VICTORIA UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF COMPUTER AND MATHEMATICAL SCIENCES

Genetic Algorithms for Cutting Stock Problems: with and without Contiguity

Robert Hinterding & Lutfar Khan

(40 COMP 12)

August, 1994

TECHNICAL REPORT

VICTORIA UNIVERSITY OF TECHNOLOGY
(P O BOX 14428) MELBOURNE MAIL CENTRE
MELBOURNE, VICTORIA, 3000
AUSTRALIA

TELEPHONE (03) 688 4249 / 4492
FACSIMILE (03) 688 4050

Footscray Campus

Genetic Algorithms for Cutting Stock Problems: with and without Contiguity

Robert Hinterding & Lutfar Khan

Abstract: A number of optimisation problems involve the optimal grouping of a finite set of items into a number of categories subject to one or more constraints. Such problems raise interesting issues in mapping solutions in genetic algorithms. These problems range from the knapsack problem to bin packing and cutting stock problems. This paper describes research involving cutting stock problems. Results show that the mapping that is used affects the solution in terms of both quality of the solution found and time taken to find solutions, and that different mappings are suitable for different variants of the problem.

1. Introduction

Genetic algorithms(GAs) have been applied to a number of ordering problems, ranging from the Travelling Salesperson Problem [Goldberg 85] to Job-shop Scheduling [Davis 85]. Typically these problems have been mapped by representing the solutions as ordered lists of alleles, each allele representing an item. Falkenauer [Falkenauer 91] has defined a sub-set of these problems as *grouping* problems. Such problems require the grouping of items into sets. An example is the classical Bin Packing Problem [Coffman et al 84] where a finite set of items is to be packed into a minimum number of bins of the same size.

In this paper we look at a variant of the bin-packing problem; the problem of packing a finite set of items into a number of categories of *different* sizes. Such problems are common in real-life in the areas of time-tabling, stock-cutting and pallet loading. The problem chosen is that of cutting a set number of rods from stock lengths of different sizes such that wastage is minimised, called the Cutting Stock Problem (CSP).

In representing solutions to such problems, the standard representation of the solution as a permutated list has a number of problems. Items to be cut may be represented as a queue in a permutated list, but the problem remains as to how to represent the different sized stock sizes (categories). Typically GAs have coped with this problem by using an intelligent decoder (builder) to assign items to categories. For example, Syswerda [Syswerda 91] uses an intelligent decoder or scheduler to assign items (jobs) from a queue (represented as a chromosome string) for a job-shop scheduling problem. Another approach is to use more than one chromosome per individual [Juliff 93][Hinterding & Juliff 93]. Different aspects of a solution are encoded onto separate chromosomes and a *decoder* builds solutions, taking specifications from the separate chromosomes.

Another problem is that when chromosomes are used to represent groups of items, crossover disrupts the groups and convergence to a solution can be very slow. Falkenauer [Falkenauer 92] uses a "grouping" chromosome to overcome this problem. Here each gene represents a group of items, and hence crossover works with groups of items rather than a list of single items.

In this paper we build on earlier work [Hinterding & Juliff 93] [Hinterding 94] to develop GAs for the CSP using two different mappings. Hinterding & Juliff implemented a multi-chromosome GA for cutting stock, and we dramatically improve and extend on their work. Hinterding [Hinterding 94] explored two different mappings for solving the Knapsack problem. We extend the use of these mappings to the cutting stock problem. Falkenauer has extended his bin packing GA [Falkenauer 94] by using local optimisation to improve his results dramatically and we compare our GAs to his.

We restrict ourselves to using GAs with only one chromosome per individual, and by not using local optimisation. We do this to see the effects of the different mapping on the GAs. Problems tested range from 20 to 126 items with one to six different stock sizes.

