PRE-PROCESSING OPERATIONS FOR THE MINIMIZATION OF OPEN STACKS PROBLEM

Horacio Hideki Yanasse
Universidade Federal de São Paulo

PRE-PROCESSING OPERATIONS FOR THE MINIMIZATION OF OPEN STACKS PROBLEM

Horacio Hideki Yanasse
Instituto Nacional de Pesquisas Espaciais
Avenida dos Astronautas 1758 – S.J.Campos, SP
horacio@lac.inpe.br

RESUMO

Dado um exemplar do problema de minimização de pilhas abertas com N padrões de corte existem $N!$ possíveis seqüências dentre as quais, provavelmente apenas algumas atingem o valor ótimo desejado. Achar uma dessas soluções ótimas requer geralmente um esforço computacional grande. O problema de minimização de pilhas abertas é, de fato, NP-árduo e os tempos de execução de vários algoritmos para resolvê-lo dependem do tamanho de N. Assim, operações de pré-processamento podem ser importantes, em particular, para reduzir o tamanho do problema quando possível. Neste trabalho revemos algumas operações de pré-processamento sugeridas na literatura e propomos algumas novas.

PALAVRAS CHAVE. Minimização de pilhas abertas, Pré-processamento, Redução de tamanho.

Área principal: OC Otimização Combinatória

ABSTRACT

Given an instance of the minimization of open stacks problem with N cutting patterns there are $N!$ feasible sequences for the problem among which, probably just a few achieve the desired optimal value. Finding one of these optimal solutions usually require a large computational effort. In fact, the minimization of open stacks problem is NP-hard and the execution times of many of the algorithms for solving it are dependent on the size of N. Therefore, pre-processing operations may be important, particularly to reduce the size of the problem when possible. In this paper we review some pre-processing operations suggested in the literature and we propose new ones.

KEYWORDS. Minimization of open stack problem, Pre-processing, Size reduction.

Main area: CO Combinatorial Optimization

1. Introduction

In some cutting settings, the sequence in which patterns are cut may be important. For instance, in wood hardboard cutting, items cut are piled up in stacks. Each stack is formed solely with items of the same type. A stack remains open until the last piece of the corresponding item type is cut. If the space available for the stacks is limited, it is desired to maintain a reduced number of open stacks during the cutting process. This is so because if the number of stacks increases beyond the available space, stacks must be removed in order to give space to the new stacks. Closed stacks can be removed definitively to another place or can be delivered to clients but open stacks, if removed, must be brought back later in the future to be completed when the corresponding item type is cut again in another pattern. This is not efficient and consumes more time and resources, therefore, we want to determine a sequence that minimizes the maximum number of open stacks during the cutting process. To this pattern sequencing problem we denote MOSP, the Minimization of Open Stack Problem.

Consider, for example, an instance of MOSP given in Table 1.

Table 1 – Data of Instance 1 with 6 patterns and 7 item types

Patterns	**Types of Items**
P_1	1 2
P_2	2 3
P_3	3 4
P_4	4 5
P_5	5 6
P_6	3 7

In this instance we have 6 patterns. Pattern P_1, for example, has item types 1 and 2. Observe that, in each pattern, we may have repetitions of an item type but, for MOSP, this is not a relevant information. Only the item types in a pattern are relevant.

Consider the cutting sequence P_1 P_2 P_3 P_4 P_5 P_6. The open stacks observed with this sequence are indicated in Table 2.

Table 2 – Open stacks observed in instance 1, for the sequence P_1 P_2 P_3 P_4 P_5 P_6.

Sequence patterns	**Open stacks**	**Observation**	**Maximum # of open stacks**
P_1	1 2	closed 1	2
P_2	2 3	closed 2	2
P_3	3 4		2
P_4	3 4 5	closed 4	3
P_5	3 5 6	closed 5 and 6	3
P_6	3 7	closed 3 and 7	2

The existing literature on the MOSP is not extense. We can separate them in works proposing heuristics for MOSP (see Yuen (1991, 1995), Yuen and Richardson (1995), Yanasse (1996), Faggioli and Bentivoglio (1998), Linhares et al (1999), Becceneri (1999), Ashikawa (2001), Linhares (2001), Oliveira and Lorena (2002a, 2002b), Yanasse et al (2002a), Becceneri et al. (2004), and works proposing exact methods to solve it (see Yanasse (1996, 1997a, 1997b), Faggioli and Bentivoglio (1998), Limeira (1998), Becceneri (1999), Yanasse and Limeira (1998, 2004), Yanasse et al. (1998, 2007), Becceneri et al. (2004)).

