A METHOD FOR SOLVING THE MINIMIZATION OF THE MAXIMUM NUMBER OF OPEN STACKS PROBLEM WITHIN A CUTTING PROCESS

Elmer S. Poliquit

A Thesis Submitted to the University of North Carolina Wilmington in Partial Fulfillment Of the Requirements for the Degree of Master of Science

Department of Mathematics and Statistics
University of North Carolina Wilmington
2008

Approved by

Advisory Committee

Matthew Tenhusien
Jeffrey Brown

John Karlof
Chair

Accepted by
Robert D. Roer
Dean, Graduate School
This thesis has been prepared in the style and format
Consistent with the journal
American Mathematical Monthly.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iv</td>
</tr>
<tr>
<td>DEDICATION</td>
<td>v</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Scope and purpose</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Previous work</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Open stacks problem</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Binary matrix presentation</td>
<td>4</td>
</tr>
<tr>
<td>1.5 Graph model</td>
<td>4</td>
</tr>
<tr>
<td>1.6 Outline of the paper</td>
<td>5</td>
</tr>
<tr>
<td>2 Pre-processing phase</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Propositions</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Algorithms for trees, simple cycles, and 1-trees as components</td>
<td>9</td>
</tr>
<tr>
<td>3 The heuristic of minimal cost node</td>
<td>26</td>
</tr>
<tr>
<td>4 The exact method</td>
<td>32</td>
</tr>
<tr>
<td>4.1 Equivalency proposition</td>
<td>33</td>
</tr>
<tr>
<td>4.2 Heuristic arc contraction</td>
<td>34</td>
</tr>
<tr>
<td>4.3 A branch and bound scheme</td>
<td>38</td>
</tr>
<tr>
<td>4.4 Exact Method: branch and bound approach</td>
<td>42</td>
</tr>
<tr>
<td>5 Conclusion and future work</td>
<td>44</td>
</tr>
</tbody>
</table>
ABSTRACT

In this paper, the problem of minimizing the maximum number of open stacks around a saw machine is addressed. A new heuristic and a branch-and-bound based exact method for the problem are presented.
DEDICATION

This thesis is dedicated to my parents.
ACKNOWLEDGMENTS

I would like to thank Dr. John Karlof, my advisor, for his constant support. I am also thankful to Dr. Matthew TenHuisen and Dr. Jeffrey L. Brown, my committee members, for their help with this thesis.
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set of cutting process</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Stacks observed following sequence</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Data presented in table 1 into a matrix</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Reverse sequence of $P_1P_2P_3P_4P_5P_6$</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>Set of patterns to be sequenced</td>
<td>14</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1. Graph of Table 1 ......................................................... 5
2. Illustration of procedure WALK in special tree 1 .................... 9
3. Illustration of a simple cycle ......................................... 10
4. Illustration of special tree II ......................................... 11
5. Computing the score of $k$ .............................................. 12
6. Tree corresponding to data in Table 5 .................................. 15
7. Graph after deleting nodes and patterns from $W_1$ to $W_9$ .......... 16
8. Graph after deleting node 8 and patterns $P_6P_7$ .................... 17
9. Graph after deleting node 6 and patterns $P_4P_5$ .................... 17
10. 1-tree where $L_2$ is empty when procedure POLYGON is applied ... 22
11. 1-tree where $L_2$ is non-empty when procedure POLYGON is applied ... 23
12. Tree after procedure POLYGON is applied ............................... 24
13. Illustration of a minor of $G$ ........................................... 27
14. Graph $G_p$ for the example ............................................ 29
15. Initialization ..................................................................... 29
16. Resulting graph after arc (2,4) is deleted ............................... 30
17. Resulting graph after arcs (2,4) and (2, 6) are deleted ............... 30
18. Resulting graph after arcs (1, 4), (1, 2), and (1, 6) are deleted ........ 31
19. Resulting graph after arcs (1, 9) and (6, 9) are deleted ............... 31
20. A graph to illustrate lower bound ....................................... 32
21. Identification of equivalencies .......................................... 34
22. A graph to illustrate the Heuristic arc contraction algorithm ....... 36
23. A graph after first contraction .......................................... 36
24. A graph after second contraction ....................................... 36
25. A graph after third contraction = $K_4$ ................................. 37
26 Initial graph $G = G_p$ ................................. 37
27 Minor of $G$ generating $K_3$ ................................. 37
28 Minor of $G$ generating $K_4$ ................................. 38
29 Identification of a node in the branching scheme ............. 39
30 Branching from node 0 ......................................... 40
31 Branching from node 0 ......................................... 40
32 Branching from node 6 ......................................... 41
33 Graph that illustrates the branch and bound .................... 43
34 Graph $G_p$ ...................................................... 44
1 INTRODUCTION

1.1 Scope and purpose

In an industrial environment, one general problem that occurs is how to find a particular sequence of production jobs that minimizes production costs. The mathematical solutions of this type of problem have been the interest of various researchers. The problem that this paper addresses is to determine a method that minimizes the maximum number of open orders of clients. This problem is known as the minimization of open stack problem or MOSP.

The industrial problem that is considered here arises in a production setting which consists of a set of products and a set of customer’s orders. For instance, in the glass industry where different piece types (set of products) are cut that are used for car or office windows. Suppose that the cutting patterns (set of customer’s orders) have already been determined around a saw machine. Each pattern is composed of piece types and as the patterns are cut, the pieces of the same type are stacked together. However, space around the vicinity is limited and hence, the number of distinct stacks should be small. It is assumed that a stack is open as soon as a new piece type is cut and it remains open until all the piece types corresponding to that stack are cut, only then can the stack be removed from the vicinity. This rule has obvious implications on handling costs and it also minimizes risks. Thus, it is necessary to minimize the maximum number of open stacks during the whole production run.

1.2 Previous work

Dyson and Gregory [2] discuss the cut pattern sequence problem aiming to reduce the number of discontinuities among piece types to be cut, i.e. if any piece type does not occur in the next pattern in the sequence, then a cutting discontinuity exists.
Madsen [3,4] presents a procedure to reduce the cut distance among equal piece types and an objective to minimize the distance between corresponding piece types, called the order spread. A spread is the number of different processed patterns between the first and the last piece types.


Yuen [8,9] presents six heuristics for solving the MOSP where the third heuristic, the Yuen3, has a very good practical behavior. Yuen and Richardson [11] present the full explanation and some optimality conditions for these six heuristics. Faggioli and Bentivoglio [10] yield a two-phase approach in solving the MOSP, the implementation of the heuristic procedure and the enumeration on the patterns using a branch and bound approach.

Yanasse [1,12] and Limeira [13] present the major concepts in the formulation for the method presented in this paper. Faggioli and Bentivoglio [14] present the formulation of MOSP as a linear integer programming (LIP), but Becceneri [15] refers that a direct attack through LIP may be infeasible which is reinforced by Yannase [16] that the problem is proven to be NP-Hard.

1.3 Open stacks problem

Consider a production setting shown in table 1, the piece types contained in each pattern is presented. The number of times that a piece type is cut in each pattern is not indicated since it is not relevant for the MOSP. For pattern $P_1$, observe that three stacks will be opened, one for $p_1$, one for $p_2$ and another for $p_4$. The same number of stacks would be open if there is a pattern with two piece types of $p_1$, five piece types of $p_2$ or three piece types of $p_4$. Observe that when closing a stack all
Table 1: Set of cutting process

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Piece type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>$p_1p_2p_4$</td>
</tr>
<tr>
<td>$P_2$</td>
<td>$p_2p_4p_5p_6$</td>
</tr>
<tr>
<td>$P_3$</td>
<td>$p_3p_4$</td>
</tr>
<tr>
<td>$P_4$</td>
<td>$p_1p_3p_5$</td>
</tr>
<tr>
<td>$P_5$</td>
<td>$p_1p_4p_5$</td>
</tr>
<tr>
<td>$P_6$</td>
<td>$p_5p_6$</td>
</tr>
</tbody>
</table>

Table 2: Stacks observed following sequence

<table>
<thead>
<tr>
<th>Pattern sequenced</th>
<th>Open piece types</th>
<th>Closed piece types</th>
<th>Number of open stacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>$p_1p_2p_4$</td>
<td>Finished $p_2$</td>
<td>3</td>
</tr>
<tr>
<td>$P_2$</td>
<td>$p_1p_2p_4p_5p_6$</td>
<td>$p_3p_4$</td>
<td>5</td>
</tr>
<tr>
<td>$P_3$</td>
<td>$p_1p_3p_4p_5p_6$</td>
<td>Finished $p_3$</td>
<td>5</td>
</tr>
<tr>
<td>$P_4$</td>
<td>$p_1p_3p_4p_5p_6$</td>
<td>Finished $p_1$ and $p_4$</td>
<td>4</td>
</tr>
<tr>
<td>$P_5$</td>
<td>$p_1p_4p_5p_6$</td>
<td>Finished $p_5$ and $p_6$</td>
<td>2</td>
</tr>
</tbody>
</table>

patterns containing the corresponding piece must have been cut.

