VICTORIA UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF COMPUTER AND MATHEMATICAL SCIENCES

Cutting Stock Problems with Contiguity and Other Objectives

Lutfar R. Khan and Robert H. Hinterding

(77 OR 4)

June, 1996

(AMS : 90C27, 68T20, 90-04, 90-08)

TECHNICAL REPORT

VICTORIA UNIVERSITY OF TECHNOLOGY
P O BOX 14428
MELBOURNE CITY MAIL CENTRE
MELBOURNE, VICTORIA, 8001
AUSTRALIA

TELEPHONE (03) 9688 4492
FACSIMILE (03) 9688 4050

Footscray Campus

Cutting Stock Problems with Contiguity and Other Objectives

Lutfar R. Khan & Robert H. Hinterding
Victoria University of Technology, Victoria, Australia

Presented at the ***SICUP Workshop on Cutting and Packing***
within **INFORMS XXXIII International Conference**
Singapore, June 1995

Cutting Stock Problems with Contiguity and Other Objectives

Lutfar R. Khan and Robert H. Hinterding

Department of Computer and Mathematical Sciences
Victoria University of Technology
PO Box 14428 MMC, Melbourne 3000
Australia

ABSTRACT. ***In cutting stock problems, it is generally preferred that once a particular item size is produced, it should be present in all subsequent production runs until the demand is met. Ideally one can produce a perfectly contiguous schedule, but it will lead to unacceptable trim-loss. It is difficult to handle the requirements of contiguity and other secondary objectives in mathematical programming frameworks. We show that genetic algorithms can successfully be used for these sort of problems and we report encouraging results.***

1. Introduction

The cutting stock problem (CSP) is defined as:

$$\text{Minimise} \quad = \sum_{j \in J} w_j x_j \qquad (1)$$

$$\text{subject to} \sum_{j \in J} a_{ij} x_j = N_i \quad \text{for} \quad i = 1,2,..,n \qquad (2)$$

$$x_j \geq 0, \text{integer for} j \in J \qquad (3)$$

where, n = number of orders.
w_j = trim per run of pattern j.
a_{ij} = number of pieces of item i in pattern j.
x_j = number of runs of pattern j.
N_i = number of pieces of item i.

If there is only one stock length L, and l_i is the length of order i, then

$$L = \sum_{i=1}^{n} a_{ij} l_i + w_j, \text{ for all } j \in J$$

Then CSP can be written as:

$$\text{Min } X = \sum_{j \in J} x_j \qquad (4)$$

subject to (2) and (3).

The cutting stock problem is one of the first decision problems of Operations Research modelled in a mathematical programming framework. This model and its variants have been widely used in the paper industry; paper is produced in standard lengths and then cut

into appropriate sizes to meet customers' demands. Other application areas include steel mills and cable industries. When the items to be produced vary in one dimension (length), the problem is called 1-dimensional CSP. Item sizes specified in two or more dimensions have similarly been modelled. If there is more than one stock length from which the orders are cut, the problem is called multiple stock length CSP. A general description and classification of cutting problems is given in Dyckhoff (1990).

The pioneering work in solving a CSP was by Gilmore and Gomory (1963). They used a linear programming model; the integrality constraints were relaxed initially to $x_j \geq 0$, and the LP solution was obtained by a clever method of pattern (column) generation. To obtain an integer solution from the LP solution, usually simple rounding is used, although other techniques have also been suggested (e.g., Johnston, 1986). Recent work on the LP approaches indicate that most instances of CSP have the integer round-up property (Marcotte, 1985), that is, an integer solution is available for the rounded objective function value (X) of continuous LP. Although the continuous LP solution is often not implementable before being converted to an integer solution, the lower bound it provides for the integer solution facilitates the finding of an optimal integer solution. Recent results (Gau, 1995, private communication) provide optimal integer solutions. These were obtained through a decomposition heuristic (Wascher and Gau, 1993), and for numerous test problems from the literature all instances with rare exceptions have the integer round-up property. It may be claimed that the solution procedure for a standard one-dimensional cutting stock problem of *minimising trim loss* has already been made efficient in a linear programming framework.

