UMA HEURÍSTICA PARA REDUÇÃO DO NÚMERO DE CICLOS DA SERRA NO CORTE DE CHAPAS

HORACIO H. YANASSE
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS
LABORATÓRIO ASSOCIADO DE COMPUTAÇÆO E MATEMÁTICA APLICADA
SÆO JOSÉ DOS CAMPOS, SÆO PAULO
BRASIL

REGINALD G. HARRIS
ALAN S.I. ZINOBER
DEPARTAMENTO DE MATEMÁTICA APLICADA E COMPUTACIONAL
UNIVERSIDADE DE SHEFFIELD
SHEFFIELD, INGLATERRA

RESUMO
O custo operacional de uma máquina de serra está diretamente relacionado com o número de ciclos da máquina. Quanto maior o número de ciclos da máquina, mais tempo de máquina é utilizado e um maior custo operacional é incorrido. Em diversos ambientes produtivos, tais custos podem ser significativos comparados com os custos da própria matéria prima sendo cortada. Assim, para minimizar os custos totais de produçÆo é necessário reduzir as sobras de material resultantes dos cortes realizados bem comn reduzir o número de ciclos da máquina. Neste trabalho prop_e-se um procedimento heurístico que tenta balancear as perdas com sobras de material e o tempo de máquina. O método opera de maneira sequencial redefinindo problemas de corte auxiliares que sÆo utilizados para gerar padr_es de corte que podem ser repetidos diversas vezes no problema original. Numa serra, várias chapas que obedecem um mesmo padrÆo de corte podem ser cortadas de uma única vez economizando-se o tempo de utilizaçÆo da máquina. Os padr_es de corte podem ser gerados utilizando-se qualquer otimizador padrÆo para problemas de corte de estoque. Testes computacionais foram realizados com alguns problemas testes e os resultados obtidos foram bastante promissores.

ABSTRACT
The reduction of saw cycles decreases the machine cutting costs which can be as important as the reduction of waste in many cutting settings. We propose a procedure that attempts to balance the waste and the machine time. It wnrks by sequentially redefining cutting problems which are used to generate patterns that can be repeated many times in the nriginal problem. The patterns can be generated using a standard optimiser for cutting stock problems. Limited computational tests performed in a few test problems showed promising results.

Keywords: cutting stock problem; cycle reduction; minimization of the number of patterns; heuristics

Acknowledgements: This work was partially funded by SERC GRANT GR/F 68942 and CNPq grant number 502405/91-0.

1. INTRODUCTION

For many customers the reduction of the number of cycles of the saw in a cutting stock problem is of great importance, that is machine costs are significant compared with the costs of material.

Many practical microcomputer based cutting stock optimiser algorithms generate cutting patterns in sequence, until all required parts are cut (see Dyckhoff et al. 1985; Hinxman, 1980; Yanasse et al. 1991 and 1992). If the optimiser algorithm does not account at all for a complete solution when establishing costs of a pattern, the resulting solutions might be quite poor.

Consider, for instance, a furniture manufacturing setting where rectangular panels of different sizes and quantitites are to be cut from large rectangular boards. The objective of the cutting stock problem is still to produce a solution of lowest possible cost. A simple tradeoff function between the number of cycles and waste wnuld be: 1 cycle = 1 sheet (or some other ratio), that is, we are prepared to accept the use of one extra sheet of material (board) if it reduces the number of cycles by 1. In this case, our objective function is to minimise the Number of boards + Number of cycles. Therefnre, we are also assuming that our problem has only nne type of board from where panels are tn be cut.

The number nf cycles of the machine is reduced by decreasing the total number of different cutting patterns in a solution. If the number of different cutting patterns is reduced in a complete solution, this means that some patterns are being repeatedly cut in nrder to produce all the required parts. The saw machine can cut more than one board at a time; it can cut a stack of boards up to a maximum height. Repeated patterns, hence, can be cut in a single saw cycle reducing the machine cutting costs.