Suppose that we want to find the maximum number of open stacks for sequence $P_1P_2P_3P_4P_5P_6$. We start cutting piece types $p_1p_2p_4$ of pattern $P_1$ with three open stacks. Pieces of pattern $P_2$ are to be cut next where the stack of piece type $p_2$ is already finished at this stage with five open stacks. Following this idea, we end up with a maximum number of open stacks of five as illustrated in table 2.

Another possible sequence is $P_3P_4P_5P_1P_2P_6$ with a maximum of four open stacks. Observe that for this 6-pattern problem, there are already $6! = 720$ possible different sequences to check which one minimizes the maximum number of open stacks. Thus, we are interested in finding the optimal sequence that yields the minimum of the maximum number of open stacks.
Table 3: Data presented in table 1 into a matrix

\[
P_{i,j} = \begin{bmatrix}
  p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \\
  P_1 & 1 & 1 & 0 & 1 & 0 & 0 \\
  P_2 & 0 & 1 & 0 & 1 & 1 & 1 \\
  P_3 & 0 & 0 & 1 & 1 & 0 & 0 \\
  P_4 & 1 & 0 & 1 & 0 & 1 & 0 \\
  P_5 & 1 & 0 & 0 & 1 & 1 & 0 \\
  P_6 & 0 & 0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

1.4 Binary matrix presentation

In the production setting with \( I \) distinct patterns given, each one of these patterns contains a combination of at most \( J \) piece types. The input of the data can be given by an \( I \times J \) binary matrix \( P \), representing patterns as rows and piece types as columns where

\[
P_{i,j} = \begin{cases} 
1 & \text{if piece type } j \text{ is to be cut from the pattern } i, \\
0 & \text{otherwise.} 
\end{cases}
\]

The data in table 1 as a binary matrix is illustrated in table 3. The first entry 1 means that in pattern \( P_1 \), piece type \( p_1 \) is present. Thus, row 1 as pattern \( P_1 \) contains piece types \( p_1p_2p_4 \).

1.5 Graph model

The input data for the MOSP can be given by a graph. In graph \( G(V, E) \), a node \( k \in V \) represents the piece type \( p_k \) and two nodes \( i \) and \( j \) are adjacent (that is, \((i, j) \in E\)) if and only if the corresponding piece types \( p_i \) and \( p_j \) are simultaneously present in the same pattern.

The alteration of the patterns given in table 1 into a graph is illustrated in figure 1. The nodes 1, 2, 3, 4, 5, 6 represent the piece types \( p_1, p_2, p_3, p_4, p_5, p_6 \), respectively.
Pattern $P_1$ contains arcs $(1,2)$, $(1,4)$, and $(2,4)$, $P_2$ contains arcs $(2,4)$, $(2,5)$, $(2,6)$, $(4,5)$, $(4,6)$, $(5,6)$, and so on.

1.6 Outline of the paper

In this paper, we consider an MOSP with at most two piece types a pattern. The method for solving the MOSP is the incorporation of the algorithms presented in the next sections. Each algorithm is illustrated with graphical examples.

The next sections are presented as follows. In section 2, we have the preprocessing procedures on the input data that simplify the problem, and the algorithms for trees, simple cycles and 1-trees are presented. In section 3, we present an algorithm for the heuristic minimal cost node of a minor of graph $G$ that uses the idea of visiting all arcs in $G_p$, a minor of $G$. In section 4, we present the exact method in a branch-and-bound scheme. This section is subdivided into 4 subsections with the inclusion of the detailed implementation of a lower bound. Finally, in section 5, we present the conclusion and future work.

2 Pre-processing phase

In this section, we present the two main objectives of the pre-processing that use the idea of a clique in a graph. We begin with a definition and then the propositions and algorithms in the following subsections.
Definition 1  A clique in a graph \( G \) is a complete subgraph \( K_h \) of \( G \). The size of a clique is the number of vertices it contains which is \( h \).

It is assumed, without loss of generality, that the input data for MOSP is given by a matrix or a graph as illustrated in subsections 1.4 and 1.5.

A pattern \( P_i \) with \( h \) different piece types defines a clique \( K_h \) of size \( h \) and it represents an obvious optimal solution when the piece types of the rest of the patterns in MOSP are contained in \( P_i \). In addition, an optimal solution to the problem can also be examined as a succession of large cliques.

The two main objectives in this phase are (i) to eliminate redundancies among the patterns and (ii) to detect conditions for which a solution can be found in polynomial time.

(i) A redundancy occurs whenever in \( G \), a clique is a sub-graph of another larger one. For instance, in table 1, \( P_6 \subseteq P_2 \), this means that \( P_6 \) can be removed during an optimal solution search of the problem, since \( P_6 \) can be inserted immediately after \( P_2 \) in an optimal sequence and no more stacks will be open. We say, \( P_2 \) dominates \( P_6 \). The dominance of a pattern happens whenever \( P_j \subseteq P_i \), where \( i \neq j \).

(ii) After eliminating the redundancies among the patterns, we check whether the sub-graphs of \( G \) have trees, simple cycles or 1-trees as components. If such components occur, we apply the algorithms presented in the next subsection to generate an optimal sequence for the piece types contained in these components. The main idea is to explore the structure of graphs which possess "small" cliques in order to find its lower bound that can be the basis for the optimal solution. Generally, an optimal solution to MOSP is given by a sequence of possibly large and overlapping cliques, connected by components such as trees. Hence, it is important to exactly solve the schedule of the latter components in polynomial time to incorporate them into the final solution.
Table 4: Reverse sequence of $P_1P_2P_3P_4P_5P_6$

<table>
<thead>
<tr>
<th>Pattern sequenced</th>
<th>Open piece types</th>
<th>Closed piece types</th>
<th>Number of open stacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_6$</td>
<td>$p_5p_6$</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$P_5$</td>
<td>$p_1p_4p_5p_6$</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>$P_4$</td>
<td>$p_1p_3p_4p_5p_6$</td>
<td>Finished $p_3$</td>
<td>5</td>
</tr>
<tr>
<td>$P_3$</td>
<td>$p_1p_3p_4p_5p_6$</td>
<td>Finished $p_5$ and $p_6$</td>
<td>5</td>
</tr>
<tr>
<td>$P_2$</td>
<td>$p_1p_2p_4p_5p_6$</td>
<td>Finished $p_1, p_2$ and $p_4$</td>
<td>3</td>
</tr>
<tr>
<td>$P_1$</td>
<td>$p_1p_2p_4$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.1 Propositions

The algorithms presented in the next subsection are based on the following propositions considering that a pattern contains at most two piece types. Each proposition is illustrated by a simple example.

Proposition 1 Let $L$ be the list of open piece types with a possible sequencing of the patterns of a MOSP, $P_1P_2 \ldots P_n$. The reverse order sequence of the patterns $P_n \ldots P_2P_1$ produces the same open piece types in reverse order, that is, list $L$ in reverse order.

Proof. Suppose the pattern sequence $P_1P_2 \ldots P_n$ generates a list of open piece types $L = l_1l_2 \ldots l_n$ where $l_i$ represents the list of open piece types in $P_i$. Then the reverse pattern sequence $P_n \ldots P_2P_1$ generates a list of open piece types $l_n \ldots l_2l_1$, that is, list $L$ in reverse order. □

To illustrate the proposition, consider for instance table 2 with a pattern sequence $P_1P_2P_3P_4P_5P_6$. Column 2 of table 2 lists the open piece types which generates a maximum number of open stack 5. Table 4 shows that the reverse order of the patterns lists the reverse order of the open piece types (see column 2) with the same maximum number of open stacks.
Proposition 2 Suppose we have a MOSP, PROB_1, with \( N \) patterns to be sequenced. Suppose another problem, PROB_2, with the same \( N \) patterns as problem \( PROB_1 \) plus a few more patterns. Then the optimal solution value of \( PROB_2 \) is greater than or equal to the optimal solution value of \( PROB_1 \).

Proof. Trivial. Take the optimal solution of \( PROB_2 \) and obtain a feasible solution to \( PROB_1 \) by simply ignoring, in the sequence, the patterns that belong only to \( PROB_2 \). □

In table 1, pattern \( P_6 \) is dominated by \( P_2 \). Removing pattern \( P_6 \) from the problem still yields an optimal solution of 5. This illustrates proposition 2.

Proposition 3 Let \( G(V, E) \), the graph obtained from a problem \( P \), be a connected non-empty graph where the nodes have minimum degree \( n (n \geq 2) \). Then, a lower bound on the optimal solution value to \( P \) is \( n + 1 \), that is, in an optimal solution to \( P \), we must have at least \( n + 1 \) open stacks at some time during cutting.

Proof. For a node of degree \( n \), its piece type will be contained in \( n \) patterns since all piece types are contained in at most two patterns. Then, for any sequencing of the patterns, a piece type, say \( i \), will remain open until all \( n \) or more patterns containing it are cut, that is, all arcs incident to node \( i \) are sequenced. Each one of these arcs is incident to a different node, say \( j \), corresponding to another piece type, which in turn, remains unfinished unless all \( n \) or more patterns containing it are sequenced. All arcs incident to any node will have to be sequenced sometime. Since no node can be finished unless all its \( n \) (or more) incident arcs are sequenced and since each one of them creates an open stack, the proposition holds. □

Take for instance, the graph of figure 1. Node 6 has a minimum degree 3 and the rest of the nodes have degree greater than or equal to 3. Therefore, any feasible sequencing of the patterns will produce at least 4 open stacks at some time during the cutting.
2.2 Algorithms for trees, simple cycles, and 1-trees as components

In this subsection, we present an algorithm for a few special cases of MOSP. We begin with the simplest case and proceed to more complicated ones. Definitions and procedures are introduced for easy understanding.