When *additional constraints* like a limit on the maximum number of items in a pattern and/or a limit on the maximum number of distinct patterns (Haessler, 1971, 1975), range of demand instead of an exact demand and associated question of absolute and percentage trims (Johnston, 1986, Goulimis, 1990), localisation of the total trim in a few patterns (Roodman, 1986), maximisation of trim in the last pattern for subsequent reuse (Sinuany-Stern and Weiner, 1994), are imposed on the CSP, the standard column generation procedure and consequent optimisation for integer solution may require substantial modifications. Approaches suggested in the literature include: repeated use of linear programming at various phases of the algorithm, branch and bound and cutting plane methods of integer programming, and heuristic generation and combination of patterns.

Once the patterns have been identified, the question of sequencing them for actual production is important not only to keep the required number of pattern changes low, but also to provide a relatively *contiguous* production plan so that the number of part-finished sizes is relatively small to ensure quality consistency and ease of storage. A sequence of patterns is perfectly contiguous if a particular size of item once started to be produced is in all successive runs until the full demand for that size is met. However, perfect contiguity will seldom be the ultimate goal because it may raise the amount of wastage to an unacceptable level. The sequencing of patterns is generally done as a second phase of the problem -- the set of patterns is determined first and then sequenced in the best possible way. Usually this results in a less contiguous solution.

Exact and heuristic algorithms based on mathematical programming have been successful in handling many instances and variants of cutting stock problems as evidenced in the above named references. One difficulty with these algorithms is the need to precisely define the

constraints and the objective function, which may contain conflicting goals, and the consequent increase in the complexity of solving them. Recently a number of innovative heuristic search techniques such as Genetic Algorithms, Simulated Annealing and Tabu Search have entered the area of combinatorial optimisation problems. Examples include Travelling Salesperson Problem (Goldberg, 1985) and Bin Packing Problem (Reeves, 1993; Falkenauer, 1994). For cutting stock problems, some relevant references using non-traditional techniques are: Dagli (1990); Lirov (1992); Dincbas et al (1992); Lutfiyya et al (1992); and Hinterding and Khan (1994).

In this paper we deal with the solutions of cutting stock problems using Genetic Algorithms (GA). In particular, the advantages of using GAs will be highlighted. In Section 2, a general introduction to GAs will be given and its application for solving CSP will be outlined. Experimental results for variants of the standard CSP will be discussed in Section 3 and Section 4 will summarise the findings and suggest possible avenues for improvements.

2. Genetic Algorithm for CSP

2.1. Introduction

Genetic algorithms (GAs) describe a relatively new class of optimisation methods, loosely based on Darwinian evolution. Suitable for finding optimal or near-optimal solutions from solutions scattered over a large search space, GAs sample areas of the search space by techniques based on natural selection, inheritance and adaptation. GAs are part of the area of Evolutionary Computation, and are typified by their use of encoded solutions and the importance they place on crossover.

The earliest GAs were developed in the sixties and seventies (Holland, 1975), in order to simulate genetic processes. Early applications were concerned solely with genetic functions, and it was not until later that the power of applying methods based on genetics to other problems was realised.

The term genetic algorithms describes the class of optimisation methods whereby a population of encoded solutions (chromosomes) evolves with selection processes favouring the 'fittest' solutions. The solutions to problems are represented or encoded in finite length strings over some finite alphabet. Classical genetic algorithms use a binary representation and fixed length strings. However many applications and hybrid genetic algorithms use larger alphabets and variable length strings. These strings or chromosomes are evaluated according to fitness. Fitness is the cost of a solution subtracted from some ideal value (generally 1).

An initial population is built by generating random permutations of a legal encoding. Populations reproduce, producing new generations. Reproduction is managed such that the fitter a solution, the more likely the chromosome representing it will reproduce. Reproduction takes place by the reproduction operators of *mutation* and *crossover*. Crossover involves creating an individual by combining features from both parents. Mutation involves randomly altering one or more genes. Crossover ensures (hopefully) that features of the parents will be inherited by the children, mutation ensures that all

combinations can exist. Ideally, the generations would continue to improve and the last generation should contain an optimal solution to the optimisation problem.

The general steps of a GA are:

Initialise a population of random chromosomes
Repeat until satisfied
    Evaluate each chromosome in terms of the individual it represents.
    Create a new generation with chromosomes produced by mating individuals from the preceding population. Some individuals from the preceding population may be copied into the new generation.

Figure 1. Steps in the Genetic Algorithm.