2. The Cutting Stock Problem

The cutting stock problem (CSP) is defined as:

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

$$\text{subject to } \sum_{j \in J} a_{ij} x_{ij} = N_i \qquad (2)$$

$$i = 1, 2, .., n$$

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

where, w_j = waste 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:

$$\text{Min } X = \sum_{j \in J} x_j$$

such that (2) and (3)

Note: $W = \sum_{j \in J} w_i x_j = X.L - \sum_{i=1}^{n} N_i l_i$

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 90].

The pioneering work in solving a CSP was by Gilmore and Gomory [Gilmore & Gomory 63]. They used a linear programming model to solve it; the integrality constraints were relaxed initially to $x_j \geq 0$, and the LP solution was obtained by a clever method of 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].

The LP approach was developed to incorporate factors such as multiple stock lengths, a limit on the number of items in a pattern and ranges of demands instead of a single demand. This approach proved generally very useful for the so-called "easy" problems where there are many orders of small sizes. As there are many alternative solutions, the non-integer solutions are often close to the integer solutions. However, when there are few orders of a relatively large size (the so-called "hard" problems) the LP approach has not been very successful.

The 1-dimensional CSP can be viewed as a Bin Packing Problem (BPP). Given a set of items and a set of bins of fixed capacity, the BPP is to assign the items to the bins in such a manner that each item is assigned to one and only one bin and the total number of bins used is a minimum. Treating the bin capacity as stock length and the item sizes as order lengths the two problems are equivalent. BPP is NP-Complete. A number of good heuristics exist for solving BPP [Coffman et al 1984]. Among the available exact algorithms for BPP, the branch and bound method of Martello & Toth [Martello & Toth 90], using the reduction procedure is quite efficient. A good source of reference for all cutting and packing problems is Sweeney & Paternoster [Sweeney & Paternoster 92].

Traditional methods of Operations Research such as integer programming and branch and bound have been used for the last few decades to obtain exact and approximate solutions to the cutting and packing problems. In the last few years, attention has been given to a number of innovative heuristic search techniques for these problems; these include Genetic Algorithms, Tabu Search and Simulated Annealing [Glover & Greenberg 89][Falkenauer & Delchambre 92][Hinterding & Juliff 93][Prosser 88][Reeves 93][Smith 85].

Contiguity. By solving a CSP by an LP model or otherwise, a set of patterns and the number of runs for each of these patterns are determined. To implement this solution at production floor level, the patterns have to be sequenced and scheduled. This sequencing factor is important for several reasons: the knife-setting changes required; the consistency in the quality of the products of the same customer order; and the storage of partly-finished and unready-for-packaging product stacks. Knife-setting difficulties can be ignored because automatic knife-changing devices are in use these days, but other problems of product quality and inventory are important. To alleviate these problems, it is desired that the production of items be, as far as practicable, " contiguous", ie., a particular size of item should be produced in successive patterns until that order size is completed. Similar requirements for other products like steel, glass, films, etc are quite possible.

Contiguous sequencing of patterns has not been well-researched. In practice, the sequencing is done in the 2nd step of a two-step process where in the 1st step, the patterns and their runs are determined. One example of 2-dimensional sequencing in this manner is [Yuen 91].

One approach to deal with contiguity requirements is to include a measure of contiguity in the objective function of the cutting stock problem. For instance, the maximum number of partly-finished orders at any instant of a production run can serve as such a measure -- the lower the number, the more contiguous the solution is. Adding a term "number of partly-finished open orders" into the objective function of a linear programming problem will make it non-linear and unwieldy for the existing LP methods for CSP. In this paper, the contiguity factor has been taken into account in some instances and solved by genetic algorithms.

3. Representing the Problem

Two different representations were used to solve the cutting stock problems using GAs. The Group based GA which uses a direct representation and an Order based GA which uses and indirect representation. The Group based GA 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. The Order based GA focuses on the order of the items such that the items can be grouped into a solution using a decoder.

3.1. Mapping using a Group based GA.

Falkenauer developed the Group based GA for the bin packing and other grouping problems [Falkenauer 92]. 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. This is a significant departure from the traditional GAs.