Case 1 Graph with a single node.

This case corresponds to having a pattern with one piece type. This is trivial.

Case 2 Special tree I - A tree with all nodes having degree 2 or less

If the graph has a single node then we are in case 1. Else, we sequence the patterns in the following procedure.

Procedure WALK

Start with a pattern that finishes a piece type completely, that is, an arc incident to a single node with degree 1 and keep sequencing next the pattern that ends any open piece type.

Definition 2 The sequence of patterns obtained using procedure WALK is called walk \( W \). A walk \( revW \) is the reverse order sequence of patterns of \( W \). The maximum number of open piece types of a walk \( W \) is its degree denoted by \( w \).

Consider the graph in figure 2, we start with pattern \( P_1 \) since node 1 has degree 1, then patterns \( P_2P_3 \) or we start with pattern \( P_3 \), then patterns \( P_2P_1 \). The procedure WALK applied yields a walk \( W = P_1P_2P_3 = P_3P_2P_1 \) with \( w = 2 \). Either of the sequencing results in an optimal solution as explained in proposition 1.

![Figure 2: Illustration of procedure WALK in special tree 1](image-url)
Case 3 *Simple Cycle* (a polygon) - nonempty graph with all nodes of degree 2

In this case, we follow the strategy of sequencing patterns in order to finish any open piece types. Any pattern can be chosen to start with the sequencing.

For instance in figure 3, the easiest way to find an optimal solution is to follow the sequence $P_1P_2P_3P_4P_6$ or its reverse order with a maximum of 3 open stacks according to proposition 3. We can also have the same optimal solution for sequence $P_1P_6P_2P_3P_4P_5$.

![Figure 3: Illustration of a simple cycle](image)

Case 4 *Special tree II* - A tree with all nodes of degree two or less with the exception of at most one of degree 3.

For this case, we introduce a another procedure that is also needed in the next cases.

**Procedure CONSTRUCT**

Let $\Gamma_k$ be the set of walks with the same end node $k$ to be sequenced. If $|\Gamma_k| \geq 2$ then $\deg(k) \geq 3$. The sequence of walks must begin with walk $W_i$ and end with walk $\text{rev}W_j$ where $W_i, W_j \in \Gamma_k$ and the degrees of $W_i$ and $W_j$ are respectively the largest and the second largest.

**Algorithm tree II**
Apply procedure WALK starting with a node of degree 1 and end with a node of degree greater than 2. When the end node of walk \( W \) has degree greater than 2 then degree \( w \) excludes the end node. With the walks obtained, apply procedure CONSTRUCT.

Consider the graph in figure 4, we can obtain three walks with the same end node 5. Suppose we have \( W_1 = P_1P_2 \) with \( w_1 = 2 \), \( W_2 = P_6 \) with \( w_2 = 1 \), and \( W_3 = P_5P_4P_3 \) with \( w_3 = 2 \). Then applying procedure CONSTRUCT, we have an optimal sequence of walks \( W_1W_2\text{rev}W_3 \) or \( W_3W_2\text{rev}W_1 \) equivalent to the sequence of patterns \( P_1P_2P_6P_3P_4P_5 \) or \( P_5P_4P_3P_6P_2P_1 \), respectively. The maximum number of open stacks is 2.

![Figure 4: Illustration of special tree II](image)

**Case 5 Trees**

The algorithm for trees uses a similar concept as the algorithm tree II. We construct a solution for the case trees by adding together the optimal solutions of its subtrees. We start with a definition of a score that is needed throughout the algorithm. Procedures WALK and CONSTRUCT are also applied.

**Definition 3** Let \( \Gamma_k \) be a set of walks with the same end node \( k \). The score of \( k \) is 
\[
s(k) = \max\{w: W \in \Gamma_k\} + \text{ADJUSTMENT}
\]

\[
\text{ADJUSTMENT} = \begin{cases} 
1 & \text{if more than one walk have the same maximum degree}(w), \\
0 & \text{otherwise}.
\end{cases}
\]
Let $W_i$ denote the walk $i$ corresponding to $\text{subtree}_i$ and $w_i$ denote its corresponding degree. Suppose that in figure 5, only walks $W_1$, $W_2$, and $W_3$ are considered so far with end node $k$. If for instance, we have $w_1 = w_3 = 2$ and $w_2 = 1$, then the score of $k$ is three. In this instance, the score of $k$ provides an approximate minimum value of the maximum number of open stacks that will result in any optimal sequencing of the patterns in the subtrees having end node $k$ as the root node.

![Figure 5: Computing the score of $k$](image)

**Algorithm TRESSES**

For this case, we follow four steps. We denote $L$ as a set of nodes with degree 1.

**Step 0** (Initialization) Set the score of all nodes in the graph equal to 0. List ($L$) all nodes with degree 1.

**Step 1** Using the procedure WALK, determine the walks $W_i$, $i = 1, 2, \ldots, n$, starting from each node in $L$ and end with a node of degree greater than two. Include its corresponding degree $w_i$ and end node $k_i$. If a walk contains only a pattern then its degree $w = 1$.

**Step 2** Delete all the nodes and patterns used in $W_i$ from the graph excluding their end nodes. With the resulting graph, list ($L$) all nodes with degree 1 and compute the score of each end node $k_i$ from the walks obtained.
Step 3 If there is only one node \( k \) in \( L \), construct a final sequence of walks with end node \( k \), following procedure CONSTRUCT and STOP. Otherwise, among the nodes in the list choose one, say \( k \), that has the smallest score. Break ties at random. Then

3.1 From the resulting graph obtained in step 2, create a walk starting from node \( k \) and end with a node of degree greater than two. If node \( k \) is adjacent to a node with zero score, then this walk must be affixed right after the sequence of walks constructed in 3.2. Otherwise, affixing is not needed.

3.2 Using procedure CONSTRUCT, create a walk equal to a sequence of patterns using the walks obtained having an end node \( k \). Affix to this sequence the pattern or patterns equated from the walk created in 3.1 and consider this as an ”extended” walk. Calculate the degree \( w \) associated to this ”extended” walk. Store this extended walk with its corresponding degree \( w \) and end node.

3.3 Delete all the walks used in obtaining the extended walk then return to step 2.

Suppose we have to sequence 22 patterns with two piece types per pattern as presented in table 5. The graph of this table is in figure 6.

The following are the steps of the algorithm that are applied in figure 6.

Step 0: Set each node score equal to 0. \( L = \{1, 4, 10, 12, 14, 18, 19, 21, 23\} \).

Step 1: Walks determined from each node in \( L \) with degree 1.
Table 5: Set of patterns to be sequenced

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Piece Type</th>
<th>Pattern</th>
<th>Piece Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>$p_1p_2$</td>
<td>$P_{12}$</td>
<td>$p_6p_{13}$</td>
</tr>
<tr>
<td>$P_2$</td>
<td>$p_2p_3$</td>
<td>$P_{13}$</td>
<td>$p_{13}p_{14}$</td>
</tr>
<tr>
<td>$P_3$</td>
<td>$p_3p_4$</td>
<td>$P_{15}$</td>
<td>$p_{15}p_{16}$</td>
</tr>
<tr>
<td>$P_4$</td>
<td>$p_3p_5$</td>
<td>$P_{16}$</td>
<td>$p_{16}p_{17}$</td>
</tr>
<tr>
<td>$P_5$</td>
<td>$p_5p_6$</td>
<td>$P_{17}$</td>
<td>$p_{17}p_{18}$</td>
</tr>
<tr>
<td>$P_6$</td>
<td>$p_6p_7$</td>
<td>$P_{18}$</td>
<td>$p_{16}p_{19}$</td>
</tr>
<tr>
<td>$P_7$</td>
<td>$p_7p_8$</td>
<td>$P_{19}$</td>
<td>$p_{16}p_{20}$</td>
</tr>
<tr>
<td>$P_8$</td>
<td>$p_8p_9$</td>
<td>$P_{20}$</td>
<td>$p_{20}p_{21}$</td>
</tr>
<tr>
<td>$P_9$</td>
<td>$p_9p_{10}$</td>
<td>$P_{21}$</td>
<td>$p_{15}p_{22}$</td>
</tr>
<tr>
<td>$P_{10}$</td>
<td>$p_8p_{11}$</td>
<td>$P_{22}$</td>
<td>$p_{22}p_{23}$</td>
</tr>
<tr>
<td>$P_{11}$</td>
<td>$p_{11}p_{12}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

starting node  $W_i$  $w_i$  end node

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$W_1 = P_1P_2$</td>
<td>$w_1 = 2$</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>$W_2 = P_3$</td>
<td>$w_2 = 1$</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>$W_3 = P_9P_8$</td>
<td>$w_3 = 2$</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>$W_4 = P_{11}P_{10}$</td>
<td>$w_4 = 2$</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>$W_5 = P_{13}P_{12}$</td>
<td>$w_5 = 2$</td>
<td>6</td>
</tr>
<tr>
<td>18</td>
<td>$W_6 = P_{17}P_{16}$</td>
<td>$w_6 = 2$</td>
<td>16</td>
</tr>
<tr>
<td>19</td>
<td>$W_7 = P_{18}$</td>
<td>$w_7 = 1$</td>
<td>16</td>
</tr>
<tr>
<td>21</td>
<td>$W_8 = P_{20}P_{19}$</td>
<td>$w_8 = 2$</td>
<td>16</td>
</tr>
<tr>
<td>23</td>
<td>$W_9 = P_{22}P_{21}$</td>
<td>$w_9 = 2$</td>
<td>15</td>
</tr>
</tbody>
</table>

Step 2: Delete all the nodes and patterns used from $W_1$ to $W_9$ excluding their end nodes. The resulting graph is in figure 7. $L = \{8, 16\}$. 