The number of times a pattern is repeated in a solution is denoted its run quantity. For many users it is desirable that the run quantity be equal to a multiple of the maximum stack height. This implies that if the maximum stack height is 5 boards, the run quantities must be 5, 10, 15, and so on. If the run quantity nf a pattern is 21 (5 cycles with last cycle not full) the user might prefer a run quantity of 20 (under-production) or even 25 (over-production).

We are assuming here an exact quantity problem, that is, all demands have to be exactly met. Hence, if for instance, the run quantity is 21 we will have 5 cycles with the last cycle not full.

In this paper we present a heuristic procedure for solving the reductinn of the saw cycles problem. The method proposed is quite simple. Since it is desirable tn have as little cycles as possible while maintaining the number of boards as low as possible, we try tn define patterns containing only panels with high demands so that they can be repeated many times. This discriminative pattern building strategy is maintained as long as the waste of the patterns obtained are not large enough.

2. ALGORITHM FOR THE REDUCTION OF SAW CYCLES

The basic idea used is quite simple. From the original problem $\mathbf{P_0}$, we define a surrogate problem $\mathbf{P_1}$ with the same panels as the original problem $\mathbf{P_0}$ but with the demands adjusted according to a factor $\mathbf{F_0}$. If $d_1,d_2,...,d_n$ are the requiremems for panels 1, 2, ..., n, respectively, in problem $\mathbf{P_0}$, problem $\mathbf{P_1}$ is obtained by dividing $d_1,d_2,...,d_n$ by $\mathbf{F_0}$ and taking only the integer part. Any cutting pattern feasible to problem $\mathbf{P_1}$ is also feasible to $\mathbf{P_0}$ and, in additinn, can be cut at least $\mathbf{F_0}$ times without having over-production.

By using a pattern generator optimiser algorithm, patterns are generated for problem $\mathbf{P_1}$ in sequence and if the pattern is "good" enough it is accepted. This pattern will be part of the overall solution for problem $\mathbf{P_0}$ having a stack height of $\mathbf{F_0}$.

When a pattern generated for problem $\mathbf{P_1}$ is not "good" enough then we reduce conveniently the panel requirements of problem $\mathbf{P_0}$ according to the pattern(s) chosen and the stack size $\mathbf{F_0}$. If no pattern was chosen, no reduction is made. A new stack size $\mathbf{F_0}$ is defined and the process is repeated as from the beginning. The process ends when all panel requirements are met.

We next describe in detail what we mean by a "good" pattern and how we define the stack size F_0.

2.1 Definition of "good" pattern

Our objective is to minimise the number of cycles and to get a solution with the least possible waste. The criterion for deciding whether a pattern is "good or not" must take into consideration the stack size. It is obvious that a pattern, which is repeated many times, will produce a poor overall solution if it has lots of waste. On the other hand, a "bad" pattern can be acceptable if it is not going to be repeated many times since its contribution to the overall solution will be minimal.

We try to incorporate in our algorithm, hence, the condition that patterns which are to be repeated many times must have a high efficieocy. If the stack size is high we accept a pattern only if its efficiency (total panel area in pattern/ board area) is greater than a cutoff value "effref".

The definition of the cutoff value "effref" is quite important. In the problems we tested we have been usiog as "effref" the value given by the best previously known overall results of the problems. Tests using different cutoff values showed that the final solutions are sensitive to this value.

Hence, a multiple pass scheme is suggested where this parameter is updated according to the previous pass results. For the initial pass, when this cutoff value is unknown, a fixed predetermined value is used.

We implemented an algorithm with 5 passes that produced quite good results in the tests performed. In each one of the passes, a different target efficiency "effref" was used. The target efficiencies were defined as follows:

Pass 1 effref = 80%. Call the efficiency achieved
E1. (e.g., E1 = 83%)

Pass 2 effref = 90%. Call the efficiency achieved
E2. (e.g., E2 = 86%)

Pass 3 effref = max {E1,E2}. Call this E3 (e.g., E3 = 86%)
and the efficiency achieved E4.
E5 = max {E3,E4}.
E6 = min {E5+2.5%,100%}.

Pass 4 effref = E6. Call the efficiency achieved E7.
E8 = max {E5,E7}.
IF (E8 > E5)
E9 = min {E8+2.5%,100%}

ELSE

E9 = min {E5+5%,100%}.