There are many similar but not equivalent optimization problems to MOSP (see Linhares and Yanasse, 2002). For instance, the Minimization of Order Spread Problem (see Foerster and Wäscher, 1998; Fink and Voss, 1999), the Matrix Bandwidth Minimization (see Chinn et al., 1982; Lim et al., 2006), the Minimization of the Discontinuities Problem (see Dyson and Gregory, 1974; Madsen, 1988).

On the other hand, there are equivalent problems to MOSP that arise in a context different from the cutting setting. According to Möhring (1990), the Gate Matrix Layout Problem, which is equivalent to MOSP (see) is equivalent to the One dimensional logic, and PLA folding in VLSI design, Interval Thickness, Node Search Game, Edge Search Game, Narrowness, Split Bandwidth, Graph path-width, Edge Separation, and Vertex Separation in graph theory. So any results derived for MOSP is also of great interest for those facing all these problems.

Recently, MOSP was focused in the Constraint Modeling Challenge 2005 (see The Fifth Workshop on Modelling and Solving Problems with Constraints), increasing the interest on this problem by the Operations Research scientific community.

In this work we focus on pre-processing operations for the MOSP problem. MOSP is NP-hard, and in hard combinatorial problems where the execution time of an algorithm to get a solution increases very quickly with the size of the problem, it is always of great interest performing problem data pre-processing to verify whether we can accomplish simplifications or reductions of the problem.

2. Pre-processing operations

Pre-processing 1: Is the problem decomposable?

One of the initial verification we can make is check whether the instance of MOSP is decomposable because of the existence of independent clusters of patterns. This verification can be performed using graphs (see Yuen and Richardson, 1995). Consider, for example, the instance of MOSP given in Table 3.

Table 3 – Data of Instance 2 with 5 patterns and 8 item types

Patterns	**Types of Items**
P_1	1 7
P_2	3 6 7
P_3	7 8
P_4	5
P_5	2 4

A pattern connection graph can be built as follows: each pattern corresponds to a vertex and, there exists an arc linking two vertices in this graph if and only if the patterns corresponding to these two vertices share at least one item type in common. So, if in this graph, two nodes are not adjacent, then their corresponding patterns are not connected, therefore, the stacks they form when they are cut have no relation to each other.

The pattern connection graph of instance 3 is presented in Figure 1. We can observe that patterns P_1, P_2 and P_3 have item types in common, while the items in patterns P_4 and P_5 just appear in these patterns. From the graph we identify 3 clusters of patterns (items) that can be treated independently for the MOSP.

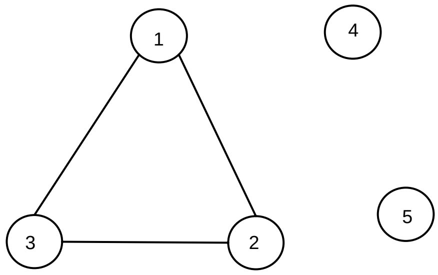

Figure 1 – Pattern connection graph of instance 3

From now on, we assume that we have performed the pre-processing 1 in our instance and we are considering patterns that form a single cluster.

Pre-processing 2: Are there dominant patterns?

A dominated pattern is one whose item types are all contained in another pattern of the problem. Dominated patterns can be withdrawn from the problem, since they can be sequenced afterwards, immediately after its dominant pattern is sequenced, without increasing the number of open stacks.

Pre-processing 3: Is this an easy solvable instance of MOSP?

By easy cases of MOSP we mean those instances that can be solved exactly by polynomial algorithms.

Some easy cases can be easily identified using the graph version problem of MOSP. Yanasse (1997c) showed that MOSP can be viewed as an arc traversing problem in a graph. In the MOSP graph, nodes correspond to item types and there exists an arc connecting two nodes if and only if there exists at least a pattern where the two item types appear together. Observe that the MOSP graph is different from the pattern graph connection presented previously.

In the MOSP graph, we want to determine a sequence to traverse all the arcs so that we minimize the maximum number of nodes simultaneously open during the traversing. A node is open when an arc incident to it is traversed for the first time and, it is closed when all arcs incident to it are traversed.

