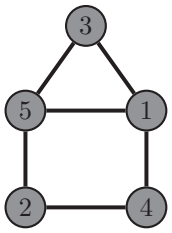

Figure 1: Graph of the instance in Table 1

Notice that, for example, costumer 2 ordered only products 4 and 5, so vertices 2 and 4 are connected because product 4 is required by costumers 2 and 4, and 2 and 5 are connected because of product 5.

To optimize the processing of the costumers orders, it is convenient to find the best sequence to manufacture the products. Considering that the products do not appear

explicitly in the graph, how will we find that sequence for products? We will focus on finding a sequence to process the costumers' orders, and the sequence for the products will come out as a consequence.

A feasible solution of this problem corresponds to a sequence of arcs in the graph. Traversing that sequence of arcs, a stack for costumer j is open when it is the first time that an arc with an end in j is traversed and the stack is closed when all arcs with an end in j have been traversed. Going along the sequence of arcs, and the corresponding ordering of the opening of the stacks, we sequence a product P_i of the original problem when all nodes corresponding to all costumers that required P_i have been opened.

There are some situations that, because of their simplicity, can be removed from the original problem while solving it and inserted later in the solution. Costumers who ordered just one product will appear in the graph as isolated vertices if that product is not required by any other costumer. In this case, that product can be the first or last in the sequence, and it will open and close a stack without any other stacks open at that same time, so it does not increase the maximum number of simultaneously open stacks.

If a product is required by only one costumer, who has also ordered other products, then that product should be manufactured just before the first of the products of that costumer's order, and the number of simultaneously open stacks will not increase [16].

In this example this happens with products 1 and 8. This instance can be reduced to only six relevant products (2, 3, 4, 5, 6 and 7) generating the same graph. Product 1 can be sequenced in the solution just before the first of the products 2,6,7, and product 8 just before the first of the products 2 and 4, without increasing the MOSP number.

Table 2: Solution of the instance in Table 1

Products:	1	6	3	7	8	2	4	5
Costumer 1	X	X		X		X		
Costumer 2							X	X
Costumer 3			X	X				
Costumer 4					X	X	X	
Costumer 5		X	X					X

For the example above, a possible solution is the sequence of vertices 1-5-3-4-2 that corresponds to the opening of the costumers' orders and consequently the sequence to manufacture the products would be 1-6-3-7-8-2-4-5.

As there are some orders that are not simultaneously open at any time, like 3 and 4, or 1 and 2, those stacks can use the same stack space, hence this sequence of products gives a maximum of three simultaneously open stacks, that is the optimum for this instance.

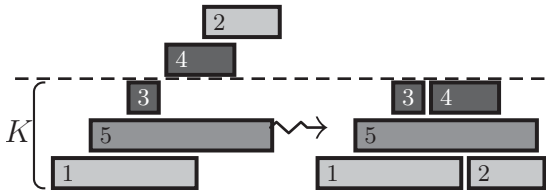

Figure 2: Non simultaneous orders can share stack space

This means that it is natural to associate the lifetime of a costumers' order in the solution with intervals of time measured not in minutes or hours but measured in terms of the number of different products in the manufacturing sequence. We have seen that we can start solving a MOSP problem with a graph, and that in the solution of the problem we can consider an interval for the time that each stack is open. We will see that an interval graph can be associated to the set of intervals in the solution and we will also use some properties of interval graphs to find the solution of MOSP instances.

3 Interval Graphs

In this section we recall Golumbic's [8] definition of an interval graph and some other concepts and theorems that will be used in our model.

Definition 3.1. An undirected graph G is called an *interval graph* if 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 non-empty intersection. I is called an *interval representation* for G.

This definition is useful, because 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.

Definition 3.2. A *chordal graph* (sometimes called *triangulated graph*) is a graph where every simple k-cycle, with $k > 3$, has a chord.