The problem in extending the Falkenauer's Bin Packing GA to the CSP where there are multiple stock lengths is how to encode from which stock length a group of items is to be cut from. Hinterding and Juliff used a multiple chromosome GA where a second chromosome was used to encode from which stock length the groups were to be cut. The second chromosome turns out to be unnecessary, as a valid group of items *implies* the stock length it should be cut from. This is the smallest stock length from which the group of items can successfully be cut.

Hence using this mapping we do not need to use a multi-chromosome GA for the Cutting Stock Problem.

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 1. 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.

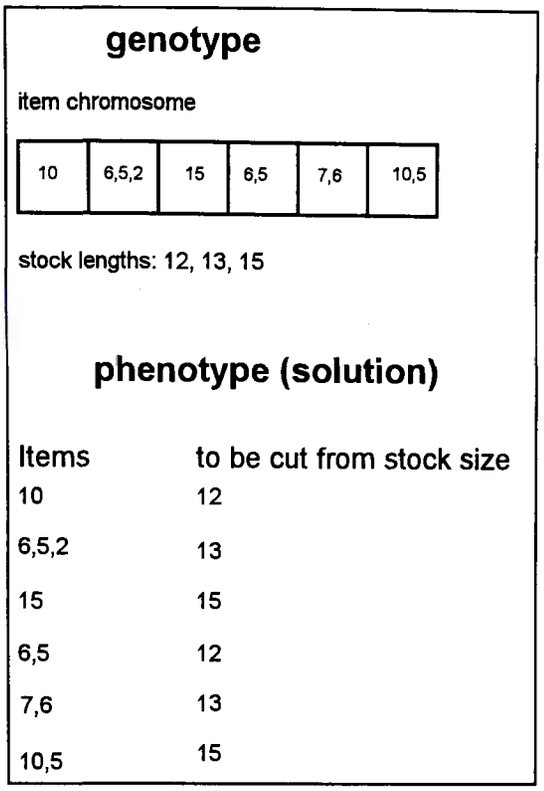

Figure 1 Symbolic representation of mapping

In the Cutting Stock Problem with contiguity, the order of the genes becomes significant. The mapping does not need to change, but different crossover and mutation operators are used to reflect the significance of the order of the genes.

3.1.1. The encoder

An intelligent encoder is used to build the initial population, and to build new groups for crossover and mutation operators. 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. The stock length is not recorded in the chromosome, but is implied by the group itself (see section 3.1).

3.1.2. The reproduction operators

Crossover.

The crossover operator is a modification of Falkenauers' Grouping crossover (BPCX) [Falkenauer & Delchambre 92]. 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 parent 1 and a segment in parent 2. The child is constructed by first copying into it the genes from parent 1 up to the insertion point. Then we copy the genes from the segment in parent 2 into the child, and lastly we copy the genes from parent 1 after the insertion point into the child. We cannot blindly copy the genes into the child chromosome as a chromosome with

duplicated items woll 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 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 with contiguity. 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.

Mutation.

The mutation operator is based on Faulkenauer's group mutation operator [Falkenauer & Delchambre 92]. 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 wastage), 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 it would result in exactly the same gene. Our group mutation operator is different from Falkenauer's as he deleted the gene with the greatest wastage 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.

For the cutting stock problem with contiguity we use our group mutation operator fifty percent of the time and we use the Remove and Reinsert mutation operator (RAR) for the other fifty percent of the time. The mutation operator is chosen randomly.

3.2. Mapping using an Order based GA.

With this mapping the chromosome represents an ordering of all the items to be cut. A decoder is needed to construct the groups from the ordering of the items, and the GA will (hopefully) explore orderings which build the best groups.

A problem with this mapping is that we cannot represent the stock length the group is to be cut from unless we use a second chromosome. As we want to explore the effect of different mappings, we have restricted ourselves to not using a multi-chromosome GA. For this reason we shall use only single stock length problems for the order based GA.