GAs are blind. Solutions are not generated with access to local knowledge, but as a result of "payoff" information from their own evaluation. The genetic search is guided by probabilistic transition rules. The decision of whether a solution should reproduce is a function of its fitness and pure chance.

2.2. Representation of the Problem

A Group based GA was used to solve the cutting stock problems. The Group based GA uses a direct representation. It uses a mapping which focuses on the groups in a solution. That is, it tries to find the best selection of the possible groups. A group is a selection of items which will be cut from a single stock length. Falkenauer (1991) developed the Group based GA for the bin packing and other grouping problems. Here the emphasis is changed from the traditional GA where genes represent a single value and its position or order in relation to other genes is significant, to the situation where genes represent a group of items, and neither the order nor the position of the genes in the chromosome or the items in the gene is significant.

In the Group based GA, each chromosome represents a number of groups of items such that all items to be cut are represented. Each gene represents a *group* of items, rather than a single item. Each group will be cut from a single stock length. This mapping is illustrated in Figure 2. The characteristics of this representation are that the number of genes is variable, the order of the genes in the chromosome has no significance, and the order of the items in each gene has no significance. These characteristics are compatible with the characteristics of the bin packing and cutting stock problems.

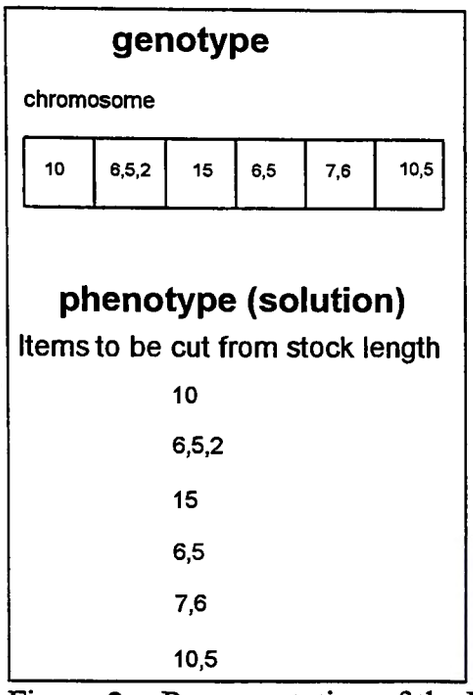

Figure 2. Representation of the Problem

2.2.1. The encoder

An intelligent encoder is used to build the initial population, and to build new groups for crossover and mutation operators. The program has been developed to solve cutting stock problems with multiple stock lengths. This encoder takes a list of stock lengths and a list of items in random order. Using a first fit algorithm it groups the items in the item list into groups to form genes. To build a group the encoder choses a stock length at random and then choses items without replacement from the list of items using a first fit algorithm. This study focuses on single stock length problems only.

2.2.2. The reproduction operators

2.2.2.1. Crossover.

The crossover operator is a modification of Falkenauer's Grouping Crossover (BPCX) (Falkenauer & Delchambre, 1992). This crossover (called the Grouping Crossover GCX) will work with chromosomes of different lengths, and does not depend on any ordering of the genes. It was designed so that the child can inherit meaningful information from both its parents. In this case it is the selection of genes (groups) the parents have.

The grouping crossover works in the following way: We randomly choose an insertion point in parent1 and a segment in parent2. The child is constructed by first copying into it the genes from parent1 up to the insertion point. Then we copy the genes from the segment in parent2 into the child, and lastly we copy the genes from parent1 after the insertion point into the child. We cannot blindly copy the genes into the child chromosome as a chromosome with duplicated items will result. A gene is only added to the child chromosome if all its items can be successfully subtracted from a list of items not yet included in the chromosome. At the end of crossover the list of items not yet included in the chromosome may not be empty, in this case the encoder is used to generate new genes (groups) from these items and the resulting gene(s) are added to the child chromosome.

Therefore, if two identical parents are chosen for crossover, an identical child may not always result.

The Grouping crossover gives no significance to the order of the genes in the parents, so a new crossover was developed to give greater significance to the order of the genes for the CSP. This new crossover, the Uniform Grouping Crossover (UGCX) works as follows: we generate a template of randomly generated binary bits which has the same length as the first parent. We then copy for each position in parent1 the gene from that position into the child chromosome, and then the gene from that position from parent2 is also copied into the child if its corresponding value in the template is a 1 only. Again the copy of a gene into the child chromosome is only carried out if the items from that genes can be successfully subtracted from a list of items not yet included in the child chromosome.