Definition 3.3. A *comparability graph* is an undirected graph which is transitively orientable, i.e., each edge can be assigned a one-way direction in such a way that the resulting oriented graph (V, F) satisfies:

$$[ab] \in F \wedge [bc] \in F \Rightarrow [ac] \in F \quad \forall a, b, c \in V$$

One theorem that characterizes interval graphs is the following (Gilmore and Hoffman, 1964):

Theorem 3.4. *Let G be an undirected graph. The following are equivalent:*

- *G is an interval graph*
- *G is chordal and its complement is a comparability graph*
- *The maximal cliques of G can be linearly ordered such that, for every vertex x of G, the maximal cliques containing x occur consecutively.*

This ordering of the maximal cliques in the interval graph will allow us to set an ordering of the vertices, helped by the following theorems from [4].

Theorem 3.5. *An interval graph G has an interval representation such that all endpoints of intervals are distinct integers.*

By using the left endpoints of the intervals, we can define a natural ordering of the vertices of an interval graph with a bijective function $\varphi : V \to \{1, ..., N\}$.

Definitions 3.6. In an interval graph $G = (V, E)$, we say that vertex i *precedes* vertex j and denote by $i \prec j$ if $\varphi(i) < \varphi(j)$, i.e. if the left endpoint of the interval correspondent to vertex i is before the left endpoint of the interval correspondent to vertex j.

We denote the set of predecessors of a vertex by $Pred(i) = \{j \in Adj(i) : \varphi(j) < \varphi(i)\}$ and the set of successors by $Succ(i) = \{j \in Adj(i) : \varphi(j) > \varphi(i)\}$.

This ordering of the vertices can also lead to edge directions, directing the edge $[ij]$ if $i \prec j$.

Definition 3.7. A vertex v_i of a graph $G = (V, E)$ is a *simplicial vertex* if its adjacency set $Adj(v_i) = \{v_j \in V : [v_i v_j] \in E\}$ is a clique.

It is known that simplicial vertices appear in all chordal graphs (Dirac, 1961):

Theorem 3.8. *Every chordal graph has a simplicial vertex. If it is not a complete graph, then it has two simplicial vertices that are not adjacent.*

In a chordal graph, any simplicial vertex can start a perfect elimination order:

Definition 3.9. A *perfect vertex elimination order* is a sequence $\sigma = [v_1, v_2, ..., v_n]$ of the vertices of the graph in which each set of predecessors $Pred(v_i)$ is a clique.

In an interval graph, for any vertex v_i represented by an interval that starts at s_i, $Pred(v_i)$ is the set of all vertices with intervals that start before s_i and end after s_i; as these intervals overlap at s_i, $Pred(v_i)$ is a clique. This vertex order is a perfect elimination order.

This conjugates the vertex order defined by the left endpoints of the intervals with the sequence of cliques that will appear in the interval graph of the solution of a MOSP problem.

We are interested in defining an ordering of the vertices that corresponds to a perfect elimination order. It is known that an interval graph H is chordal and it has at least two simplicial vertices where a perfect vertex elimination order can be started. Locating a simplicial vertex and eliminating it will create another simplicial vertex and its subsequent elimination and so on. Also because H is an interval graph, its maximal cliques can be linearly ordered in such a way that, for every vertex v_i in H, the maximal cliques containing v_i occur consecutively. The perfect elimination order sets the order of the intervals, because if we follow the order of the eliminated vertices we have the order in which intervals must start.

An alternative characterization of interval graphs given by Olariu [12] uses this linear ordering of the vertices as well.

Theorem 3.10. *$G = (V, E)$ is an interval graph if and only if there exists a linear ordering $\varphi : V \to \{1, ..., N\}$ such that $\forall i, j, k \in V : \varphi(i) < \varphi(j) < \varphi(k)$ we have $[ik] \in E \Rightarrow [ij] \in E$.*

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

4 An IP model

Given an instance of the problem, we first build a graph $G = (V, E)$, associating each order to a vertex and creating an arc joining vertex i and j if and only if order i has a product in common with order j. 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 order to the interval of time in which the order is being processed, we can use the graph to model what intervals should occur simultaneously and what intervals should precede others.

According to the sequence in which the products are made, there may be more or less open orders simultaneously. Each arc of the future interval graph means that, for a period of time, the two orders (the respective vertices of the arc) will remain both open. The initial graph contains only the arcs that must be there, in any possible sequence in which the products can be made.

The rest of the arcs that are added later to the graph will differ according to the sequence of the products. It is the choice of these arcs that defines which are the other simultaneously open orders. Our model consists in finding out which edges F 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 orders.