3.2.1. The decoder

The choice of decoder is very important. If we use the next fit algorithm to build the groups, then crossover is too disruptive of the groups. The next fit algorithm is to take the next item from the chromosome and put it into the current group if it will fit and otherwise start a new group using the item. The basis of this problem is that a minor reordering of the items can lead to a major change in the decoded groups [Hinterding 94][Falkenauer 91]. The use of a first fit algorithm was successfully used to overcome this problem with the knapsack problem [Hinterding 94] and it is used here as well.

3.2.2. The reproduction operators

Crossover We use the Uniform Order Based crossover (UOB) [Davis 91], as relative rather than absolute order should be more significant.

Mutation We use both swap and RAR mutation.

4. The Fitness Function

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

$$cost = \frac{\sum_{1,n} \sqrt{waste_i / stock\_length_i} + \frac{number\_wasted}{n}}{n}$$

where

n = number of groups
$stock\_length_i$ = the stock length that $group_i$ will be cut from
$waste_i$ = $stock\_length_i$ - sum of items in $group_i$
number_wasted = number of stock lengths with wastage

The fitness function contains two terms. The first is to reduce the wastage, we take the square root of this term to give values near the limits of the range (0-1) extra weight. This is done as we wish to concentrate the wastage. The second term encourages solutions where fewer stock lengths contain wastage, as this leads to better solutions.

4.1. Stock cutting with contiguity

The evaluation function for the stock cutting problem with contiguity is calculated as the result of the following cost function subtracted from the fitness ceiling of 1.0.

$$cost = \frac{\sum_{1,n} \sqrt{waste_i / stock\_length_i} + 10\left(\frac{no\_open\_items}{no\_different\_item\_lengths}\right)^2}{10+n}$$

where

n = number of groups
$stock\_length_i$ = the stock length that $group_i$ will be cut from
$waste_i$ = $stock\_length_i$ - sum of items in $group_i$
number_wasted = number of stock lengths with wastage

The second term in the fitness function is used to maximise the contiguity by minimising the number of open items. In fact we want to minimise the maximum number of open items. We square this term, to give values in the middle of the range (0-1) extra weight. The weight of 10 for the second term, was determined experimentally to give best results for Problem 4a.

As stated earlier, contiguity is difficult to quantify. By definition, a set of patterns is either contiguous or not. When an item, once started is included in all subsequent runs until finished, the patterns are said to be sequenced contiguously. Ideally, any breach of this condition will destroy contiguity.

In practice however, perfect contiguity will not be the ultimate goal because it may raise the amount of wastage to an unacceptable level. Therefore, the opposing objective functions of waste minimisation and contiguity have been combined in the fitness function. The number of open items indicates the number of item sizes started to be produced, but not yet finished at the end of the current run. Fewer open items means better contiguity.

5. The GA

The basic Genetic Algorithm used in both cases is a steady-state GA based on the description of OOGA in Davis [Davis 91]. Tournament selection is used with a tournament size of 2 as this was faster and gave comparable results to roulette wheel selection with linear normalisation. It was developed using Smalltalk/V for Windows. The following parameters 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. This was done so that the separate effects of these reproduction operators could be determined.

The mutation rate for the GA is 100 - Crossover rate. The mutation rate is the percentage of chromosomes of the replacement population that will undergo mutation. If Poisson Mutation is false, then mutation of one gene is generally carried out. The exceptions are swap mutation

where two genes are swapped, and grouping mutation where three genes will be deleted and rebuilt. 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.

6. Results

For this study we used mainly five different problems, the first three (Problems 1-3) are from [Hinterding & Juliff 93]. Problem 4 was derived from Problem 3 and modified to make it harder. Problem 5 was taken from a paper by Goulimis [Goulimis 90]. Two versions of each problem was created. One version has multiple stock lengths, and the other uses only a single stock length. The single stock length problems have an "a" suffix.