MOSP graphs that are trees, one-trees, complete graphs, are easy to solve and to identify (see, Yanasse, 1996, 1997b). Many other polynomially solvable cases can be derived using special structured MOSP graphs. However, verifying whether an instance falls in one of these cases may require a computational effort that may be not worthed; it may be better to solve the original problem itself by some general method.

Pre-processing 4: Is it possible to reduce the number of items?

If there exists two adjacent nodes i and j in the MOSP graph, both having degree 2 then, to any feasible order in which the nodes are closed, it is possible to construct an alternative corresponding sequence, whose number of open nodes is less than or equal to the one obtained with the original sequence and where node j is closed immediately before node i is opened or node i is closed immediately before node j is opened (see Yanasse et al, 2002b). In other words, for the sake of finding a solution to MOSP, it is possible to collapse the arc (i, j) of the MOSP graph, obtaining a reduced graph with one node less. In other words, we downsized the problem in term of total number of item types.

Pre-processing 5: Is it possible to reduce the number of patterns?

Consider the MOSP instances 3 and 4 given in Tables 4 and 5, respectively.

Table 4 – Data of Instance 3 with 8 patterns and 6 item types

Patterns	**Types of Items**
P_1	1 2 4
P_2	2 4 5
P_3	3 4
P_4	1 3 5
P_5	1 4 5
P_6	5 6
P_7	4 6
P_8	2 6

Table 5 – Data of Instance 4 with 6 patterns and 6 item types

Patterns	**Types of Items**
P_1	1 2 4
P_2	2 4 5 6
P_3	3 4
P_4	1 3 5
P_5	1 4 5
P_6	5 6

The MOSP graph corresponding to instances 3 and 4 is presented in Figure 2. We observe that different MOSP instances generate the same MOSP graph. The optimal solution to the MOSP graph is the same for both problems. Therefore, these instances are computationally equivalent in terms of their solution. It is worth remembering that from the optimal solution of the MOSP graph we can determine, in polynomial time, a corresponding optimal solution to the original instances 3 and 4.

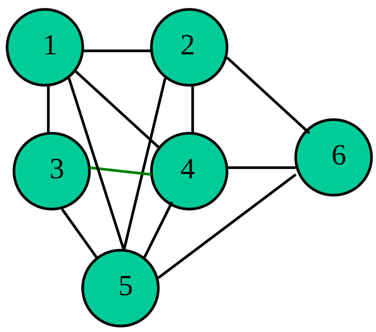

Figure 2: MOSP graph of instances 3 and 4

This equivalence of solutions led us to find alternative smaller instances of MOSP that produces the same MOSP graph of a given instance. Since MOSP is NP-hard, the smaller the alternative instance, the better, since the computational time to solve the problem increases with its size.

In pre-processing 5 we propose an algorithm, denoted by REDUCE, that generates an alternative potentially smaller sized MOSP instance that is equivalent to the original instance in terms of the MOSP graph.

Algorithm REDUCE

Given: Instance of MOSP

Initialization: Generate the MOSP graph.

Classify all arcs as type 1.

Main Step: While there are arcs type 1 in the graph do

BeginWhile

Select a node, say *i*, whose degree considering only arcs type 1 is different than zero;

Determine the largest clique containing node *i* and arcs type 1 incident to it. Store the clique (pattern);

Classify all arcs of this clique as type 2.

EndWhile.

Illustration of Algorithm REDUCE

Let us apply algorithm REDUCE to instance 3 given in Table 4. The corresponding MOSP graph is given in Figure 2.

In the first pass of the "while loop" of REDUCE, we have that all arcs are type 1. Let us select, for instance, node 6. The largest clique containing node 6 and arcs type 1 incident to it is the complete subgraph composed of nodes {2, 4, 5, 6}, that is pattern P_1 equal to {2 4 5 6}. The resulting graph, after classifying the arcs of this clique as type 2 arcs, is illustrated in Figure 3.

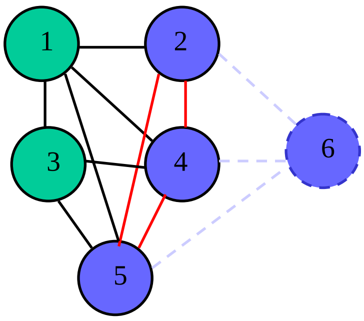

Figure 3 – Graph after first iteration of the "while loop" in REDUCE

Let us select now any node having at least one arc of type 1 incident to it say, for instance, node 3. The largest clique containing node 3 and arcs type 1 incident to it is the complete subgraph composed of nodes {1, 3, 4, 5}, that is pattern P_2 equal to {1 3 4 5}. The resulting graph, after classifying the arcs of this clique as type 2 arcs, is illustrated in Figure 4.