4.1 The decision variables

We set an ordering for starting the costumers' orders by assigning a number to each one of them, with a bijective function $\varphi : V \to \{1, ..., N\}$. This linear ordering of the vertices is set by the decision variables x_{ij}:

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

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

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

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

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

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

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 orders from costumers i and j will never be processed at the same time, so they can share the same stack space, as seen in Figure 3.

Figure 3: Interval i opens and closes before j starts

To explain the relations between the intervals horizontally, we will also use a set of variables s, based on the asymmetric representatives formulation for the vertex coloring problem by Campelo et al [5].

Definition 4.1. Given a graph $G = (V, E)$, a linear ordering of V, and a coloring of V such that adjacent vertices have different colors, we say that a vertex $i \in V$ is a *representative* if i precedes all other vertices with the same color of i. We say that vertex $i \in V$ *represents* vertex $j \in V$ if i and j have the same color and if i is a representative.

In the graph correspondent to Figure 3, vertex i would represent vertex j. If we assign colors to the vertices of the desired interval graph, such that no two adjacent vertices have the same color, we can count the maximum number of simultaneously open orders by counting the minimum number of different colors needed, because simultaneously open orders will get different colors, and orders that do not overlap can have the same color. The variables that we will use are:

$$s_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ represents vertex } j \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j \in V : [ij] \notin E$$

Note that if $i \in V$ is a representative then $s_{ii} = 1$.

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

4.2 The main restrictions of the model

We will now study the relations between the binary integer variables x, y, s and the integer variable K to build the restrictions for our model.

4.2.1 Linear ordering of the vertices

The linear ordering of the vertices brings two basic inequalities. The first one states that either vertex i precedes vertex j or vice-versa. This is expressed by:

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

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

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

An important remark upon the variables y_{ij} is that they establish the precedences between the closing and opening of the intervals. As one of the conditions for the variable y_{ij} to be equal to 1 is that vertex i precedes vertex j, equal to say that $x_{ij} = 1$, then we must have:

$$y_{ij} \leq x_{ij} \quad \forall i, j \in V, i \neq j, [ij] \notin E \tag{3}$$

4.2.2 Obtaining an interval graph

To guarantee that the graph $H = (V, E \cup F)$ is an interval graph, we use in the model the characterization given in Theorem 3.10. We will consider three different vertices $i, j, k \in V$ and analyze in what circumstances the arcs $[ik]$ and $[ij]$ exist or have to be added. Let us separate in two cases: the arc $[ik] \in E$ and $[ik] \notin E$.

For the first case, the arc $[ik] \in E$, let us suppose that arc $[ij] \notin E$, otherwise H is already 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 $[ij] \in F$, i.e., $x_{kj} = 0 \Rightarrow y_{ij} = 0$ which can be stated by the linear inequality

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

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 (3) we have $y_{ij} = 0$ and the inequality is also valid.

In the example shown in Figure 1, vertices 5, 4 and 2 fall in this case, so the formulation of the problem will contain the inequality:

$$y_{54} \leq x_{24}$$

The ordering of the vertices in the solution 1-5-3-4-2 makes $x_{24} = 0$ and this inequality forces $y_{54} = 0$, meaning that the arc [54] is added to the graph, indicating that intervals 4 and 5 overlap in the solution, i.e., the orders from costumers 4 and 5 will be processed simultaneously at some point.

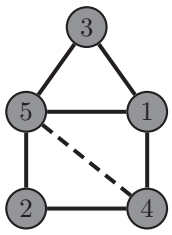

Figure 4: Interval graph of the solution in Table 2

Now let us consider the second case where $[ik] \notin E$ and also consider $[ij] \notin E$, because otherwise the result was guaranteed. Start by 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 we must also add the arc $[ij]$, for the graph to be an interval graph, as in Figure 5.

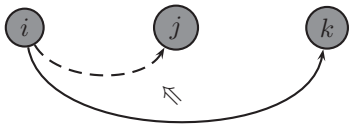

Figure 5: Olariu's characterization of interval graphs

Using the correspondent variables y this is to say that

$$x_{kj} = 0 \wedge y_{ik} = 0 \Rightarrow y_{ij} = 0$$

and this can be represented by the inequality

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

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. If $k \prec j$, the adding of the arc $[ik]$ to the graph does not force to add the arc $[ij]$. In the remaining three cases, $j \prec i$ forces $y_{ij} = 0$ because of (3).