All the results were produced by averaging the results of 20 runs. Each run is divided into an equal number of intervals. The mean and standard deviation of the fitness values are calculated for each of the intervals over the batch of 20 runs. The parameters for each of the variants of the GA was optimised for best performance on Problem 4a, and run on all the other problems using the same settings. Typical solutions are shown in Appendix 2.

We have dramatically improved on the results for Problems 1-3 as reported by Hinterding & Juliff. They were unable to solve these problems completely within 20,000 evaluations, our results show these problems solved completely in well under 1,000 evaluations.

We note that for the cutting stock problem considering contiguity the Group based GA is clearly better than the Order based GA. In Figure 2, we graph the percentage crossover used against the maximum fitness reached for Problem 4a using the order based GA. This graph illustrates that crossover degrades the performance of the GA.

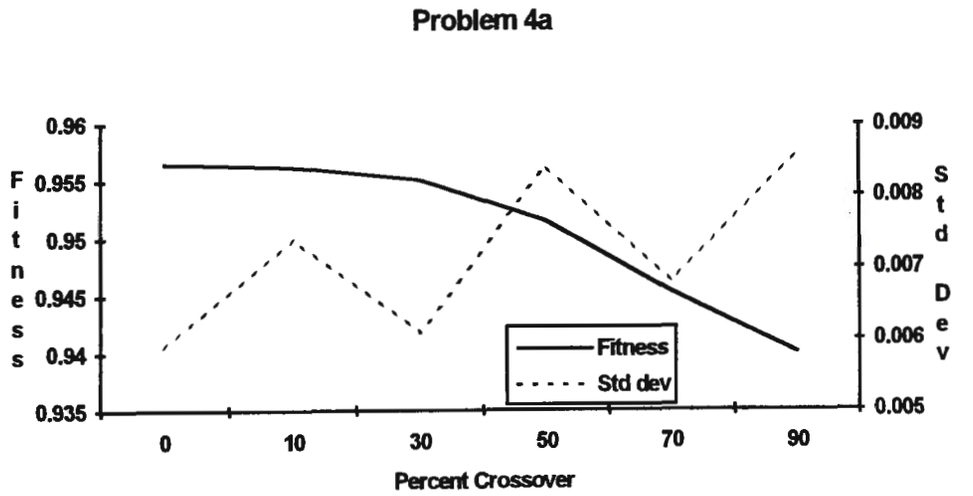

Figure 2: Order Based GA, percentage crossover vs average fitness

For the cutting stock problem with contiguity, both GAs give comparable results while the Group based GA is slightly better. To achieve good results, the number of evaluations had to be increased significantly.

Falkenauer [Falkenauer 94] improved his Bin Packing GA by using local optimisation inspired by the reduction method of [Mantello & Toth 90]. His paper gives details of generating *very* difficult bin packing problems. We ran the Group based GA on a number of these difficult problems. For each of these problems the optimum solution is known. On the 30 item problems we were able to solve it completely 75% of the time, and on 60 item problems 25% of the time. With 100 item problems were able to get within one stock length of the optimum within 5,000 evaluations. We conclude that his local optimisation method is extremely effective as Falkenauer was able to solve completely problems of up to 501 items. We also note that this local optimisation method cannot be used with the Order based GA, as the groups do not exist in the chromosome.

[Reeves 93] also used a reduction procedure for solving the bin packing problem by a GA. He reduced the problem size by deleting that subset of items which could fill one or more bins completely (or near completely). This type of reduction can be done with Order based mappings, and is easy to execute and can save substantial computation time. However, as Reeves acknowledges, it can lead to suboptimal solutions. In fact, the greater the number of items deleted by this reduction procedure, the greater the risk and level of suboptimality. Falkenauer's use of Martello & Toth's reduction does not lead to suboptimality.

