An integer programming framework for sequencing cutting patterns based on interval graph completion

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

* ESEIG, Polytechnic Institute of Porto
Rua D.Sancho I, 981, Vila do Conde
cristinalopes@eu.ipp.pt

† Department of Production and Systems, University of Minho
Campus de Gualtar, Braga
vc@dps.uminho.pt

ABSTRACT

We derived a framework in integer programming, based on the properties of a linear ordering of the vertices in interval graphs, that acts as an edge completion model for obtaining interval graphs. This model can be applied to problems of sequencing cutting patterns, namely the minimization of open stacks problem (MOSP). By making small modifications in the objective function and using only some of the inequalities, the MOSP model is applied to another pattern sequencing problem that aims to minimize, not only the number of stacks, but also the order spread (the minimization of the stack occupation problem), and the model is tested.

Keywords: Integer programming, Interval graphs, Sequencing cutting patterns

1. INTRODUCTION

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

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

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

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

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

The minimization of open stacks problem is known to have tight relations with problems in graph theory such as treewidth, vertex separation and the profile of a matrix. In studying these problems, we found a type of graphs called interval graphs that can play an important role in this work.

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. [1]

These graphs can be used to describe a solution of the pattern sequencing problems, by modeling the duration of the intervals in time in which the same piece type is being cut. Using several properties of this type of graphs we will see that it is possible to derive a general framework that can be used to model the minimization of open stacks problem and to model many related problems.

MOSP is modeled as an interval graph completion problem. An initial integer programming model was derived, using the addition of arcs to the graph and the properties of interval graphs to achieve a solution, and based on the following characterization of interval graphs by Olariu:

A graph $G = (V,E)$ is an interval graph if and only if there exists a linear ordering $\varphi : V \to \{1,...,N\}$ such that $\forall i,j,k \in V : \varphi(i) < \varphi(j) < \varphi(k)$ we have $[ik] \in E \Rightarrow [ij] \in E$. [2]

The model is strengthened with inequalities derived from the relationship between the chromatic number of a graph and the number of intersecting intervals.

The MOSP model is applied to different problems. By making small modifications in the objective function and using only some of the inequalities, the MOSP model is applied to the minimum interval graph completion problem. Another pattern sequencing

problem that aims to minimize, not only the number of stacks, but also the order spread (the minimization of the stack occupation problem) is considered, and the model is tested.

There is also another pattern sequencing problem called the Minimization of Tool Switches (MTSP) which is addressed with this framework, using the similarities between this problem and the MOSP, but for this problem the model has a limited use.

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

2. MODELING THE MINIMIZATION OF OPEN STACKS

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

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

MOSP has been proved to be a NP-hard problem [3].

As suggested in [4], an instance of the MOSP can be associated with a graph having a vertex for each item that is cut and an edge between two vertices if the corresponding items are present in the same cutting pattern.

To optimize the number of stacks, it is convenient to find the best sequence to process the cutting patterns. Considering that the patterns do not appear explicitly in the MOSP graph constructed in this way, how will we find that sequence for the cutting patterns? We will focus on finding a sequence to open the stacks, rather than on sequencing the cutting patterns. That is not a problem, because it is possible to take a solution for the ordering of the vertices of the graph and construct a sequence for the corresponding cutting patterns [5].

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

2.1. The variables

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

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

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 items in the current instance. If $x_{ij} = 1$ then item i starts being cut before the item j is, even though the corresponding stacks may overlap or not, i.e., in spite of having an arc between the two vertices or not.

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

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

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

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

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

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

To explain the relations between the intervals horizontally, we will add an extra set of variables z, based on the asymmetric representatives formulation for the vertex coloring problem by Campêlo et al. [6]. 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.

If we assign colors to the vertices of the desired interval graph, such that no two adjacent vertices have the same color, we can count the maximum number of simultaneously open stacks by counting the minimum number of different colors needed, because simultaneously open stacks will get different colors, and stacks that do not overlap can have the same color.

The variables that we will use are:

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

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

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

2.2. The main model

Using this variables we present the following integer programming model for the MOSP:

Minimize K

Subject to:

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

$$y_{ij} - x_{ij} \leq 0 \qquad \forall i, j = 1, ..., N, i < j, [ij] \notin E \tag{2}$$

$$y_{ij} + x_{ji} \leq 1 \qquad \forall i, j = 1, ..., N, j < i, [ij] \notin E \tag{3}$$

$$y_{ij} - x_{kj} \leq 0 \qquad \forall i, j, k = 1, ..., N, k < j, [ij] \notin E, [ik] \in E \tag{4}$$

$$y_{ij} + x_{jk} \leq 1 \qquad \forall i, j, k = 1, ..., N, j < k, [ij] \notin E, [ik] \in E \tag{5}$$

$$0 \leq y_{ik} - y_{ij} + x_{kj} \leq 1 \qquad \forall i, j, k = 1, ..., N, k < j, [ij], [ik] \notin E \tag{6}$$