Pass 5 effref = E9.

The best overall solution of the 5 passes is kept.

The reasoning for choosing such efficiency reference values is the following. The first two passes are just "guesses"; we do not know anything about what we can expect from the solution of the problem. The higher reference value of the second pass is an attempt of the algorithm of driving the search towards a potential good solution with higher efficiency. The third pass is an opportunity to correct the previous passes guesses. In many of the cases tested, after this pass a good solution was already generated. Passes 4 and 5 are refinements. They just attempt to identify possible better efficiency solutions to the problem.

As the stack size decreases, we can afford to be less restrictive with respect to the acceptable patterns' efficiencies for their contribution to the overall solution waste will be smaller.

We have been using the following cutoff values in our problem tests with relative good success:

cutoff value = effref * correctinnfactor;

where

correctionfactor = 1, if stack size is greater or equal to the maximum stack allowed per cutting cycle (MAXSTACK),

correctionfactor = 1.0 (MAXSTACK + 1 - stacksize)*0.02, if stack size is greater than 1 and smaller than the maximum stack allowed per cutting cycle,

correctionfactor = 0, if stack size is equal to 1. When the stack size is equal to one, any generated pattern is chosen even if it has a low efficiency for otherwise, we would be unable to satisfy all panel requirements exactly at the end.

The correctionfactor may be adjusted when the relative weights of boards and cycles differ in the objective function. For instance, if a sheet of board is worth three cycles, then we could be even more restrictive in accepting patterns that are repeated many times, therefore, enforcing our priority nn the number of boards rather than the number of cycles.

2.2 Definition of the stack height F_0

The definition of the stack height F0 is based on the three largest requirements of the panels and the maximum stack size per cutting cycle (MAXSTACK).

Let max_1, max_2, max_3 be the 1st largest, the 2nd largest and the 3rd largest requirements of the panels in problem P_0. Let maxdemand be the smallest multiple nf MAXSTACK which is greater or equal to $max_1/2$; $maxdemand_1$ be the largest multiple of MAXSTACK smaller or equal to max_2 and $maxdemand_2$ be the largest multiple of MAXSTACK smaller or equal to max_3. Therefore, if F_0 is equal to maxdemand, then the demand for the panel that has the largest requirements in problem P_0 will be equal to 1 in problem P_1. Observe that

this is the smallest value for F_0, multiple of MAXSTACK, that will still keep the demand of the panel having the largest requirement in problem P_0 equal to 1 in problem P_1. The first value of F_0 is maxdemand.

The subsequent values of F_0 are defined as follows: every time after a stack size F_0 is tried and at least one pattern is accepted, the next value of F_0 is $maxdemand_1$ of the new updated P_0 problem (obtained from the previous P_0 with the panel requirements reduced accordingly to the accepted patterns and corresponding stack size); if no patterns is accepted for the current stack size F_0, the next value of F_0 is $maxdemand_2$, if the current value is greater than $maxdemand_2$ and, the current value - MAXSTACK, otherwise. This process goes on until the stack size F_0 reaches the MAXSTACK value. In this case the next stack size F_0 is one less the current one.

As we previously discussed, it is important that we get "good" patterns when the stack size is large. We require the use of a good generating pattern program in conjunction with our proposed scheme. Many generating pattern programs sort the panels according to some criteria (see Yanasse et al, 1991) and impose that the first panel in the sorted list (a particular, perhaps odd sized - large, long, wide - panel) has got to be in the pattern being generated. This might lead only to "bad" patterns in terms of the criterion we are using. In this case, it is important to have some mechanism that relaxes this imposition so that other, perhaps more promising patterns are generated.

The correcting factor we propose is quite arbitrary. From the limited tests we performed we observed that the solution is also sensitive to this parameter value. A reduction of the efficiency by the factor of (n-1)/n seems to produce also good results for small demand problems, where n is the current F_0 value and n is smaller than MAXSTACK.