Group based, no contiguity
Population: 75 Allow duplicates: false Replacement rate: 50%
Crossover rate: 30% Poisson mutation: ($\lambda = 4$)
Crossover: grouping
Mutation: grouping

Problem	evaluations	mean fitness	std dev	found at
1	1184	1.0	0.0	407
2	1184	1.0	0.0	740
3	1184	1.0	0.0	407
4	2294	0.9995	0.0022	2294
5	2294	0.9998	0.0007	2294

Problem	evaluations	mean fitness	std dev	found at
1a	1184	0.9133	0.0	296
2a	1184	0.9227	0.0018	1184
3a	1184	1.0	0.0	407
4a	1184	0.9642	0.0	851
5a	2294	0.8479	0.007	2294

Order based, no contiguity
Population: 100 Allow duplicates: false Replacement rate: 10%
Crossover rate: 0% Poisson mutation: ($\lambda = 4$)
Crossover: UOB
Mutation: RAR

Problem	evaluations	mean fitness	std dev	found at
1a	1200	0.9133	0.0	320
2a	3200	0.9198	0.0029	3200
3a	4200	1.0	0.0	2560
4a	5200	0.9588	0.0063	5200
5a	10200	0.8489	0.0048	10200

Group based, with contiguity
Population: 100 Allow duplicates: false Replacement rate: 70%
Crossover rate: 70% Poisson mutation: ($\lambda = 4$)
Crossover: UGCX
Mutation: 50% grouping, 50% RAR

Problem	evaluations	mean fitness	std dev	found at
1	2155	0.9860	0.0031	2155
2	5275	0.9796	0.013	5275
3	4235	0.9828	0.0041	4235
4	5275	0.9610	0.014	5275
5	10435	0.9889	0.0073	10435

Problem	evaluations	mean fitness	std dev	found at
1a	2155	0.956	0.0	1739
2a	5275	0.9212	0.0171	5275
3a	4235	0.9387	0.0318	4235
4a	5275	0.9335	0.0103	5275
5a	10600	0.8469	0.0118	10600

Order based, with contiguity
Population: 100 Allow duplicates: false Replacement rate: 70%
Crossover rate: 90% Poisson mutation: ($\lambda = 5$)
Crossover: UOB
Mutation: RAR

Problem	evaluations	mean fitness	std dev	found at
1a	2200	0.9554	0.0021	1990
2a	5700	0.9190	0.0109	5140
3a	4300	0.9096	0.0127	4300
4a	5700	0.9236	0.0145	5700
5a	10600	0.8454	0.0132	10600

7. Discussion

We can see from the results for the cutting stock problem without contiguity that the Group based GA is significantly better than the Order based GA. It is also evident that with the Order Based GA the use of crossover degrades the solutions found. Other order based crossovers have been tried but better results were obtained with the Uniform Order based crossover. We attribute this result to the fact that while there are orderings which will map to the best solution, this is a many to one mapping and there is no ordering information which the crossover operator can exploit. Any permutation of the groups is a solution is an equivalent solution, as is any permutation of the items in any group.

This is in contrast to the Knapsack Problem results in [Hinterding 94] where the Order based GA gave better results than the Selection based GA. Here the ordering performed was to move "better" genes towards the front of the chromosome.

In fact, the results of the Order Based GA on the CSP could be interpreted as indicating that reordering performed by the crossover operator interfered with the mutation operator. This suggests that there is no ordering information to process using this mapping.

For the cutting stock problem with contiguity the relative order of the groups becomes significant. The performance of the Order based and Group based GAs is roughly the same on this problem, with the Group based GA giving slightly better results. The Order based GA now has ordering information to process, as best results were obtained using 90% crossover. The crossover operator for the Group based based GA was modified so that ordering information could be processed. This modification was successful as best results were obtained with a crossover rate of 70%.

Comparing the results for the CSP with contiguity to the CSP without contiguity, we see they differ in three ways for a CSP with contiguity:

- groups with the same pattern are brought together.
- the number of distinct item lengths in groups are reduced.
- the number of distinct patterns are reduced.