2.2.2.2. Mutation.

The mutation operator is based on Falkenauer's group mutation operator (Falkenauer & Delchambre, 1992). A number of genes are chosen and deleted. The items from the deleted genes are then used by the encoder to build new groups. These new groups are then added to the chromosome. The purpose of mutation in this case is to bring new groups into the chromosome. The genes to be deleted are chosen from those which do not cut exactly from the stock length (ie those with some trim), and then randomly if there are insufficient of these. Note that we must delete at least two genes, as deleting only one gene and then rebuilding the chromosome would result in exactly the same chromosome. Our group mutation operator is different from Falkenauer's as he deleted the gene with the greatest trim and then some others chosen at random. He adds the newly built genes to the end of the chromosome, while we insert them into the chromosome at a randomly chosen site.

2.2.3. The Fitness Function

The basic evaluation function for each solution of the cutting stock problem is calculated as the result of the following cost function subtracted from a fitness ceiling of 1.0.

$$\text{cost} = \alpha.\left(1-\frac{L}{(L+\text{wastage})}\right)+\beta.\left(\frac{\text{open}}{\text{ordsize}}\right)+\chi.\left(1-\frac{L+\text{maxwaste}-\text{minwaste}}{2*L}\right)$$

where,

m = number of groups (pattern runs)
L = the stock length
wastage = $\sum_{i=1,m}(L-\text{sum of lengths of items in group}_i)$
open = number of unfinished items at the present run
ordsize = number of distinct orders
maxwaste = maximum of the trims (wastages) in the patterns
minwaste = minimum of the trims in the patterns
α, β and χ are weights chosen between 0 and 1 ($\alpha+\beta+\chi \leq$)

The fitness function contains three terms, each having values between 0 and 1. The first is to enforce the reduction of trim. The second term encourages the sequencing of patterns in such a manner that there are few open items. The third term is provided to generate patterns with either very high or very low trim. The concentration of trim leads to better solutions, as new patterns are generated by combining small and large trim patterns so that fewer stock lengths are needed.

The basic Genetic Algorithm used is a steady-state GA based on the description of OOGA in Davis (1991). It was developed using Smalltalk/V for Windows, for running on PCs. It is written in a flexible manner that permits changes in parameter during the search. The running time was not a priority, and it was observed that reasonable solutions were always found in a few minutes of running time. The following parameters for this GA can be set:

- Population Size - set the size of the population.

- Allow Duplicates - set a flag to allow or disallow duplicates to exist in the population. If duplicates are not allowed, any duplicates produced by reproduction are discarded while they still count as an evaluation. We determine whether two chromosomes are the same by comparing their genotypes.

- Number of Evaluations - set the number of evaluations for the run. We use evaluations rather than generations so that we can compare between runs where the population size and replacement rate are different.

- Replacement Rate - set the percentage of the population that will be replaced by reproduction in one generation. The rate can be set from 0 to 100%.

- Crossover Rate - set the percentage of the replacement population that will be replaced by crossover in one generation. The remainder of the replacement population will be produced by mutation. The rate can be set from 0 to 100%.

- Poisson Mutation - use a Poisson distributed random variable to determine how many genes to mutate in a chromosome.

- Poisson Mean - set the mean (λ) for the Poisson distributed random variable.

In the Genetic Algorithm used, a new chromosome is produced either by crossover or mutation but not both. The mutation rate for the GA is (100 – crossover rate). The mutation rate is the percentage of chromosomes of the current population that will undergo mutation. If Poisson Mutation is false, then mutation of one gene is generally carried out. If Poisson Mutation is true, then the number of genes to be mutated in a chromosome is determined by sampling a Poisson distributed random variable with mean λ.

3. Results

The results of this study will be illustrated using two small problems from Hinterding & Juliff (1993). The experience with some large data sets will then be discussed. The first version of the GA was presented by Hinterding & Juliff (1993), and it has been enhanced a number of times since then. Hinterding & Khan (1994) presented results on cutting stock problem using an earlier enhanced version.