$$0 \leq y_{ij} - y_{ik} + x_{jk} \leq 1 \qquad \forall i, j, k = 1, ..., N, j < k, [ij], [ik] \notin E \tag{7}$$

$$\sum_{i=1}^{j-1} x_{ij} + \sum_{i=j+1}^{N} (1 - x_{ji}) - \sum_{\substack{i=1 \\ [ij] \notin E}}^{N} y_{ij} + 1 \leq K \qquad \forall j = 1, ..., N \tag{8}$$

$$y_{ij} + y_{ki} \leq 1 \qquad \forall i, j, k = 1, ..., N \text{ with } [ij], [ik] \notin E, [jk] \in E \tag{9}$$

$$y_{ij} + y_{jk} \leq 1 \qquad \forall i, j, k = 1, ..., N \text{ with } [ij], [jk] \notin E, [ik] \in E \tag{10}$$

$$y_{ij} + y_{lk} \leq 1 \qquad \forall i, j, k, l = 1, ..., N \text{ with } [ij], [kl] \notin E, [jl], [ik] \in E \tag{11}$$

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

$$y_{ik} + y_{ki} + y_{jl} + y_{lj} \leq 1 \qquad \begin{array}{l} \forall i, j, k, l = 1, ..., N \text{ with } i \neq j \neq k \neq l, \\ [ik], [jl] \notin E, [ij], [jk], [kl], [li] \in E \end{array} \tag{13}$$

$$\begin{array}{l} y_{il} + y_{li} + y_{ik} + y_{ki} + y_{jl} + \\ + y_{lj} + y_{jm} + y_{mj} + y_{mk} + y_{km} \end{array} \leq 3 \qquad \begin{array}{l} \forall i, j, k, l, m = 1, ..., N \text{ with } i \neq j \neq k \neq l \neq m, \\ [ik], [il], [jl], [jm], [km] \notin E, [ij], [jk], [kl], [lm], [mi] \in E \end{array} \tag{14}$$

$$\sum_{i=1}^{N} z_{ii} = K \tag{15}$$

$$\sum_{\substack{i=1 \\ [ij] \notin E}}^{N} \sum_{\substack{j=1 \\ [ij] \notin E}}^{N} z_{ij} = N \tag{16}$$

$$\sum_{\substack{i=1 \\ [ij] \notin E}}^{N} z_{ij} = 1 \qquad \forall j = 1, ..., N \tag{17}$$

$$z_{ij} \leq y_{ij} \qquad \forall i, j = 1, ..., N \text{ with } [ij] \notin E \tag{18}$$

$$z_{ij} + z_{ik} - y_{jk} - y_{kj} \leq 1 \qquad \forall i, j, k = 1, ..., N \text{ with } [ij], [ik], [jk] \notin E \tag{19}$$

$$z_{ij} \leq z_{ii} \qquad \forall i, j = 1, ..., N \text{ with } [ij] \notin E \tag{20}$$

$$z_{ij} + z_{ik} \leq z_{ii} \qquad \forall i, j, k = 1, ..., N \text{ with } j < k, [ij], [ik] \notin E, [jk] \in E \tag{21}$$

$$z_{ij} + z_{ik} + z_{il} \leq z_{ii} \qquad \begin{array}{l} \forall i, j, k, l = 1, ..., N \text{ with } j < k < l, \\ [ij], [ik], [il] \notin E, [jk], [kl], [lj] \in E \end{array} \tag{22}$$

$$z_{ij} + z_{ik} + z_{il} + z_{im} \leq z_{ii} \qquad \begin{array}{l} \forall i, j, k, l, m = 1, ..., N \text{ with } j < k, j < l, k < m, \\ [ij], [ik], [il], [im] \notin E, [jk], [jl], [jm], [kl], [km], [lm] \in E \end{array} \tag{23}$$

$$\begin{array}{l} z_{il} + z_{li} + z_{ik} + z_{ki} + z_{jl} + \\ + z_{lj} + z_{jm} + z_{mj} + z_{mk} + z_{km} \end{array} \leq 2 \qquad \begin{array}{l} \forall i, j, k, l, m = 1, ..., N \text{ with } i \neq j \neq k \neq l \neq m, \\ [ik], [il], [jl], [jm], [km] \notin E, [ij], [jk], [kl], [lm], [mi] \in E \end{array} \tag{24}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j = 1, ..., N \text{ with } i < j \tag{25}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i, j = 1, ..., N \text{ with } i \neq j, [ij] \notin E \tag{26}$$

$$z_{ij} \in \{0, 1\} \qquad \forall i, j = 1, ..., N \text{ with } [ij] \notin E \tag{27}$$

$$K \in \mathbb{N} \tag{28}$$

Having developed a fully functional integer programming model for the minimization of open stacks problem, we then explore some variants of this model.

3. MINIMUM INTERVAL GRAPH COMPLETION

The main idea behind the integer programming model presented is the completion of the MOSP graph with suitable fill edges, with the purpose of constructing an interval graph. There are several edge completion problems documented in literature [7]. Here we address the Minimum Interval Graph Completion, which searches for the minimum number of fill edges that should be added to a graph to obtain an interval graph. With small changes in the objective function and using some of the previous constraints, we can build an integer programming model for this problem in Graph Theory.