For small sized problems we observed that some panels may ideally belong to patterns with smaller run quantities although their requirements compared to the others in the problem are large. They, therefore, tend to be inserted in patterns with large run quantities when applying our proposed method. A perturbation of the method above might be tried, consisting of temporarily inverting the order of the run quantities. Suppose, for instance, that some panels may ideally belong to patterns with a run quantity of 4 but are partially used in the run quantity of 5. A perturbation of the method would be to force the first pattern to have a run quantity of 4 (that is the dividing constant of the demands is 4) before starting at 5. Several passes could be tried, forcing the first pattern to have run quantities of 1,2,3,4 etc. and keeping the best overall solution.

The scheme proposed to minimise the number of cycles can be very advantageous mainly when the requirements of the panels are large. By dividing the demands by a constant the new problem can be quite small sized, hence, getting a solution for this problem is quite fast. If good patterns are obtained, the overall performance of the method in terms of computer time can be very good.

3. COMPUTATIONAL RESULTS OF THE REDUCTION OF SAW CYCLES ALGORITHM

The algorithm for reducing the number of saw cycles has been applied to a few test problems and a typical example of the type of results obtained is presented in Tables 1 and 2.

Table 1

Summary of previously obtained results

(using a comercial pattern generation optimiser)

Problem	No. of Boards	No. of Patterns	No.of Cycles
1	126	49	49
2	130	77	78
3	260	4	29
4	7	5	5
5	17	10	10
6	21	11	12

Total boards+cycles = 744

Table 2

Summary of the results obtained by saw reduction algorithm

Problem No.	of Boards	No. of Patterns	No.of Cycles
1	131	6	15
2	135	6	15
3	260	5	28
4	7	4	4
5	19	6	6
6	22	7	7

Total boards+cycles = 649

As can be seen the results are promising with an average improvement over the previous solutions around 13%. The best possible value of the function (Total boards+cycles) for these problems, taking into consideration the area of the panels, area of the boards and maximum possible stack height is 551.

4. CONCLUSIONS

The proposed procedure for reducing the number of saw cycles assumed that the cutting problem has only a single type of board from where the panels are to be cut. Algorithms for multiple type board problems have yet to be studied.

In practice there are still other cutting restrictions that were not explicitly taken into consideration, for instance, there are cases where the users want to minimise the order spread, and/or the number of open stacks during the production run is limited. A modified procedure can be used in these cases. We just need to provide a regeneration mechanism for the value of the stack size so that when an order is finished and/or a stack is closed the stack size is allowed to increase again. This can be done without much effort.

The method might become slow for large data set problems, and in practice, a mechanism that limits the number of passes can be incorporated so that the *running* time is kept within desired values. As we stated before, in most of the tests performed we had obtained quite good results after 3 passes. Also, we should observe that if the target efficiency is higher it is more likely that the running time for the pass is increased. This is so because it is more likely that a pattern generated is not accepted at the early stages and hence, it is more likely that the total number of patterns generated until the algorithm finishes will be higher. As we know the running time of the algorithm is directly related to the number of patterns generated. Therefore, we are not being optimistic if we expect a reduction of about 20% per pass in running time if we decide to stop the algorithm before all 5 passes.

It was brought to our attention recently that a quite *similar* idea had been proposed *previously* (see Haessler, 1975) to reduce the number of patterns in a one dimensional trim problem, while still minimising the trim loss.

5. REFERENCES

YANASSE, Horacio H., ZINOBER, Alan S.I., HARRIS, Reginald G. Two dimensional cutting stock: board length determination. Instituto Nacional de Pesquisas Espaciais, SÆo José dos Campos, INPE-5383-PRE/1749, April 1992.

DYCKHOFF, H., ABEL, D., GAL, T., KRUSE, H.J. Trim loss and related problems. OMEGA, v.13, n.1, p.59-72, 1985.

HINXMAN, A.I. The trim loss and assortment problems a survey. EJOR, v.5, p.8-18, 1980.

YANASSE, Horacio H., ZINOBER, Alan S.I., HARRIS, Reginald G. Two-dimensional cutting stock with multiple board sizes. Journal of the *Operational Research* Society, v.42, n.8, p.673-683, 1991.

HAESSLER, R.W. Controlling cutting pattern changes in one- dimensional trim problems. Operations Research, v.23, n.3, p.483-493, 1975.