14
Step 3: End nodes 8 and 16 have the same score. We select $k = 8$.

3.1 Starting from node 8, $W_{10} = P_7P_6$ with $w_{10} = 2$ and end node 6.

Since $s(7) = 0$, the score of the adjacent node, affixing is needed.

3.2 Using procedure CONSTRUCT: $W_{11} = W_3revW_4W_{10} = P_9P_8P_{10}P_{11}P_7P_6$ with $w_{11} = 3$ and end node 6.

Figure 7: Graph after deleting nodes and patterns from $W_1$ to $W_9$

Step 2: Delete nodes 7, 8 and patterns $P_6P_7$, the resulting graph is in figure 8.
$L = \{6, 16\}$.

<table>
<thead>
<tr>
<th>End node</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
</tr>
</tbody>
</table>

Step 3: $k = 6$

3.1 Starting from node 6, $W_{12} = P_5P_4$ with $w_{12} = 2$ and end node 3.
Since $s(5) = 0$, then affixing is needed.

3.2 Using procedure CONSTRUCT: $W_{13} = W_{11}\text{rev}W_5W_{12} = P_9P_8P_{10}P_{11}P_7P_6P_{12}P_{13}P_5P_4$
with $w_{13} = 3$ and end node 3.

3.3 Walks deleted: $W_5W_{11}W_{12}$. Return to step 2.
Step 2: Delete nodes 5, 6 and patterns $P_4P_5$, the resulting graph is in figure 9.

$L = \{3, 16\}$.

\[
\begin{array}{c|c}
\text{End node} & \text{Score} \\
3 & 3 \\
15 & 2 \\
16 & 3 \\
\end{array}
\]

Step 3: $k = 3$

3.1 Starting from node 3, $W_{14} = P_{14}$ with $w_{14} = 1$ and end node 15.

Adjacent node 15 has score equal to 2. Affixing is not needed.

3.2 Using procedure CONSTRUCT: $W_{15} = P_9P_8P_{10}P_{11}P_7P_6P_{12}P_{13}P_5P_4P_3P_2P_1$ with
\( w_{15} = 3 \) and end node 15.

3.3 Walks deleted: \( W_1 W_2 W_{13} \). Return to step 2.

Step 2: Delete node 3 and pattern \( P_{14} \), the resulting graph is left with nodes 15 and 16 connected by pattern \( P_{15} \). \( L = \{15, 16\} \).

<table>
<thead>
<tr>
<th>End node</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
</tr>
</tbody>
</table>

Step 3: \( k = 15 \)

3.1 Starting from node 15, \( W_{16} = P_{15} \) with \( w_{16} = 1 \) and end node 16.

Adjacent node 3 has score equal to 3. Affixing is not needed.

3.2 Using procedure CONSTRUCT: \( W_{17} = P_9 P_8 P_{10} P_{11} P_{12} P_7 P_6 P_{13} P_5 P_4 P_3 P_2 P_1 P_{14} P_{21} P_{22} \) with \( w_{17} = 3 \) and end node 15.

3.3 Walks deleted: \( W_9 W_{14} W_{15} \). Return to step 2.

Step 2: Delete node 15 and pattern \( P_{15} \), the resulting graph is left with a single node 16. \( L = \{16\} \).

Step 3: \( k = 16 \), using procedure CONSTRUCT then we have one possible optimal sequence equal to \( W_{17} W_6 W_7 W_{16} \text{rev} W_8 = P_9 P_8 P_{10} P_{11} P_7 P_6 P_{12} P_{13} P_5 P_4 P_3 P_2 P_1 P_{14} P_{21} P_{22} P_{17} P_{16} P_{18} P_{15} P_{19} P_{20} \) with a maximum number of open stacks equal to 3.

Case 6 1-Trees - "Almost trees"

Definition 4 A 1-tree is formed by connecting a tree with a node outside the tree with two arcs so that a cycle is formed.
For this case, we follow the same steps in algorithm TREES until reaching a polygon subgraph. Then we apply a new procedure, procedure POLYGON. Procedure POLYGON has three modified steps from algorithm TREES and two other procedures, procedure CONSTRUCT1 and procedure CONSTRUCT2. The two other procedures are constructed to ensure that the maximum number of open stacks is kept to the least possible value.

Algorithm 1-TREES

We suppose that the resulting graph is a polygon after using the algorithm for TREES. We denote $L_i$ as a set of walks where $i = 1, 2, 3$.

Procedure POLYGON

Select a node $k_1$ with the largest score in the polygon. Split the list of walks obtained from using algorithm TREES into two lists, $L_1$ and $L_2$, in descending order of degree $w$ where $L_1$ contains walks with end node $k_1$ and $L_2$ contains the remaining walks. Consequently, we have one of the following cases with its procedure in generating an optimal sequencing:

(i) $L_1$ and $L_2$ are empty.

Apply the procedure for the cycle case.

(ii) $L_1$ is non-empty but $L_2$ is empty.

Starting from a pattern in the polygon incident to node $k_1$, determine a sequence of patterns using the procedure in cycle case and consider this as a walk with degree $w = 2$ and end node $k_1$. Then apply procedure CONSTRUCT using all the walks obtained with end node $k_1$.

(iii) $L_1$ and $L_2$ are non-empty.

Let $P_a$ and $P_b$ be the two incident patterns to node $k_1$ in the polygon. Introduce an artificial node $k_a$ incident to $P_a$, separating node $k_1$ and its incident pattern
$P_b$ so that the resulting graph forms a tree. Set the score of $k_1$ and $k_a$ equal to zero. Then apply the following modified steps:

**Step 1** Using procedure WALK, determine new walks $W_i$ and $W_{i+1}$ starting from nodes $k_1$ and $k_a$ with end nodes that have scores greater than zero. These walks $W_i$ and $W_{i+1}$ are the succeeding walks from the walks obtained using algorithm TREES. Include to these new walks their corresponding degrees $w$ and their end nodes. List these new walks as $L_3$. If a walk contains only a pattern then its degree $w = 1$.

**Step 2** Delete all the nodes and patterns used from the walks newly obtained in the graph excluding their end nodes. With the resulting graph, list (L) all nodes with degree 1 and compute the score of each end node from all the walks obtained.

**Step 3** If there is only one node $k$ in L, construct a final sequence of walks with end node $k$ using all the walks listed in $L_1$, $L_2$, and $L_3$, following procedure CONSTRUCT1 and STOP. Otherwise, among the nodes in L choose one, say $k$, that has the smallest score. Then

3.1 From the resulting graph obtained in step 2, create a walk starting from node $k$ and end with a node’s score greater than zero. Add this walk to $L_2$. If node $k$ is adjacent to a node with zero score, then this walk must be affixed right after the sequence of walks constructed in 3.2. Otherwise, affixing is not needed.

3.2 Using procedure CONSTRUCT2, create a walk equal to a sequence of patterns using the walks obtained having an end node $k$. Affix to this sequence the pattern or patterns equated from the walk created in 3.1 and consider this as an “extended” walk. Calculate the degree $w$ associated to this “extended”
walk. Store this extended walk with its corresponding degree \( w \) and end node.

Add this walk to \( L_3 \).

3.3 Delete all the walks used in obtaining the "extended" walk, rewrite \( L_2 \) and \( L_3 \) with the new walks excluding the deleted walks then return to step 2.

**Procedure CONSTRUCT1**

Let \( W_v \) be the walk having the largest degree \( w \) in \( L_2 \). Let \( W_s \) and \( W_t \) be the last walks listed in \( L_3 \) with end node \( k \) where the degree \( w_t \geq w_s \). Let \( L_1 = \{W_c,W_f,\ldots,W_e\} \) where \( W_c,W_f,\ldots,W_e \) are in descending order of degree \( w \).

If degree \( w_c = w_f \) and it is greater than \( w_t \) and \( w_v \), i.e. \( w_c = w_f > w_t \) and \( w_c = w_f > w_v \), then

a. Construct a sequence of walks with end node \( k_1 \) from \( L_1 \) using procedure CONSTRUCT;

b. Construct a sequence of walks with end node \( k \) from \( L_2 \) and \( L_3 \) using procedure CONSTRUCT where the second walk in the sequence must have the first pattern that is connected to \( k_1 \);

c. Construct a final sequence by affixing the sequence in b to a.