This study is of exploratory nature. The basic form of the fitness function used in the runs of the GA is given in Section 2.2.3, although the relative weights α, β and χ shown were not always used. In some cases, different values of the parameters were used as it is not possible to specify a particular set of parameters which will be *the best* for all instances of a problem. However, the general range of parameter values that work for particular problem types can be estimated empirically. For this study, we found the following parameters values generally effective:

Population size: 50
Crossover rate: 60%
Mutation: Poisson , mean 3
Duplicates in population: No
Replacement rate: 50%
α = 0.5 to 0.9
β = 0.01
χ = 0.001

Problem 1: Stock length 14
20 items

Item Length	3	4	5	6	7	8	9	10
No. req.	5	2	1	2	4	2	1	3

A minimum trim solution, as found by the GA in first generation, is as follows:

Solution 1 (Problem 1)

run	pattern	trim	open items
1	3 + 4 + 7	0	3
1	3 + 10	1	4
1	3 + 3 + 7	1	4
1	3 + 10	1	3
1	4 + 10	0	1
1	6 + 8	0	3
1	5 + 9	0	3
1	6 + 8	0	1
1	7 + 7	0	0

Open item means an item has been started, but not finished in the current run.

The algorithm aims to minimise total absolute trim and also to minimise the number of open items. The weight on the first objective is high, so the initial solutions pay little attention to

other factors as is evidenced by the staggered patterns and a large number of open items. Continuing the search, the algorithm found alternative solutions. The best solution after 500 evaluations is as follows:

Solution 2 (Problem 1)

run	pattern	trim	open items
2	7+7	0	0
1	3+3+3+3	2	1
1	3+10	1	1
2	4+10	0	0
1	5+9	0	0
2	6+8	0	0

This solution has the same trim, but it has a fully contiguous sequence; there is no interruption in the production.

Problem 2: stock length 25
60 items

Item Length	5	6	7	8	9	10	11	12
No. req.	7	12	15	7	4	6	8	1

Solution 1 (Problem 2)

run	pattern	trim	open items
1	7+8+10	0	3
1	6+6+6+7	0	4
1	6+7+12	0	4
2	7+8+10	0	4
2	5+5+6+9	0	6
2	5+9+11	0	7
3	6+8+11	0	6
1	7+8+10	0	5
1	6+6+6+7	0	4
2	7+7+11	0	4
1	5+10	10	3
1	7+7+10	1	2
1	7+7+11	0	0

The above solution, obtained in under 200 evaluations, is trim-optimal but includes a poorly selected and sequenced set of patterns. Continuing the search for 1000 evaluations, the following solution was obtained:

Solution 2 (Problem 2)

run	pattern	trim	open items
4	6+6+6+7	0	1
5	7+7+11	0	2
1	11+12	2	2
1	11+11	3	1
2	8+8+8	1	2
1	5+5+7+8	0	1
2	5+9+9	2	1
3	5+10+10	0	0

This solution has one interruption in production (item size 7); apart from that the sequence is contiguous.

For relatively large problems, the performance of the GA was encouraging. A number of test problems were taken from literature (Wascher and Gau, 1993). While the GA was able to find the optimal trim solution to most of the problems, in some cases it took more than 20000 evaluations. The primary purpose of this study was not to discover an optimal (integer) trim solutions, and the effort was directed towards applying the GA to find *alternative solutions* to the same problem based on the secondary objectives. Some such situations will be described in the following.

Contiguous Solution. As stated earlier, the measure for contiguity for the GA is taken as "maximum number of open items" . It turned out to be an effective measure for small problems. For large problems with large stock lengths, the contribution of the first term of the fitness function, i.e., the objective of trim minimisation, was very high and the algorithm failed to identify highly contiguous solutions. Some computational results are given below, the problem names refer to Wascher and Gau (1993):

Problem	No. Distinct Orders	Stock length	Optimal Solution	Evaluations *	No. Stock	Max. Open
BAR861	19	315	34	750	34	16
				4,000	34	13
				20,000	34	5
GOU902	22	4300	59	10,000	60	13
				15,000	60	12
JOH862	7	4734	200	600	200	7
				2,500	200	5
				6,000	200	6
JOH864	17	1344	84	2,500	84	16
				25,000	84	4
				50,000	84	5
WAE841	5	1520	89	10,000	89	5
				15,000	89	4

* Each row represents a separate run of the algorithm.

Different problems were run for different numbers of evaluations, as when there was no improvement in the fitness function for a large number of evaluations, the algorithm was terminated.