Let us select now any node having at least one arc of type 1 incident to it say, for instance, node 2. The largest clique containing node 2 and arcs type 1 incident to it is the complete subgraph composed of nodes {1, 2, 4, 5}, that is pattern P_3 equal to {1 2 4 5}. In the resulting graph, after classifying the arcs of this clique as type 2 arcs, there are no arcs of type 1 left and we are finished. The equivalent MOSP problem obtained is given in Table 6.

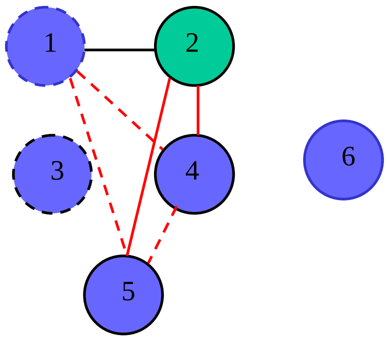

Figure 4 – Graph after second iteration of the “while loop” in REDUCE

Table 6 – Equivalent MOSP problem to Instances 3 and 4 with 3 patterns and 6 item types

Patterns	**Types of Items**
P_1	2 4 5 6
P_2	1 3 4 5
P_3	1 2 4 5

So, we were able to reduce an instance of MOSP with 8 patterns to an equivalent one with only 3 patterns.

Algorithm REDUCE tries to determine an equivalent MOSP instance with the smallest possible number of patterns. For each arc incident to a node i in the MOSP graph we must have at least one pattern to represent that arc in the problem. To represent $n > 1$ arcs incident to a node i with a single pattern, all the nodes incident to these n arcs must be in a single clique of the graph. Therefore, the minimum number of patterns necessary to represent all arcs incidents to a node i in the MOSP graph is equal to the minimum number of cliques required to cover all arcs incident to node i. Algorithm REDUCE tries to determine a set of cliques with the minimum possible cardinality that covers each node of the MOSP graph.

Pre-processing 6: Reducing the number of nodes

Let A_i be the set of nodes in the MOSP graph that are adjacent to node i. Let i and j be nodes of a MOSP graph such that $\{A_i \cup \{i\}\} = \{A_j \cup \{j\}\}$. Then there exists always an optimal solution to MOSP where i and j are closed one immediately after the other. Using this result, we can reduce the number of item types, by considering nodes i and j as “twin” nodes, that is, they are two but they are always together as if they are a single node (they are open together, they are closed together).

Consider a MOSP graph instance given in Figure 5. In the MOSP graph of instance 5, $\{A_1 \cup \{1\}\}$ and $\{A_8 \cup \{8\}\}$ are equal. Therefore, items 1 and 8 can be consolidated into a single “twin-node” (1&8) (see Figure 6).

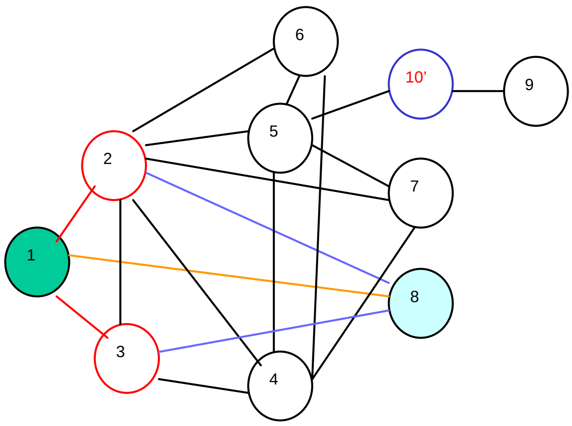

Figure 5 – Instance 5 of MOSP

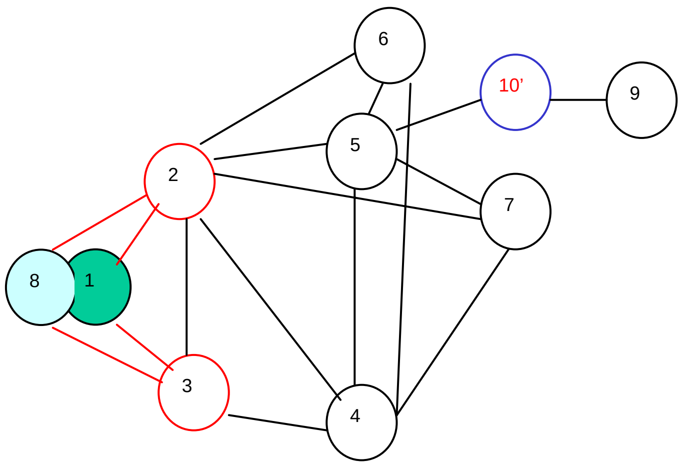

Figure 6 – Reduced instance of MOSP

Without loss of generality, let us consider now that the MOSP instance has been reduced using algorithm REDUCE (pre-processing 5).