Otherwise, start with a sequence from \( W_c,W_f,\ldots,W_e,W_t,W_s \), then all walks in \( L_2 \) and end with \( revW_v \).

**Procedure CONSTRUCT2**

Let \( k \) be the node selected. Let \( W_s \) and \( W_t \) be the walks with end node \( k \) having the largest degree \( w \) from \( L_2 \) and \( L_3 \), respectively. Break ties at random. If \( w_t \geq w_s \) then a sequence of walks is \( W_t,W_i,W_l,\ldots,W_j,revW_s \) where \( W_t,W_i,\ldots,W_j \) are the walks in descending order of degree \( w \) from \( L_2 \) with end node \( k \). Otherwise, \( W_s,W_i,W_l,\ldots,W_j,revW_t \).

The following are examples in finding the optimal sequence of a 1-tree graph:
**Example 1.** \(L_1\) is non-empty but \(L_2\) is empty.

Given a graph in figure 10, we begin by using the algorithm TREES.

![Graph](image)

Figure 10: 1-tree where \(L_2\) is empty when procedure POLYGON is applied

**Step 0:** Each node score is set to 0. \(L = \{1, 7, 9\}\).

**Step 1:** Walks determined from each node in \(L\) with degree 1.

<table>
<thead>
<tr>
<th>starting node</th>
<th>(W_i)</th>
<th>(w_i)</th>
<th>end node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(W_1 = P_1)</td>
<td>(w_1 = 1)</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>(W_2 = P_7P_6)</td>
<td>(w_2 = 2)</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>(W_3 = P_9P_8)</td>
<td>(w_3 = 2)</td>
<td>2</td>
</tr>
</tbody>
</table>

**Step 2:** Delete all the nodes and patterns used from \(W_1\) to \(W_3\) excluding their end nodes. The resulting graph is a polygon. \(L = \phi\).

<table>
<thead>
<tr>
<th>End node</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Procedure POLYGON: \(k_1 = 2\). \(L_1 = \{W_3, W_2, W_1\}\) and \(L_2 = \phi\).

Starting from node 2, \(W_4 = P_3P_5P_4P_2\) with \(w_4 = 2\) and end node 2.

Procedure CONSTRUCT: \(W_2W_3W_1\text{rev}W_4 = P_7P_6P_5P_8P_1P_2P_4P_5P_3\) with a maximum number of open stacks equal to 3.

**Example 2.** \(L_1\) and \(L_2\) are non-empty.

Consider the graph in figure 11.
After using algorithm TREES, the resulting graph is a polygon. The following are the walks left and their end node's scores:

From step 1 using the algorithm TREES:

<table>
<thead>
<tr>
<th>starting node</th>
<th>$W_i$</th>
<th>$w_i$</th>
<th>end node</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>$W_1 = P_7P_6$</td>
<td>$w_1 = 2$</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>$W_2 = P_8$</td>
<td>$w_2 = 1$</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>$W_3 = P_{10}P_9$</td>
<td>$w_3 = 2$</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>$W_4 = P_{11}$</td>
<td>$w_4 = 1$</td>
<td>3</td>
</tr>
</tbody>
</table>

For $k = 12$, $W_{10} = P_{14}P_{13}P_{15}P_{16}P_{12}$ with $w_{10} = 3$ and end node 4.

For $k = 17$, $W_{12} = P_{19}P_{18}P_{20}P_{21}P_{17}$ with $w_{12} = 3$ and end node 4.

<table>
<thead>
<tr>
<th>End node</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>
Applying (iii) in procedure POLYGON, we have the resulting graph in figure 12 with $k_1 = 4$.

![Figure 12: Tree after procedure POLYGON is applied](image)

$L_1 = \{W_{10}, W_{12}\}$ and $L_2 = \{W_1, W_3, W_2, W_4\}$.

Step 1: Walks determined from nodes 4 and 4:\

<table>
<thead>
<tr>
<th>starting node</th>
<th>$W_i$</th>
<th>$w_i$</th>
<th>end node</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$W_{13} = P_4P_5$</td>
<td>$w_{13} = 2$</td>
<td>1</td>
</tr>
<tr>
<td>43</td>
<td>$W_{14} = P_3$</td>
<td>$w_{14} = 1$</td>
<td>3</td>
</tr>
</tbody>
</table>

and $L_3 = \{W_{13}, W_{14}\}$.

Step 2: Delete nodes 4, 43, 5 and patterns $P_3P_4P_5$. The resulting graph is left with nodes 1, 2, 3 and patterns $P_1P_2$. $L = \{1, 3\}$.

<table>
<thead>
<tr>
<th>End node</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Step 3: $k = 3$

3.1 Starting from node 3, $W_{15} = P_2$ with $w_{15} = 1$ and end node 2. $S(2) = 1$, affixing is not needed.

3.2 Using procedure CONSTRUCT2: Extended walk $W_{16} = W_3W_4\text{rev}W_{14} = P_{10}P_9P_{11}P_3$ with $w_{16} = 2$ and end node 2.
3.3 Walks deleted: \( W_3W_4W_{14}, L_2 = \{W_1, W_2, W_{15}\} \) and \( L_3 = \{W_{13}, W_{16}\} \). Return to step 2.

Step 2: Delete node 3 and pattern \( P_2 \). The resulting graph is left with nodes 1, 2 and pattern \( P_1 \). \( L = \{1, 2\} \).

\[
\begin{array}{cc}
\text{End node} & \text{Score} \\
1 & 3 \\
2 & 2 \\
\end{array}
\]

Step 3: \( k = 2 \)

3.1 Starting from node 2, \( W_{17} = P_1 \) with \( w_{17} = 1 \) and end node 1. \( S(1) = 3 \), affixing is not needed.

3.2 Using procedure CONSTRUCT2: Extended walk \( W_{18} = W_{16}W_{13}\text{rev}W_2 = P_{10}P_9P_{11}P_5P_2P_8 \) with \( w_{18} = 2 \) and end node 1.

3.3 Walks deleted: \( W_2W_{15}W_{16}, L_2 = \{W_1, W_{17}\} \) and \( L_3 = \{W_{13}, W_{18}\} \). Return to step 2.

Step 2: Delete node 2 and pattern \( P_1 \). The resulting graph is left with a single node 1.

\[
\begin{array}{cc}
\text{End node} & \text{Score} \\
1 & 3 \\
\end{array}
\]

Step 3: Single node \( k = 1 \) is left. \( L_1 = \{W_{10}, W_{12}\} \), \( L_2 = \{W_1, W_{17}\} \) and \( L_3 = \{W_{13}, W_{18}\} \).

\( W_v = W_1 \) with \( w_1 = 2 \), \( W_s = W_{18} \) with \( w_{18} = 2 \) and \( W_t = W_{13} \) with \( w_{13} = 2 \).

Since \( w_{10} = w_{12} > w_{13} \) and \( w_{10} = w_{12} > w_1 \), then applying procedure CONSTRUCT1, we have

a. The sequence of walks with end node \( k_1 = 4 \) is equal to \( W_{10}\text{rev}W_{12} = P_{14}P_{13}P_{15}P_{16}P_{12}P_{17}P_{21}P_{20}P_{18}P_{19} \).

b. The sequence of walks with end node \( k = 1 \) is equal to \( W_{18}W_{13}W_{17}\text{rev}W_1 = P_{10}P_9P_{11}P_9P_2P_8P_4P_5P_1P_6P_7 \) and
c. Then, the optimal sequence equals $W_{10}\text{rev}W_{12}W_{18}W_{13}W_{17}\text{rev}W_{1} = P_{14}P_{13}P_{15}P_{16}P_{12}P_{17}P_{21}P_{20}P_{18}P_{19}P_{10}P_{11}P_{3}P_{2}P_{8}P_{4}P_{5}P_{1}P_{6}P_{7}$ with a maximum number of open stacks equal to 4.

3 The heuristic of minimal cost node

This section presents the heuristic of minimal cost node that uses the idea of visiting each arc in $G_p$ once. We begin with definitions of terms and introduce the heuristic.

**Definition 5** Arc contraction is an operation by removing an arc $(u, v)$ from a graph $G$ and joining nodes $u$ and $v$ into a single node. All other arcs incident to $u$ or $v$ become incident to the single node.

**Definition 6** A graph $G$ is isomorphic to a graph $H$ if there exists a one-to-one function $\Psi$ from $V(G)$, the vertex set of $G$, onto $V(H)$ such that arc $(u, v) \in E(G)$, the arc set of $G$, if and only if $(\Psi(u), \Psi(v)) \in E(H)$.

**Definition 7** A graph $G_p$ is called a **minor** of a graph $G$ if $G_p$ is isomorphic to a graph that can be obtained by zero or more arc contractions from $G$ in any order.

Consider the graphs $G$ and $G_p$ as illustrated in figure 13. The graph $G_p$ is obtained from $G$ by arc contraction of arcs $(m, n)$ and $(r, s)$. Thus, graph $G_p$ is a minor of $G$.