All these features are desirable for the solution, but only the first can be achieved by the second operation of the traditional two-step process.

From the results for the CSP with contiguity, we see that there are some problems with the fitness function. There are solutions with the same fitness value, but differing contiguity (see Appendix 2). This occurs because we are optimising two conflicting measures , wastage and contiguity. If we did not worry about wastage, the contiguity problem is trivial. The solution is: cut the items in item length order ignoring wastage; perfect contiguity results. The conflict occurs as to minimise wastage, the GA attempts to concentrate wastage, while to maximise contiguity the wastage may have to be distributed. These conflicting aims can affect the quality of the solutions.

8. Conclusion

We have shown that the Cutting Stock Problem with and without contiguity can be successfully solved using Genetic Algorithms. It was relatively easy to modify the Group based and Order based GAs to include consideration of contiguity. Traditional OR methods do not handle CSP with contiguity well.

We have shown the results of different mappings and crossovers for GAs on these problems. The Group based GA is obviously better for the Cutting Stock Problem. For the Cutting Stock Problem with contiguity both the GAs work equally well. Only the Group based GA can be modified to include the local optimisation used by Falkenauer.

A pattern now starts to emerge as to the suitability of various mappings for different grouping problems. From [Hinterding 94] we see that an Order based GA is superior as the Knapsack Problem can be successfully mapped into an ordering problem. For CSP without contiguity the Group based GA is superior to the Order based GA, as it contains no ordering information for it to process. Lastly for CSP with contiguity both GAs are equally successful. The Order based GA as it has ordering information to process. The Group based GA was successful because the crossover was modified so that it could process ordering information.

References

[Coffman et al 84] Coffman, E.G., Garey, M.R. and Johnson, D.S., *Approximation Algorithms for Bin-Packing - an Updated Survey*, in G. Ausiello, M. Lucertini and P. Serafini (eds), *Algorithm Design for Computer System Design*, Springer Verlag, Vienna, pp.49-106.

[Davis 91] Davis, L.(ed), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold 1991.

[Dyckhoff 90] Dyckhoff, H., *A Typology of Cutting and Packing Problems*, European Journal of Operational Research, Vol 44 (1990), pp. 145-159.

[Falkenauer & Delchambre 92] Falkenauer, E.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 94] Falkenauer, Emanuel, *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.

[Gilmore & Gomory 63] 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.

[Glover & Greenberg 89] Glover, F. and Greenberg, H.J., *New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intelligence*, Vol 39 (1989), pp.119-130.

[Goldberg 85] 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.

[Goldberg 89] Goldberg, David E., *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley 1989.

[Goulimis 90] Goulimis, Constantine, *Optimal solutions for the cutting stock problem,* European Journal of Operational Research, Vol 44 (1990), pp. 197-208.

[Hinterding & Juliff 93] Hinterding, Robert and Juliff, Kate, *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 94] Hinterding, Robert, *Mapping, Order-independent Genes and the Knapsack Problem*, Proceedings of the First IEEE Conference on Evolutionary Computation (ICEC'94), pp. 13-17, Orlando 1994.

[Johnston 86] Johnston, R.E., *Rounding Algorithms for Cutting Stock Problems*, Asia-Pacific Journal of Operational Research, Vol 3 (1986), pp 166-171.

[Johnston 86] Johnston, R.E., *Rounding Algorithms for Cutting Stock Problems*, Asia-Pacific Journal of Operational Research, Vol 3 (1986), pp 166-171.

[Juliff 93] Juliff, K., *A multi-chromosome genetic algorithm for parellel loading,* Proceedings of the Fifth International Conference on Genetic Algorithms, pp 476-73, Urbana-Champaign 1993.

[Prosser 88] Prosser, Patrick. *A Hybrid Genetic Algorithm for Pallet Loading*, Proceedings of 8th European Conference on Artificial Intelligence, London 1988.