Again, in the example of Figure 1, vertices 3, 4 and 2 fall in this second case, so the formulation of the problem will contain the inequality:

$$y_{34} \leq x_{24} + y_{32}$$

As $x_{24} = 0$ because $4 \prec 2$, if the arc [32] were added, we would have $y_{32} = 0$, and this inequality would set $y_{34} = 0$. As $3 \prec 4$, it would say that the arc [34] would be added to the graph. In the optimal solution 1-5-3-4-2 the arc [32] is not added, and therefore the variable is $y_{32} = 1$, leaving the variable y_{34} free to be 0 or 1.

The model can also be strengthened with the following three inequalities that can be proved simply by observing that in an interval graph both variables on the left are not allowed to be simultaneously equal to one without contradicting Theorem 3.10.

$$y_{ij} + y_{ki} \leq 1 \quad \forall i, j, k \in V \text{ with } [ij], [ik] \notin E, [jk] \in E \tag{6}$$

$$y_{ij} + y_{jk} \leq 1 \quad \forall i, j, k \in V \text{ with } [ij], [jk] \notin E, [ik] \in E \tag{7}$$

$$y_{ij} + y_{lk} \leq 1 \quad \forall i, j, k, l \in V \text{ with } [ij], [kl] \notin E, [jl], [ik] \in E \tag{8}$$

4.3 A lower bound

By Theorem 3.4, at each node of interval graph H, except at the first K ones, there is an interval that begins and one that ends, so that all the cliques in the sequence of the perfect elimination order are maximal, except the first K ones. And because every appearance of a vertex in these cliques must be consecutive and there must be N cliques in the sequence, at each instant only one vertex changes in the cliques. This change corresponds to the closing of an interval and the beginning of another.

The precedences of the opening and closing of the intervals are declared by the variables y_{ij}. For every vertex

j, the sum $\sum_{i=1}^{N} y_{ij}$ counts how many intervals must finish before interval j starts and the sum $\sum_{j=1}^{N} y_{ij}$ counts how many intervals will start after i finishes. The vertex that finishes first is the one that has the greatest $\sum_{j=1}^{N} y_{ij}$.

If we sum up the variables x_{ij} we will find the position of each vertex in the sequence of vertices. 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. The beginning of an interval j happens at instant $\sum_{i=1}^{N} x_{ij} + 1$. It is the number of intervals that have started before j plus the interval j itself. So the number of intervals that are open at that instant is $\sum_{i=1}^{N} x_{ij} + 1 - \sum_{i=1}^{N} y_{ij}$ because we need to subtract the number of intervals that have already been closed before that instant. This leads to the main lower bound for the MOSP:

$$\sum_{\substack{i=1 \\ i \neq j}}^{N} x_{ij} - \sum_{\substack{i=1 \\ [ij] \notin E}}^{N} y_{ij} \; + 1 \leq K \qquad \forall j = 1, ..., N \qquad (9)$$

If one puts each interval in a line, as in Figure 2, the number of lines that we have open when an interval starts is a lower bound for the maximum number of open stacks.

In the example presented before, when interval 5 starts there are two open stacks,

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

which is a lower bound for the MOSP (which is three). The same inequality correspondent to the moment that interval 2 starts gives a better lower bound:

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

4.4 Strengthening the model

The inequalities that we have seen so far are sufficient to have a valid model and to guarantee that the solution will be an interval graph. But we can reinforce our model by adding some constraints related to additional properties of interval graphs.

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,

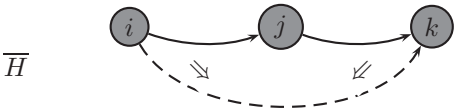

Figure 6: $\overline{H}$ must be transitively orientable

with an orientation $i \prec j$ and $j \prec k$, then if the arc $[ik]$ exists, it must be oriented as in $i \prec k$.