The heuristic of minimal cost node uses arcs in $G_p$, a minor of graph $G$, as the basis for determining the least number of arcs to be sequenced in order to close the node. The order in which the arcs are removed from $G_p$ follows the order in which the patterns are sequenced [12].
Figure 13: Illustration of a minor of $G$

Algorithm

To begin with the algorithm, we denote the following:

- $V$ is a set of nodes of $G_p$;
- $E$ is the set of arcs of $G_p$;
- $\Omega(k)$ is the degree of the node $k \in V$ excluding the arcs incident to $k$ already visited;
- SETV is an ordered set of nodes of $V$ yet to be sequenced. The nodes are ordered in an increasing order of degree $\Omega(k)$. A node $k \in V$ is in SETV if and only if $\Omega(k) \geq 1$;
- ARC is an ordered set of arcs of $G_p$ already visited;
- OPEN is the set of nodes in $V$ already open (at least one arc incident to this node is in set ARC and not all arcs incident to this node is in the set ARC);
- $s$ is the number of arcs in ARC (the number of arcs already visited);
- $\xi'$ is the maximum number of open stacks generated by the heuristic.

For a given graph $G_p$, we sequence the patterns to be cut in the following steps:
**Step 0:** Initialization. Label each node of $G_p$ with its corresponding degree $\Omega$. OPEN and ARC are empty. Set $\xi'$ and $s$ equal to zero.

**Step 1:** List in SETV all the nodes from $G_p$ in non-decreasing order of degree $\Omega$. Select two adjacent nodes $n$ and $m$ in $G_p$ that have the smallest degree $\Omega$. List these nodes in OPEN and arc $(n, m)$ in ARC. Delete arc $(n, m)$ in $G_p$ and with the resulting graph, update the new degree $\Omega$ of nodes $n$ and $m$. Set $\xi' = 2$ and $s = 1$.

**Step 2:** List in SETV all the nodes left in the graph. If SETV is empty, then STOP. Otherwise, find a node $p$ with the smallest degree $\Omega$ adjacent to the nodes in OPEN. If there are more than one adjacent nodes, choose a node $p$ adjacent to a node with the smallest degree $\Omega$) in OPEN. Add node $p$ in OPEN. Add the arcs incident to node $p$ in ARC and delete these arcs in the graph. With the resulting graph, label each node with its corresponding degree $\Omega$. If $\Omega(n) = 0$, then delete this node in OPEN and in the graph.

**Step 3:** List the nodes in OPEN excluding the nodes with degree $\Omega = 0$ and ARC including the new arcs. Generate the new $\xi'$ and $s$. Return to step 2.

To illustrate the heuristic with an example, consider the MOSP graph $G = G_p$ in figure 14 where each pattern has two piece types. $V = \{1, 2, \ldots, 9\}$ and $E = \{(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (2, 4), (2, 6), (3, 7), (3, 9), (4, 6), (5, 7), (5, 8), (8, 9), (6, 9), (7, 9), (8, 9)\}$.

Step 0: Figure 15 shows the initialization. OPEN = $\{\}$ and ARC = $\{\}$. $\xi' = 0$ and $s = 0$.

Step 1: SETV = $\{2, 3, 4, 8, 5, 6, 7, 9, 1\}$. Two adjacent nodes $n = 2$ and $m = 4$. OPEN = $\{2, 4\}$ and ARC = $\{(2, 4)\}$. $\xi' = 2$ and $s = 1$. The resulting graph is in figure 16.
Step 2: SETV = \{2, 4, 3, 8, 5, 6, 7, 9, 1\}. p = 6, add node 6 in OPEN. Add arcs (2, 6) and (4, 6) in ARC. Delete these arcs, the resulting graph is in figure 17.

Step 3: OPEN = \{2, 4, 6\} and ARC = \{(2, 4), (2, 6), (4, 6)\}. $\xi' = 3$ and $s = 3$. Return to step 2.

Step 2: SETV = \{2, 4, 6, 3, 8, 5, 7, 9, 1\}. p = 1, add node 1 in OPEN. Add arcs (1, 4), (1, 2), (1, 6) and delete these arcs in the graph. The resulting graph is in figure 18. The degree $\Omega(n = 2, 4) = 0$, delete nodes 2 and 4 in OPEN and in the graph.

Step 3: OPEN = \{1, 6\} and ARC = \{(2, 4), (2, 6), (4, 6), 1, 4), (1, 2), (1, 6)\}. $\xi' = 4$ and $s = 6$. Return to step 2.

Step 2: SETV = \{6, 3, 8, 5, 7, 1, 9\}. p = 9, add node 9 in OPEN. Add arcs (1,
Figure 16: Resulting graph after arc (2,4) is deleted

Figure 17: Resulting graph after arcs (2,4) and (2, 6) are deleted

9), (6, 9) and delete these arcs in the graph. The resulting graph is in figure 19.

Step 3: OPEN = \{1, 9\} and ARC = \{(2, 4), (2, 6), (4, 6), (1, 4), (1, 2), (1, 6), (1, 9), (6, 9)\}. $\xi' = 3$ and $s = 8$. Return to step 2.

Step 2: SETV = \{8, 3, 1, 9, 5, 7\}. $p = 3$, add node 3 in OPEN. Add arcs (1, 3), (3, 9) and delete these arcs in the graph. The resulting graph is left with nodes 1, 3, 5, 7, 8, and 9.

Step 3: OPEN = \{1, 3, 9\} and ARC = \{(2, 4), (2, 6), (4, 6), (1, 4), (1, 2), (1, 6), (1, 9), (6, 9), (1, 3), (3, 9)\}. $\xi' = 3$ and $s = 10$. Return to step 2.

Step 2: SETV = \{3, 1, 9, 8, 5, 7\}. $p = 7$, add node 7 in OPEN. Add arcs (1, 7), (3, 7), (7, 9) and delete these arcs in the graph. The resulting graph is left with nodes 1, 5, 7, 8, and 9.
Step 3: OPEN = \{1, 7, 9\} and ARC = \{(2, 4), (2, 6), (4, 6), (1, 4), (1, 2), (1, 6), (1, 9), (6, 9), (1, 3), (3, 9), (1, 7), (3, 7), (7, 9)\}. $\xi' = 4$ and $s = 13$. Return to step 2.

Step 2: SETV = \{7, 1, 9, 8, 5\}. $p = 5$, add node 5 in OPEN. Add arcs (1, 5), (5, 7), (5, 9) and delete these arcs in the graph. The resulting graph is left with nodes 1, 5, 8, and 9.

Step 3: OPEN = \{1, 5, 9\} and ARC = \{(2, 4), (2, 6), (4, 6), (1, 4), (1, 2), (1, 6), (1, 9), (6, 9), (1, 3), (3, 9), (1, 7), (3, 7), (7, 9), (1, 5), (5, 7), (5, 9)\}. $\xi' = 4$ and $s = 16$. Return to step 2.

Step 2: SETV = \{1, 5, 9, 8\}. $p = 8$, add node 8 in OPEN. Add arcs (1, 8), (5, 8), (8, 9) and delete these arcs in the graph. No nodes left in the graph.
Step 3: OPEN = \{ \} and ARC = \{(2, 4), (2, 6), (4, 6), (1, 4), (1, 2), (1, 6), (1, 9), (6, 9), (1, 3), (3, 9), (1, 7), (3, 7), (7, 9), (1, 5), (5, 7), (5, 9), (1, 8), (5, 8), (8, 9)\}. ξ' = 4 and s = 19. Return to step 2.

Step 2: SETV = \{ \}. Stop.

Finally, we have an optimal sequence of arcs (2, 4), (2, 6), (4, 6), (1, 4), (1, 2), (1, 6), (1, 9), (6, 9), (1, 3), (3, 9), (1, 7), (3, 7), (7, 9), (1, 5), (5, 7), (5, 9), (1, 8), (5, 8), (8, 9) with a maximum number of open stacks equal to four.

4 The exact method

To have an idea of how we determine the exact method for solving a given MOSP, consider the graph \( G \) in figure 20. In this example, we can easily identify the maximum clique of size 4. Considering only this clique and deleting all other arcs and nodes of the original graph, the resulting graph has all nodes with degree 3. Hence, by proposition 3, we have at least 4 open stacks at some time during the cutting process. Moreover, by proposition 2, 4 is the lower bound on the number of open stacks for the original graph. The exact method follows this way of exploring the graph. However, for a more complicated graph, we need to explore the structure of graphs which holds ”small” cliques as stated in section 2.

![Figure 20: A graph to illustrate lower bound](image)
bound approach. It includes the concept of equivalent solutions, the algorithm for heuristic arc contraction that gives a guaranteed lower bound for a given minor of $G$ and the process of the branch and bound scheme.

4.1 Equivalency proposition

The concept of the following proposition generates shortcuts in the enumeration for the optimal sequencing for a given $G_p$.

Let $N_b(i)$ be the "neighborhood" of a node $i$, the set of nodes that are adjacent to node $i$, and $\tilde{N}_b(i) = N_b(i) \cup \{i\}$ in $G_p$. Then, we say that nodes $i$ and $j$ are equivalent whenever $N_b(i) = N_b(j)$ or $\tilde{N}_b(i) = \tilde{N}_b(j)$.