For the five examples shown, optimal solutions in terms of total trim were obtained in four of them, and for the fifth it was within one stock length of optimum. In terms of contiguity, it can be seen that in general when the algorithm is allowed to run for longer periods, fewer items remain open. If the weight for contiguity β is increased, the algorithm fails to find the optimal solution. One advantage of using the GA is that a fully sequenced set of patterns is obtained as a solution.

On the whole, the performance of the GA in maintaining contiguity needs improvement as sometimes simple manual rescheduling of the results would give improved contiguity. As can be expected, higher number of evaluations does not guarantee better results.

Maximise Trim in Last Pattern. Sinuany-Stern & Weiner (1994) reported a problem instance where it is required that the trim in the last pattern is as large as possible, so that the leftover trim can be a reusable stock. For a stock length of 600 cm, and a demand for 150 pieces of 123 cm, 60 pieces of 103 cm, 60 of 30 cm and 60 of 17 cm, the optimal solution is to use 46 stocks having a total trim of 150 cm, which was easily found by the GA. The authors did not attempt to find this optimal solution because they later imposed an additional condition that each item should have an allowance of 0.5 cm for cutting operation by disc saw. Using increased sizes of items (i.e., 123.5 cm in place of 123 cm), the optimal solution turns out to be 47 stocks. Using an integer programming model Sinuany-Stern & Weiner did find a solution with 47 stocks and with a trim in the last pattern of 354 cm. Using our GA, several solutions were found that allowed for cutting by disc saw (adding 0.5 to item size) and a large trim in the last pattern. The best one found in 100 generations (around 2500 evaluations) is given below:

run	pattern	trim
11	3 * 17.5 + 3 * 30.5 + 2 * 103.5 + 2 * 123.5	20
2	2 * 30.5 + 4 * 103.5 + 1 * 123.5	15
3	2 * 17.5 + 7 * 30.5 + 1 * 103.5 + 2 * 123.5	20
27	1 * 103.5 + 4 * 123.5	25
3	6 * 17.5 + 4 * 123.5	10
1	3 * 17.5 + 2 * 30.5	486.5

Although this solution has a relatively large trim in the last pattern as desired, it is not the best possible solution. Redistributing item size 17.5 from the last pattern to the third pattern would leave even larger trim in the last pattern. If the GA were run sufficiently long, that solution (or even a better one) would most likely be found by it; of course there is no guarantee because of the stochastic nature of the search algorithm.

Roodman (1986) discussed cutting stock problems where the emphasis is on having few patterns with large cuts or trims. A problem instance (BAR861 from Washer and Gau, 1993) was solved with an emphasis on localising the trim in as few patterns as possible. The GA found a solution which is optimal in terms of total trim (34 stocks) and had trims in only two of the patterns. These examples illustrate the usefulness of GA in handling auxiliary objectives.

4. Summary and Conclusion.

The GA as presented in this study can be expected to give optimal solutions to most one-dimensional cutting stock problems, where the objective is to minimise the total trim. The number of evaluations required, and thus the running time, may be very high compared to standard LP based algorithms, particularly for large problem instances. The real strength of GA lies in not finding an optimal solution to standard CSP, but in its ability to deal with problems with secondary objectives.

From our experience, a comparison of LP based algorithms and GAs can be done as follows:

- For standard CSP, the LP based algorithms are efficient, but cannot easily incorporate secondary objectives. GAs do not appear to be as efficient as LP based algorithms for standard problems, but is more versatile in dealing with secondary objectives.

- LP based algorithms usually start with very poor or incomplete solutions and good solutions are only obtained at the end. A GA always starts with workable solutions and steadily improves on these.

- There is some performance guarantee in LP based algorithms (because of available bounds), but GAs have no such guarantee. They are governed by rules of probability and may at times perform poorly.

- We can produce sequenced solutions using a GA in a single stage; this is an advantage over standard LP techniques which usually require a second stage for sequencing.

The contiguity results in this study have not been as good as expected, although some solutions are quite good. Two areas need further investigation : firstly, the type of crossover used should be explored further so that the ordering of the groups (patterns) is given more significance and secondly the term in the fitness function reflecting the contiguity factor may have to be improved.

The objective of concentrating trim to a few patterns can be very efficiently handled by the GA. In fact, this requirement reinforces the search for better solutions in the population, and facilitates the finding of least-trim solutions.