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 graph:

$$y_{ij} + y_{jk} - 1 \leq y_{ik} \quad \forall i, j, k \in V, [ij], [jk], [ik] \notin E \quad (10)$$

4.4.1 Chords in k-cycles

Another way to reinforce the model is to reduce the number of arcs that are added to the original graph. If the graph G is completed to become an interval graph, it has to be chordal, so in every k-cycle for $k \geq 4$ sufficient chords must be added. In a 4-cycle defined by the ordered vertices $ijkl$, we need to add at least one of the arcs in the diagonal. In the complement graph there will remain the

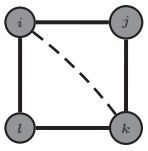

Figure 7: A 4-cycle must have a chord

other diagonal arc, or none, whose existence is flagged by the variables y. The need to add one of the arcs $[ik]$, $[ki]$, $[jl]$ or $[lj]$ can be expressed by the restriction:

$$y_{ik} + y_{ki} + y_{jl} + y_{lj} \leq 1 \qquad \forall [ik], [jl] \notin E, [ij], [jk], [kl], [li] \in E \qquad (11)$$

A 5-cycle needs at least two chords, but not any chord will do, because the two chords must share a vertex, as in Figure 8 (a).

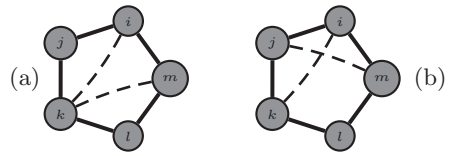

Figure 8: 5-cycles with two chords: (a) is chordal, (b) is not chordal

In a 5-cycle defined by the ordered vertices $ijklm$, the restriction should be:

$$y_{il} + y_{li} + y_{ik} + y_{ki} + y_{jl} + y_{lj} + y_{jm} + y_{mj} + y_{mk} + y_{km} \leq 3 \quad \forall [ik], [il], [jl], [jm], [km] \notin E, [ij], [jk], [kl], [lm], [mi] \in E \tag{12}$$

The number 3 comes from the fact that we need at least 2 chords from the possible 5 chords, so there must be at most 3 chords left in the complement graph, causing the defined variables y to sum at most 3.

4.5 Coloring the vertices

The value of the optimum of the MOSP is equal to the size of the biggest clique in the solution graph $\omega(H)$ and, because interval graphs are perfect graphs, it is equal to the chromatic number of the graph $\chi(H)$, which is the number of colors needed to assign to the vertices of the graph such that there are no two adjacent vertices of the same color.

In the representatives formulation for the vertex coloring problem [5], the number of different colors is counted by the number of representatives vertices, i.e. $s_{ii} = 1$. Hence

$$\sum_{i=1}^{N} s_{ii} = K \tag{13}$$

All N vertices must have representatives

$$\sum_{\substack{i=1 \\ [ij] \notin E}}^{N} \sum_{\substack{j=1 \\ [ij] \notin E}}^{N} s_{ij} = N \tag{14}$$

but each vertex has only one representative:

$$\sum_{\substack{i=1 \\ [ij] \notin E}}^{N} s_{ij} = 1 \quad \forall j = 1, ..., N \tag{15}$$

A vertex i represents a vertex j ($s_{ij} = 1$) only if i and j share the same stack ($y_{ij} = 1$):

$$s_{ij} \leq y_{ij} \quad \forall i, j = 1, ..., N \text{ with } [ij] \notin E \tag{16}$$

and only if i is a representative

$$s_{ij} \leq s_{ii} \quad \forall i, j = 1, ..., N \text{ with } [ij] \notin E \tag{17}$$

In the previous example (Figure 4), vertex 3 can only represent vertices 2 or 4, which is stated by these inequalities because variables y_{32} and y_{34} are the only non-null variables of type y_{3j}, and only when $s_{33} = 1$, which means that interval 3 is the "leader"of a new stack.

Definition 4.2. For a graph $G = (V, E)$, the anti-neighborhood of a set of vertices $U \subseteq V$ is the set of vertices that are not adjacent to any vertex in U:

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

A vertex in the anti-neighborhood of each clique can represent only one vertex of the clique.