**Equivalency proposition.** If $G_p$ has two different nodes, say $i$ and $j$, with $N_b(i) = N_b(j)$ or $\tilde{N}_b(i) = \tilde{N}_b(j)$, then from a given feasible solution of the problem where $p_i$ is not immediately followed by $p_j$ or vice versa, it is possible to construct a new sequence with $p_i$ and $p_j$ appearing consecutively. Moreover, the maximum number of open stacks by the new sequence is less than or equal to the original sequence.

**Proof.** We assume without loss of generality that $1, \ldots, i, \ldots, j, \ldots, n, i \neq j$ is a feasible sequence of the $n$ vertices of $G_p$, and $N_b(i) = N_b(j)$. When we sequence all the incident arcs to $i$, the set of open stacks associated to its neighboring vertices is the same set that would be opened by $p_j$. Therefore, it is possible to sequence $j$ immediately before $i$, since the total number of open stacks remains the same. A similar argument can be used to prove the case when $\tilde{N}_b(i) = \tilde{N}_b(j)$. □

With equivalency proposition, we can reduce equivalent nodes in $G_p$ to construct a new graph $G_p$ in determining a new lower bound. Furthermore, it can assist in the enumeration of the feasible solution schema.

In figure 20, part (a), an initial $G_p$ is given. Since nodes 2 and 4, nodes 5 and 3 are equivalent nodes, then we can reduce the initial $G_p$ as shown in part (b). From
part (b), we see another equivalence among nodes 7 and 3. Then, we get another graph after reduction method is applied in part (c). Observe that $|G_p|$ decreases from 8 to 5, and the number of arcs decreases from 15 to 7. In this instance, a level of hierarchy on the equivalence relations exists that must be realized. Nodes 3 and 7 are equivalent only after the first reduction is applied. Hence, we can sequence 3 and 7 but node 3 must precede node 7 in the sequence.

![Figure 21: Identification of equivalencies](image)

Therefore, if node $i$ and $j$ in $G_p$ with the condition that $N_b(i) = N_b(j)$ or $\tilde{N}_b(i) = \tilde{N}_b(j)$, for the sake of finding an optimal sequencing, we can delete node $i$ or node $j$ and all arcs incident to it.

4.2 Heuristic arc contraction

We need to determine the best lower bound by following the algorithm for heuristic arc contraction. We denote the following:

- $lb_p = \max \{|P_j|, j = 1, 2, ..., n\}$ where $|P_j| =$ degree of the pattern $P_j$. 

34
• $lb_d = 1 + \min \{\deg(n) \mid n \text{ is a node of } G_p\}$,

• $lb_c = \max \{K_i \mid K_i \text{ is a minor of } G_p\}$,

• $|G| = \text{order of } G_p$,

• $d_{\text{min}} = \text{minimum degree of the nodes in } G_p$.

Bounds $lb_p$ and $lb_d$ are instant. Since for $lb_p$, the number of unfinished stacks cannot be less than the maximum number of piece types of any pattern of a MOSP. $lb_d$ is derived from the fact that a piece type $p_i$, of minimum degree in $G_p$, is incident to all those $p_j$'s where $p_i$ and $p_j$'s belonging to the same pattern. Since some piece type has to be sequenced first at least this number of stacks will be opened.

The bound $lb_c$, as stated in [17], indicates that a lower bound is obtained if $G_p$ is a complete graph.

**Algorithm**

**Step 0** Initialization. Determine the $lb_p$, $lb_d$, $|G_p|$, and $lb$ where $lb = \max \{lb_p, lb_d\}$.

**Step 1** If $G_p$ is a complete graph, then $lb_c = lb$. Stop. If $|G_p| \leq lb$, then $lb_c = lb$.

Stop. Else, let $S$ be the node set of $G$ in non-decreasing order of degree $(n)$.

**Step 2** Choose an arc $(i,j)$ in $G_p$ that has the smallest node degree in $S$. Contract arc $(i,j)$ and with the resulting graph, if $d_{\text{min}} \geq 2$ and $d_{\text{min}} + 1 > lb$, then the new $lb = d_{\text{min}} + 1$. Return to step 1.

Consider the following patterns: $P_1 = \{p_1, p_3, p_6\}$, $P_2 = \{p_1, p_4, p_5\}$, $P_3 = \{p_2, p_4, p_7\}$, $P_4 = \{p_2, p_5, p_6\}$, $P_5 = \{p_3, p_4\}$. The corresponding graph is shown in figure 22 where nodes 1, 2, 3, 4, 5, 6, 7 represent piece types $p_1$, $p_2$, $p_3$, $p_4$, $p_5$, $p_6$, $p_7$ respectively. The value for the $lb_c$ is generated using the algorithm in the following way:

**Step 0:** Initialization. $lb_p = 3$, $lb_d = 3$, $|G| = 7$ and $lb = 3$. 35
Figure 22: A graph to illustrate the Heuristic arc contraction algorithm

Step 1: $G_p$ is not a complete graph. $|G_p| > lb$. $S = \{7, 3, 1, 2, 5, 6, 4\}$.

Step 2: Arc $(7, 2)$ is selected. Contract arc $(7, 2)$ and the resulting graph is shown in figure 23. The $dmin = 3$, thus the new $lb = 4$. Return to step 1.

Figure 23: A graph after first contraction

Step 1: $|G_p| = 6 > lb$. $S = \{2, 3, 1, 5, 6, 4\}$.

Step 2: Arc $(3, 4)$ is selected. Contract arc $(3, 4)$ and the resulting graph is shown in figure 24. The $dmin = 3$, thus the new $lb$ is still 4. Return to step 1.

Figure 24: A graph after second contraction

Step 1: $|G_p| = 5 > lb$. $S = \{1, 2, 4, 5, 6\}$.
Step 2: Arc (1, 6) is selected. Contract arc (1, 6) and the resulting graph is shown in figure 25. The $d_{min} = 3$, thus the new $lb$ is still 4. Return to step 1.

![Figure 25: A graph after third contraction = $K_4$](image)

Step 1: $G_p$ is a complete graph. Thus, $lb_c = 4$. Stop.

The set $S$ needs to be updated in every iteration to keep the lower bound as large as possible. For instance in a three-regular graph as shown in figure 26, if set $S$ is sorted just once, two different cliques, $K_3$ and $K_4$ are obtained as illustrated in figures 27 and 28, respectively.

![Figure 26: Initial graph $G=G_p$](image)

![Figure 27: Minor of $G$ generating $K_3$](image)

37
4.3 A branch and bound scheme

We use a greedy method for the branch and bound scheme. By first branching from a node 0 and searching the next most favorable node for branching in order to find an optimal solution for a given MOSP. In branching, we need to choose a node that result in the least possible number of incomplete piece types. For illustration, we denote the following:

- $S$ is the set of all piece types of the open stack problem;
- $S_i^u$ is the set of unfinished piece types at branching node $i$, $i = 0, 1, 2, ..., n$;
- $S_o^i$ is the set of currently open stacks at branching node $i$ where $S_o \subseteq S_u \subseteq S$;
- $LB_1$ is the maximum number of open piece types in each pattern;
- $v_j^k$ is the number of open stacks that results when going from node $j$ to $k$, a successor of $j$;
- $LB_2 = \min\{v_j^k\}$;
- $LB_3$ is the number of open stacks that resulted when branching from the predecessor of $j$. If $p$ is the predecessor of $j$ then the lower bound is $v_j^k$;
- $LB_j = \max\{LB_2, LB_3, LB_p\}$, $j > 0$, where $p$ is the predecessor of node $j$, and the overall lower bound for node $j$ including its predecessor node;
\( \begin{align*}
\bullet \ LB^0 &= \max\{LB_1, LB_2\}; \\
\bullet \ UP^j &= |S_u^i| = \text{upper bound on the number of open stacks at each node which is the total number of piece types of the problem still left to be cut.} \\
\bullet \ v^{\text{up}} \text{ is the value of the best solution so far for the open stacks problem. Initially,} \\
&\quad v^{\text{up}} = |S|.
\end{align*} \\
\end{align*} 

In the branching scheme, each node \( i \) will be identified by the sets \( S_u^i \) and \( S_o^i \) (see figure 29).

![Figure 29: Identification of a node in the branching scheme](image)

For each piece type \( p_j \in S_u^i \), create a new branching node say \( s \), with \( S_u^s = S_u^i \setminus \{p_j\} \) and \( S_o^s \) is the set of open unfinished stacks that is obtained from all the patterns containing piece type \( p_j \) that are sequenced.

We start the enumeration at node 0 with starting sets \( S_o^0 = \emptyset \) and \( S_u^0 = S = \{1, 2, ..., m\} \) (see figure 30). Then choose the most favorable node using a greedy criterion for branching. When any lower bound in node \( j \) is greater than or equal to \( v^{\text{up}} \), node \( j \) can be fathomed. When \( LB^j \geq UP^j \), node \( j \) can be fathomed and \( v^{\text{up}} \) is updated with \( LB^j \) if \( v^{\text{up}} > LB^j \).