Pre-processing 7: Reducing the number of patterns

Let A_i be the set of nodes in the MOSP graph that are adjacent to node *i*. Let *i* and *j* be nodes of a MOSP graph such that $A_i = A_j$. Then there exists always an optimal solution to MOSP where *i* and *j* are closed one immediately after the other. Suppose we have an instance of MOSP obtained by algorithm REDUCE. If such nodes exist then, there exists always an optimal solution to MOSP where the pattern containing *i* and the pattern containing *j* are sequenced one immediately after the other, therefore, we can consider these patterns as a single "twin pattern", or in other words, they stick together in an optimal solution to the problem.

Consider the MOSP graph instance 5 of Figure 5. In the MOSP graph of instance 5, A_6 and A_7 are equal. Therefore, patterns P^6 = {6 2 5 4} and P^7 = {7 2 5 4} that are part of the patterns obtained with algorithm REDUCE, can be consolidated into a single "twin-pattern" $P^{6\&7}$ = {6 2 5 4} & {7 2 5 4}.

3. Concluding remarks

In this work we reviewed some pre-processing operations that can be performed to reduce the size of MOSP instances. Pre-processing 1 and 2 were proposed in the literature, pre-processing operations 3, 4, 6 and 7 are derived from earlier results of the literature, and algorithm REDUCE in pre-processing 5 is a new contribution being presented in this article. We believe that all pre-processing operations, and in particular pre-processing 5, are a relevant contribution for those interested in solving MOSP because the size reduction accomplished with these pre-processing operations can be significant. For instance, in the illustration presented in the previous section, just applying pre-processing 5, we were able to reduce the size of an instance from 8 to 3 patterns.

The costs of the pre-processing operations presented in section 2 are not high. If M is the number of item types and N is the number of patterns, the computational complexity of all the pre-processing operations is $O(M^2N^2)$.

Acknowledgements

This work is partially financed by CNPq and FAPESP.

References

Ashikaga, F. M. Um método frugal para o problema de minimização de pilhas abertas, *Dissertação (Mestrado em Ciência no Curso de Engenharia Eletrônica e Computação na Área de Informática)* , ITA/CTA, São José dos Campos, 2001.

Becceneri, J. C. O problema de seqüenciamento de padrões para a minimização do número máximo de pilhas abertas em ambientes de cortes industriais, *Tese (Doutorado em Ciências no Curso de Engenharia Eletrônica e Computação na Área de Informática)*, ITA/CTA, São José dos Campos, 1999.

Becceneri, J.C., Yanasse, H.H. and Soma, N.Y. (2004), A method for solving the minimisation of the maximum number of open stacks problem within a cutting process. *Computers and Operations Research* 31(14): 2315-2332.

Chinn, P., Chvatalova, J., Dewdney, A.K. and Gibbs, N.E. (1982), The bandwidth problem for graphs and matrices – a survey. *Journal of Graph Theory* 6, 223-254.

Dyson, R.G. and Gregory, A.S. (1974), The cutting stock problem in the flat glass industry, *Operational Research Quarterly* 25(1), 41-53.

Faggioli, E. and Bentivoglio, C.A. (1998), Heuristic and exact methods for the cutting sequencing problem, *European Journal of Operational Research*, **110**(3), 564-575.

Fink, A. and Voss, S. (1999), Applications of modern heuristic search methods to pattern sequencing problems. *Computers & Operations Research* 26, 17-34.

Foerster, H. and Wäscher, G. (1998), Simulated annealing for order spread minimization in sequencing cutting patterns. *European Journal of Operational Research* 110, 272-281.

Lim, A., Rodrigues, B. and Xiao, F. (2006), Heuristics for matrix bandwidth reduction. *European Journal of Operational Research* 174, 69-91.