For a vertex i in the anti-neighborhood of a 2-clique $\{jk\}$ we have the inequality

$$s_{ij} + s_{ik} \leq s_{ii} \quad \forall i, j, k = 1, ..., N \text{ with } j < k, [ij], [ik] \notin E, [jk] \in E \tag{18}$$

and for a 3-clique $\{jkl\}$

$$s_{ij} + s_{ik} + s_{il} \leq s_{ii} \quad \forall i, j, k, l = 1, ..., N \text{ with } j < k < l, [ij], [ik], [il] \notin E, [jk], [kl], [lj] \in E \tag{19}$$

and for a 4-clique $\{jklm\}$

$$s_{ij} + s_{ik} + s_{il} + s_{im} \leq s_{ii} \quad \forall i, j, k, l, m = 1, ..., N \text{ with } j < k, j < l, k < m, [ij], [ik], [il], [im] \notin E, [jk], [jl], [jm], [kl], [km], [lm] \in E \tag{20}$$

In our example, the 2-clique $\{2, 4\}$ originates the inequality

$$s_{32} + s_{34} \leq s_{33}$$

which means that the vertex 3, which is in the anti-neighborhood of that clique, can only represent vertex 2 or vertex 4 and only if 3 is a representative, i.e., order from costumer 3 can use the same stack space than order from costumer 2 or costumer 4 but not both, and only if costumer 3 is the "leader"of that stack (see Figure 2).

4.6 The formulation of the model

Given an instance of a MOSP problem, let the graph $G = (V; E)$ be the associated graph and $|V| = N$. Considering the variables and the inequalities explained previously, our new mathematical formulation for the MOSP is:

Minimize K
Subject to: (1) to (20) and

$$x_{ij} \in \{0, 1\} \quad \forall i, j = 1, ..., N \text{ with } i \neq j \tag{21}$$

$$y_{ij} \in \{0, 1\} \quad \forall i, j = 1, ..., N \text{ with } i \neq j, [ij] \notin E \tag{22}$$

$$s_{ij} \in \{0, 1\} \quad \forall i, j = 1, ..., N \text{ with } [ij] \notin E \tag{23}$$

$$K \in \mathbb{N} \tag{24}$$

5 Computational tests

The model was tested on some instances of the Constraint Modeling Challenge 2005, available at:

http://www.cs.st-andrews.ac.uk/∼ipg/challenge/instances.html

Computational tests were performed with ILOG OPL Development Studio 5.5 on an Intel®Core2 Duo T7200@2.00GHz 0.99GB RAM. 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 are recorded in Table 3.

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

Table 3: Computational results of the IP model for the minimization of open orders

Instance	No. clients (Nodes)	Best Objective value	Best LB	Gap	Runtime (s)	Nodes in search tree
Harvey wbo_10_10_1	10	**3**	3	0%	11,06	0
Harvey wbo_10_20_1	10	**5**	5	0%	5,50	209
Harvey wbop_10_20_10	10	**5**	5	0%	3,26	44
Simonis Problem 10_20_150	10	**9**	9	0%	0,75	0
Wilson nwrsSmaller4_1	10	**3**	3	0%	0,75	0
Wilson nwrsSmaller4_2	10	**4**	4	0%	0,54	0
Harvey wbo_15_15_1	15	**3**	3	0%	17,07	10
Harvey wbo_15_30_1	15	**4**	4	0%	3,65	0
Harvey wbop_15_30_15	15	**11**	11	0%	997,04	21433
Harvey wbp_15_15_35	15	**14**	14	0%	1,53	0
Simonis Problem 15_15_100	15	**11**	11	0%	177,17	5056
Wilson nwrsSmaller4_3	15	**7**	7	0%	1,25	0
Miller	20	**13**	13	0%	18731,73	14245
Shaw Instance_1	20	**14**	14	0%	45323,62	167846
Shaw Instance_15	20	**14**	13	7%	57173,01	319000
Shaw Instance_2	20	**12**	12	0%	10066,76	19741
Simonis Problem 20_10_1	20	**9**	7	22%	7993,87	8.097
Simonis Problem 20_20_100	20	**19**	19	0%	3,50	0
Wilson nrwsLarger4_2	20	**12**	12	0%	14,29	5
Wilson SP_1	25	**9**	8	11%	378,10	2934
Harvey wbo_30_10_1	30	**10**	7	30%	32536,84	631
Harvey wbo_30_30_1	30	**4**	3	25%	1907,06	0
Harvey wbop_30_10_1	30	**18**	10	44%	900,79	0
Simonis Problem 30_30_1	30	**21**	13	38%	2990,03	54

References

[1] M. G. Banda and P. J. Stuckey, "Dynamic programming to minimize the maximum number of open stacks," *Informs Journal On Computing*, vol. 19, pp. 607–617, 2007.