Consider for the following instance of a MOSP: \( P_1 = \{p_1, p_2\}, P_2 = \{p_1, p_4\}, P_3 = \{p_1, p_3\}, P_4 = \{p_2, p_5\}, P_5 = \{p_3, p_4\}, P_6 = \{p_2, p_4\}, P_7 = \{p_3, p_5\}, P_8 = \{p_2, p_3\}, P_9 = \{p_1, p_5\}, P_{10} = \{p_2, p_6\} \). We start with node 0, where \( S_u^0 = \emptyset \) and
$S_u = \{p_1, p_2, p_3, p_4, p_5, p_6\}$. At node 0, $LB_1 = 2$, $UP^0 = 6$ and $v^u = 6$. Branching at node 0 produces 6 new nodes as shown in figure 31.

For these nodes we have:

$S^1_o = \{p_2, p_3, p_4, p_5\}$, $UP^1 = 5$, $v^1_o = 5$, $LB_3 = 5$;

$S^2_o = \{p_1, p_3, p_4, p_5, p_6\}$, $UP^2 = 5$, $v^2_o = 6$, $LB_3 = 6$;

$S^3_o = \{p_1, p_2, p_4, p_5\}$, $UP^3 = 5$, $v^3_o = 5$, $LB_3 = 5$;

$S^4_o = \{p_1, p_2, p_3\}$, $UP^4 = 5$, $v^4_o = 4$, $LB_3 = 4$;

$S^5_o = \{p_1, p_2, p_3\}$, $UP^5 = 5$, $v^5_o = 4$, $LB_3 = 4$;

$S^6_o = \{p_2\}$, $UP^6 = 5$, $v^6_o = 2$, $LB_3 = 2$.

Hence, at node 0, $LB_2 = \min \{v^i_o, i = 1, 2, ..., 6\} = 2$ and $LB^0 = \max \{LB_1, LB_2\} = 40$. 
Following the scheme we fathom nodes 1, 2, and 3 since the lower bound on these nodes equal the currently upper bound for the optimal value of the problem.

We then branch the most favorable node which is node 6 (see figure 32).

![Figure 32: Branching from node 6](image)

For these nodes we have:

- \( S^7 = \{p_2, p_3, p_4, p_5\}, v^7_o = 5; \)
- \( S^8 = \{p_1, p_3, p_4, p_5\}, v^8_o = 5; \)
- \( S^9 = \{p_1, p_2, p_4, p_5\}, v^9_o = 5; \)
- \( S^{10} = \{p_1, p_2, p_3\}, v^{10}_o = 4; \)
- \( S^{11} = \{p_1, p_2, p_3\}, v^{11}_o = 4. \)

Hence, at node 0, \( LB_2 = \min\{v^i_o, i = 1, 2, \ldots, 6\} = 4 \) and \( LB^6 = \max\{LB_2, LB_3, LB^0\} = 4 \) and \( UP^6 = 5 \). Since \( UP^6 > LB^6 \), then we can not fathomed node 6 but nodes 7, 8 and 9 are fathomed.

We then branch the next favorable node which is node 10 and for these nodes we have:

- \( S^{12} = \{p_2, p_3, p_5\}, v^{12}_o = 4; \)
- \( S^{13} = \{p_1, p_3, p_5\}, v^{13}_o = 4; \)
- \( S^{14} = \{p_1, p_2, p_5\}, v^{14}_o = 5; \)
\[ S_{o}^{15} = \{p_1, p_2, p_3\}, \quad v_{o}^{15} = 5. \]

Hence, at node 10, \( LB_2 = 4 \), \( LB^{10} = 4 \) and \( UP^{10} = 4 \). Since \( LB^{10} = UP^{10} \), any solution from this node and through has an optimal value of 4. We, therefore, update \( v^{up} \) to 4 and fathom node 10 and consequently, all its successors. Nodes 4 and 5 are also fathomed since their lower bounds equal to the current upper bound. Then the procedure ends with an optimal value of 4 and the optimal sequence follows by sorting the patterns that contain piece type \( p_6 \) first, the patterns that contain piece type \( p_5 \) is next and the patterns that complete any other uncompleted piece types until all piece types are completed. One possible solutions for the sequence is \( P_{10}P_{4}P_{7}P_{9}P_{1}P_{2}P_{3}P_{5}P_{6}P_{8} \) with a lower bound 4.

4.4 Exact Method: branch and bound approach

In practice, the following algorithm can take a long time to solve larger number of patterns that forms a large and overlapping cliques.

**Algorithm**

**Step 0** Determine the graph \( G \). All components that are trees are removed and call this removed set \( D \).

**Step 1** Execute the pre-processing on \( G \) to obtain \( G_p \) following section 2.

**Step 2** Execute the Heuristic of Minimal Cost Node to obtain \( \xi' \) and the sequence of patterns following section 3.

**Step 3** Execute the Heuristic Arc Contraction to get the lower bound \( lb_c \) following subsection 4.2.

**Step 4** If \( \xi' = lb_c \) (upper bound = lower bound) then the optimal solution is found in step 2. Else, look for the equivalence set \( E \) and implement the reduction process among the equivalent nodes as illustrated in subsection 4.1 to construct
a new graph $G_p$. Then execute the Heuristic Arc Contraction to get the new $lb_c$. After which, execute a branch and bound in order to obtain the optimal solution and insert the deleted equivalent nodes.

**Step 5** Insert the vertices of $D$ into the final sequence.

To illustrate the algorithm, consider the following instance of MOSP:

- $P_1 = \{p_1, p_3\}$, $P_2 = \{p_2, p_3\}$, $P_3 = \{p_3, p_5\}$, $P_4 = \{p_4, p_5\}$, $P_5 = \{p_5, p_6\}$,
- $P_6 = \{p_5, p_7\}$, $P_7 = \{p_5, p_{10}\}$, $P_8 = \{p_5, p_{12}\}$, $P_9 = \{p_5, p_{11}\}$, $P_{10} = \{p_6, p_9\}$,
- $P_{11} = \{p_6, p_{11}\}$, $P_{12} = \{p_6, p_8\}$, $P_{13} = \{p_6, p_{10}\}$, $P_{14} = \{p_6, p_7\}$, $P_{15} = \{p_7, p_8\}$,
- $P_{16} = \{p_7, p_{12}\}$, $P_{17} = \{p_7, p_{10}\}$, $P_{18} = \{p_8, p_9\}$, $P_{19} = \{p_9, p_{12}\}$, $P_{20} = \{p_9, p_{11}\}$,
- $P_{21} = \{p_8, p_{11}\}$, $P_{22} = \{p_{10}, p_{11}\}$, $P_{23} = \{p_{11}, p_{12}\}$, $P_{24} = \{p_8, p_{12}\}$, $P_{25} = \{p_8, p_{10}\}$,
- $P_{26} = \{p_{10}, p_{13}\}$, $P_{27} = \{p_{13}, p_{14}\}$, $P_{28} = \{p_{13}, p_{15}\}$, $P_{29} = \{p_{13}, p_{16}\}$.

![Figure 33: Graph that illustrates the branch and bound](image)

Step 0: The graph is shown in figure 30 where $p_i$ represents node $i$. Removed set $D = \{P_1, P_2, P_3, P_4, P_{26}, P_{27}, P_{28}, P_{29}\}$.
Step 1: Pre-processing. No redundancy occurs among the patterns. The graph $G$ shows two components as trees and when these trees are removed, we have $G_p$ as shown in figure 31. We have optimal sequences $P_1P_2P_3P_4$ and $P_{26}P_{27}P_{28}P_{29}$ for the two trees that are removed which will be inserted in the final sequence with the same maximum number of open stacks equal to 2.

![Figure 34: Graph $G_p$](image)

Step 2: Heuristic of Minimal Cost Node generates $\xi' = 6$, $s = 21$ and one optimal sequence $P_{19}P_{20}P_{23}P_{18}P_{21}P_{24}P_{10}P_{11}P_{12}P_3P_3P_8P_{22}P_{25}P_{13}P_7P_4P_6P_5P_{16}P_{17}$.

Step 3: Heuristic Arc Contraction generates $lb_c = 6$.

Step 4: Since $\xi' = lb_c = 6$, then we have an optimal solution in step 2.

Step 5: Insert the patterns from $D$, we have a final sequence $P_{19}P_{20}P_{23}P_{18}P_{21}P_{24}P_{10}$ $P_{11}P_{12}P_5P_5P_{22}P_{25}P_{13}P_7P_{14}P_6P_{15}P_{16}P_{17}P_{26}$, $P_{27}$, $P_{28}$, $P_{29}P_1P_2P_3P_4$ with a maximum number of open stacks 6.

5 Conclusion and future work

In this paper, the algorithms presented focus on the graphical model of a given MOSP. Each algorithm generates an optimal solution except for the heuristic of minimal cost node which generates the optimal sequence and its upper bound for
the maximum number of open stacks.

In some industrial environments such as glass, wood or steel industries, the optimal solutions using the heuristic are acceptable. The heuristic is a way to motivate a more efficient exact method for the MOSP with the utilization of a more profound computer programs, even though this is NP-hard problem [16].

In some real industrial settings, the maximum number of open stacks might be limited. And with this limitation, the optimal sequencing of the patterns may not satisfy the requirements. This arises to a more intricate study of two NP-hard problems, the cutting stock problem and the pattern sequencing problem. In [18], the formulation for the cutting and sequencing problem is suggested and in [19], the heuristic method for this problem is proposed.
REFERENCES