Limeira, M. S. Desenvolvimento de um algoritmo exato para a solução de um problema de seqüenciamento de padrões de corte, *Dissertação (Mestrado em Computação Aplicada)*, Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 1998.

Linhares, A. Industrial pattern sequencing problems: some complexity results and new local search models**,** *Tese (Doutorado em Computação Aplicada)*, Instituto Nacional de Pesquisas Espaciais**.** São José dos Campos, 2001.

Linhares, A. and Yanasse, H.H. (2002), Connections between cutting-pattern sequencing, VLSI design, and flexible machines. *Computers and Operations Research*, vol.29(12): 1759-1772.

Linhares, A., Yanasse, H.H. and Torreão, J.R.A. (1999), Linear gate assignment: a fast statistical mechanics approach. *IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems* 18(12):1750-1758.
Madsen, O.B.G. (1988), An application of travelling-salesman routines to solve pattern-allocation problems in the glass industry, *Journal of the Operational Research Society* 39/3, 249-256.
Möhring, R.H. (1990), Graph problems related to gate matrix layout and PLA folding. *Computing*, 7, 17-51,
Oliveira, A.C.M. and Lorena, L.A.N. (2002), A Constructive Genetic Algorithm for Gate Matrix Layout Problems. *IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems* 21(8), p. 969-974
Oliveira A. C. M. and Lorena, L. A. N. 2-opt population training for minimization of open stack problem In Bittencourt, G. and G. L. Ramalho (Eds.), *Advances in Artificial Intelligence*, Springer Lecture Notes in Artificial Intelligence Series vol. 2507, pp. 313-323 , 2002.
The Fifth Workshop on Modelling and Solving Problems with Constraints, IJCAI 05, Edinburgh (2005). http://www.dcs.st-and.ac.uk/~ipg/challenge/index.html, and http://www.dcs.st-and.ac.uk/~ipg/challenge/ModelChallenge05.pdf
Yanasse, H.H. (1996), Minimization of open orders - polynomial algorithms for some special cases. *Pesquisa Operacional* 16(1):1-26.
Yanasse, H.H. (1997a), On a pattern sequencing problem to minimize the maximum number of open stacks, *European Journal of Operational Research*, 100, 454-463.
Yanasse, H.H. (1997b), An exact algorithm for the tree case of the minimization of open orders problem. *XXIX Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, Salvador, BA.
Yanasse, H.H. (1997c), A transformation for solving a pattern sequencing problem in the wood cut industry". *Pesquisa Operacional*, 17(1), 57-70.
Yanasse, H.H., Becceneri, J.C. and Soma, N.Y. (1998), Um algoritmo exato para um caso especial do problema de sequenciamento de padrões. *XXX SBPO*, Curitiba, PR.
Yanasse, H.H., Becceneri, J.C, and Soma, N.Y. (2002a), A heuristic to solve a pattern sequencing problem based on partial orderings. *XXII Encontro Nacional de Engenharia de Produção / VIII International Conference on Industrial Engineering and Operations Management*, Curitiba, PR.
Yanasse, H.H., Becceneri, J.C, and Soma, N.Y. (2002b), Ordenamento parcial para reduzir o espaço de busca de uma solução ótima para um problema de seqüenciamento de padrões. *XXXIV Simpósio Brasileiro de Pesquisa Operacional*, Rio de Janeiro, RJ.
Yanasse, H.H., Becceneri, J.C. and Soma, N.Y. (2007), Um algoritmo exato com ordenamento parcial para solução de um problema de programação da produção: experimentos computacionais. *Gestão & Produção* 14(2):353-361.
Yanasse, H.H. and Limeira, M.S. (1998), Um algoritmo branch-and-bound para resolver um problema de sequenciamento de padrões, *XXX Simpósio Brasileiro de Pesquisa Operacional*, Curitiba, Paraná.
Yanasse, H.H. and Limeira, M.S. (2004), Refinements on an enumeration scheme for solving a pattern sequencing problem. *International Transactions in Operations Research* **11**, 277-292 (2004).
Yuen, B.J. (1991), Heuristics for sequencing cutting patterns, *European Journal of Operational Research* 55, p.183-190.
Yuen, B.J. (1995), Improved heuristics for sequencing cutting patterns, *European Journal of Operational Research* 87, p.57-64.
Yuen, B.J. and Richardson, K.V. (1995), Establishing the optimality of sequencing heuristics for cutting stock problems, *European Journal of Operational Research*, 84, 590-598.