[2] P. Baptiste, "Simple MIP formulations to minimize the maximum number of open stacks," in *Constraint Modeling Challenge*. Edinburgh, Scotland: IJCAI 2005, 31 July 2005, pp. 9–13.

[3] J. C. Becceneri, H. H. Yanasse, and N. Y. Soma, "A method for solving the minimization of the maximum number of open stacks problem within a cutting process," *Computers & Operations Research*, vol. 31, no. 14, pp. 2315–2332, 2004.

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

[5] M. Campêlo, V. A. Campos, and R. C. Corrêa, "On the asymmetric representatives formulation for the vertex coloring problem," *Discrete Applied Mathematics*, vol. 156, no. 7, pp. 1097–1111, 2008.

[6] J. Díaz, J. Petit, and M. Serna, "A survey of graph layout problems," *ACM Computing Surveys*, vol. 34, no. 3, pp. 313–356, Sep 2002.

[7] A. Fink and S. Voss, "Applications of modern heuristic search methods to pattern sequencing problems," *Computers & Operations Research*, vol. 26, no. 1, pp. 17–34, 1999.

[8] M. C. Golumbic, *Algorithmic graph theory and perfect graphs*. New York: Academic Press, 1980.

[9] G. Laporte, J. J. S. Gonzalez, and F. Semet, "Exact algorithms for the job sequencing and tool switching problem," *IIE Transactions*, vol. 36, pp. 37–45, 2004.

[10] A. Linhares and H. H. Yanasse, "Connections between cutting-pattern sequencing, VLSI design, and flexible machines," *Computers & Operations Research*, vol. 29, no. 12, pp. 1759–1772, 2002.

[11] I. C. Lopes and J. M. V. Carvalho, "Using interval graphs in an order processing optimization problem," in *Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2010*. London, UK: WCE2010, 30 June - 2 July 2010, pp. 1722–1728.

[12] S. Olariu, "An optimal greedy heuristic to color interval graphs," *Information Processing Letters*, vol. 37, no. 1, pp. 21–25, 1991.

[13] A. C. Oliveira and L. A. Lorena, *2-Opt Population Training for Minimization of Open Stack Problem*, ser. Lecture Notes in Artificial Intelligence. Springer, 2002, vol. 2507, ch. Advances in Artificial Intelligence, pp. 313–323.

[14] S. C. Tang and E. V. Denardo, "Models arising from a flexible manufacturing machine, part I: Minimization of the number of tool switches," *Operations Research*, vol. 36, no. 5, pp. 767–777, Sep-Oct 1988.

[15] H. Yanasse, "Minimization of open orders - polynomial algorithms for some special cases," *Pesquisa Operacional*, vol. 16, no. 1, pp. 1–26, Jun 1996.

[16] H. H. Yanasse, "On a pattern sequencing problem to minimize the maximum number of open stacks," *European Journal of Operational Research*, vol. 100, pp. 454–463, 1997.

[17] H. H. Yanasse and M. J. P. Lamosa, "An integrated cutting stock and sequencing problem," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1353–1370, 2007.

[18] H. H. Yanasse and E. L. F. Senne, "The minimization of open stacks problem: A review of some properties and their use in pre-processing operations," *European Journal of Operational Research*, vol. 203, no. 3, pp. 559 – 567, 2010.

[19] B. J. Yuen, "Heuristics for sequencing cutting patterns," *European Journal of Operations Research*, vol. 55, no. 2, pp. 183–190, Nov 1991.

[20] B. J. Yuen and K. V. Richardson, "Establishing the optimality of sequencing heuristics for cutting stock problems," *European Journal of Operations Research*, vol. 84, pp. 590–598, 1995.