We will not need the variables z_{ij} because the number of stacks is irrelevant in the minimum interval graph completion problem. Therefore, inequalities (8), (15) to (24), (27) and (28) are droped for this case.

The objective is simply completing the graph with the smallest number of edges to obtain an interval graph. The sum of all variables y gives the number of edges that are not added to the graph G when completing it to an interval graph H. By maximizing this sum, we get a minimum number of added edges.

More formally, the objective function for the minimum interval graph completion problem is

$$\max \sum_{[ij] \notin E} y_{ij} \tag{29}$$

4. MINIMIZING THE STACK OCCUPATION

The model we have developed for the minimization of open stacks can be used in another pattern sequencing problem, where the objective is to find an optimal sequence to process the cutting patterns in order to minimize the occupation of the stacks.

The problem we address now is similar to minimizing the flow time of the orders: besides having the minimum number of open stacks, we also want to minimize the sum of the time that the stacks remain open within the system.

The sequence in which preset cutting patterns are processed can affect the flow and total completion time, so it is desirable to optimize the occupation of the stacks to eliminate unnecessary dispersion.

When considering the MOSP, it is usual to find more than one optimal solution, in the sense that there is more than one sequence of the cutting patterns that achieves the same maximum number of open stacks. We may be interested in choosing between these optimal solutions of the MOSP according to a different criterion. A natural choice is the minimization of the order spread.

Noticing that in most instances there are alternative optimal solutions for the MOSP, we tried to take the problem further and added a second step with a new objective function: the minimization of the order spread. This pattern sequencing problem similar to the MOSP is also related with the minimum interval graph completion problem.

Our model consists in finding out which arcs should be added to the original MOSP graph $G = (V, E)$ in order to get an interval graph $H = (V, E \cup F)$ that minimizes the stack occupation while keeping the minimum number of simultaneously open stacks.

The model we present is divided in two steps. In a first step, the

minimum number of open stacks is determined, and then in a second step, we search for a new sequence of the patterns that improves the total stack spread while using the optimal number of open stacks.

In the first step the formulation is the same as before, with the objective to minimize the maximum number of open stacks. Then, in the second step, the objective becomes the minimization of the stack spread. To minimize the average order spread is equivalent to minimizing the total stack spread. This is also equivalent to minimizing the number of fill-in zeros obtained in the matrix of the description of the cutting patterns after the columns have been rearranged to match the sequence in which the patterns will be processed.

This is done by minimizing the number of arcs that are added to the MOSP graph in order to obtain an interval graph. As the variables y_{ij} are 1 when an arc is not added to the graph, we can minimize the number of added arcs by maximizing the sum of the variables y_{ij}. Therefore the objective function in step 2 is expression (29).

To guarantee that the optimal number of open stacks does not increase from step 1 to step 2, some of the inequalities have to be modified accordingly. Let us denote the optimal number of open stacks found in step 1 by $MOSP^*$. For step 2, in the inequalities (8) and (15), the variable K is replaced by $MOSP^*$.

5. COMPUTATIONAL RESULTS

The integer programming models were tested on the instances of the Constraint Modeling Challenge 2005, available at: http://www.cs.st-andrews.ac.uk/ ipg/challenge/instances.html

The instances were provided by the participants in the challenge and present different kinds of difficulty, such as size, sparseness and symmetry. Computational tests were performed with ILOG OPL Development Studio 5.5 on an Intel®Core2 Duo T7200 @2.00GHz 0.99GB RAM. For each instance, the best objective value found by the model, the best lower bound, the gap, the number of nodes of the search tree and the runtime were recorded.

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

For the problem of minimizing the stack occupation, in the second step we were able to obtain the optimal solution in every instances tested. This second step allowed to reduce the order spread in almost every instance, while maintaining the same optimal number of open stacks. This reduction was very significant in many cases, decreasing around 75% of the number of added edges.

For the Minimum Interval Graph Completion Problem, in all of the instances tested, the optimal solution was reached and proved optimal.

6. ACKNOWLEDGEMENTS

This work was financially supported by the Portuguese Foundation for Science and Technology (FCT) and supported by ESEIG - Superior School of Industrial Studies and Management - Polytechnic Institute of Porto.

7. REFERENCES

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

[2] D. G. Corneil, S. Olariu, and L. Stewart, "The ultimate interval graph recognition algorithm? (Extended Abstract)," in *Symposium on Discrete Algorithms*, 1998, pp. 175–180.

[3] 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.

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

[5] ——, "A transformation for solving a pattern sequencing problem in the wood cut industry," *Pesquisa Operacional*, vol. 17, no. 1, pp. 57–70, 1997.

[6] 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, GRACO 2005 - 2nd Brazilian Symposium on Graphs, Algorithms and Combinatorics.

[7] M. C. Golumbic, H. Kaplan, and R. Shamir, "On the complexity of DNA physical mapping," *Advances in Applied Mathematics*, 1994.