Estimating the best parameter values for the GA to solve the different problem types is an area for further research. Apart from improving upon the contiguity aspect of the algorithm, some other objectives we intend to study include: minimising the number of distinct patterns, controlling the number of items in a pattern, and the location of particular item sizes in preferred positions in the sequence of patterns.

Acknowledgement: The authors thank T. Gau for providing test problems with optimal solutions.

References

Dagli, C., *Knowledge-based Systems for Cutting Stock Problems*, European Journal of Operational Research, Vol 44 (1990), pp 160-166.

Davis, L. (ed), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold 1991.

Dincbas, M., Simonis, H., and Van Hentenryk, P., *Solving a Cutting Stock-Problem with the Constraint Logic Programming Language CHIP*, Mathematical and Computer Modelling, Vol 16/1 (1992), pp 95-106.

Dyckhoff, H., *A Typology of Cutting and Packing Problems*, European Journal of Operational Research, Vol 44 (1990), pp 145-159.

Falkenauer, E., *A Genetic Algorithm for Grouping*, Proceedings of the Fifth International Symposium on Applied Stochastic Models and Data Analysis, Granada, Spain 1991.

Falkenauer, E.A, & Delchambre, A., *A Genetic Algorithm for Bin Packing and Line Balancing*, Proceedings of 1992 IEEE International Conference on Robotics and Automation (RA92), pp 1186-1193, Nice 1992.

Falkenauer, E., *Setting New Limits in Bin Packing with a Grouping GA Using Reduction*, Technical Report RO108, Department of Industrial Automation, Research Centre for the Belgium Metalworking Industry, Brussels Belgium, 1994.

Gau, T. *Private Communication on CSP Integer Solutions*, (1995).

Gilmore, P.C. and Gomory, R.E., *A Linear Programming Approach to the Cutting Stock Problem; Part II*, Operations Research, Vol 11 (1963), pp 863-888.

Goldberg, D.E. and Lingle, R. *Alleles, Loci, and the Travelling Salesman Problem.* Proceedings of an International Conference on Genetic Algorithms and their Applications pp 154-159, 1985.

Goulimis, C., *Optimal solutions for the cutting stock problem*, European Journal of Operational Research, Vol 44 (1990), pp 197-208.

Holland, J. H. , Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, 1975.

Hinterding, R.H. and Juliff, K., *A Genetic Algorithm for Stock Cutting: An exploration of Mapping Schemes*, Technical Report 24COMP3, Department of Computer and Mathematical Sciences, Victoria University of Technology, Victoria Australia, Feb. 1993.

Hinterding, R.H. and Khan, L.R., *Genetic Algorithms for Cutting Stock Problems: with and without Contiguity*, Proceedings of the Workshop on Evolutionary Computation in Association with 7th Australian Joint Conference on Artificial Intelligence, November 1994, pp 175 - 196.

Johnston, R.E., *Rounding Algorithms for Cutting Stock Problems*, Asia-Pacific Journal of Operational Research, Vol 3 (1986), pp 166-171.

Lutfiyya, H., McMillin, B., Poshyanonda, P. and Dagli, C., *Composite Stock Cutting Through Simulated Annealing*, Mathematical and Computer Modelling, Vol 16/1 (1992), pp 57-75.

Lirov, Y., *Knowledge-based Approaches to Cutting Stock Problem*, Mathematical and Computer Modelling, Vol 16/1 (1992), pp 107-125.

Marcotte, O., *The Cutting Stock Problem and Integer Rounding*, Mathematical Programming, Vol. 33 (1985), pp 82-92.

Reeves, C. *Hybrid Genetic Algorithms for Bin-packing and Related Problems*, submitted to: Annals of Operations Research, 1993.

Roodman, G.M. *Near-optimal Solutions to one-dimensional cutting stock problems*, Computers and Operations Research, Vol 13 (1986), pp 713-719.

Sinuany-Stern, Z. and Weiner, I., *The One Dimensional Cutting Stock Problem Using Two Objectives*, Journal of the Operational Research Society, Vol 45/2 (1994), pp 231-236.

Wascher, G. and Gau, T. , Test Data for the One-Dimensional Cutting Stock Problem,Technical Report Number 93/9, Technical University Braunschweig (1993).