[Reeves 93] Reeves, Colin. *Hybrid Genetic Algorithms for Bin-packing and Related Problems*, submitted to: Annals of Operations Research 1993.

[Smith 85] Smith, D. *Bin Packing with adaptive search*, Proceedings of an International Conference on Genetic Algorithms pp.202-206, 1985.

[Syswerda 91] Syswerda, G., *Schedule Optimization Using Genetic Algorithms*, Handbook of Genetic Algorithms, Davis, L. (ed), 1991, Van Nostrand Reinhold.

[Sweeney & Paternoster 92] Sweeney, P.E. & Paternoster, E.R., *A Categorized, Application-Oriented Research Bibliography on Citting and Packing Problems*, Journal of the Operational Research Society, Vol 43 (1992), pp.691-706.

[Yuen 91] Yuen, B. J., *Heuristics for Sequencing Cutting Patterns*, European Journal of Operational Research, Vol 55 (1991), pp 183-190.

APPENDIX 1. The Problems

Problem 1: stock lengths 10, 13, 15
Problem 1a: 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

Problem 2: stock lengths 10, 13, 15
Problem 2a: stock length 15
50 items

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

Problem 3: stock lengths 10, 13, 15, 20, 22, 25
Problem 3a: stock length 25
60 items

Item Length	3	4	5	6	7	8	9	10
No. req.	6	12	6	5	15	6	4	6

Problem 4: stock lengths 13, 20, 25
Problem 4a: 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

Problem 5: stock lengths 4300, 4250, 4150, 3950, 3800, 3700, 3550, 3500
Problem 5a: stock length 4300
126 items

Item Length	2350	2250	2220	2100	2050	2000	1950	1900	1850
No. req.	2	4	4	15	6	11	6	15	13

Item Length	1700	1650	1350	1300	1250	1200	1150	1100	1050
No. req.	5	2	9	3	6	10	4	8	3

APPENDIX 2. Typical Solutions

Problem 1 without contiguity

Ord: 7 Ord: 8 Wastage: 0 | 15
Ord: 3 Ord: 10 Wastage: 0 | 13
Ord: 3 Ord: 10 Wastage: 0 | 13
Ord: 7 Ord: 8 Wastage: 0 | 15
Ord: 6 Ord: 7 Wastage: 0 | 13
Ord: 3 Ord: 10 Wastage: 0 | 13
Ord: 4 Ord: 5 Ord: 6 Wastage: 0 | 15
Ord: 4 Ord: 9 Wastage: 0 | 13
Ord: 3 Ord: 3 Ord: 7 Wastage: 0 | 13
Total Wastage: 0 No. waste items: 0 Stock used: 9

Problem 1 with contiguity

Ord: : 4 : 9 Wastage: 0 | 13 Open: 1
Ord: : 3 : 3 : 3 : 4 Wastage: 0 | 13 Open: 1
Ord: : 3 : 3 : 7 Wastage: 0 | 13 Open: 1
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 1
Ord: : 7 : 8 Wastage: 0 | 15 Open: 1
Ord: : 5 : 8 Wastage: 0 | 13 Open: 0
Ord: : 10 Wastage: 0 | 10 Open: 1
Ord: : 10 Wastage: 0 | 10 Open: 1
Ord: : 10 Wastage: 0 | 10 Open: 0
Total Wastage: 0 No. waste items: 0 Stock used: 10 No. different orders: 8

Problem 1a without contiguity

Ord: : 3 : 3 : 8 Wastage: 0 | 14 Open: 2
Ord: : 5 : 9 Wastage: 0 | 14 Open: 2
Ord: : 4 : 10 Wastage: 0 | 14 Open: 4
Ord: : 7 : 7 Wastage: 0 | 14 Open: 5
Ord: : 3 : 3 : 8 Wastage: 0 | 14 Open: 4
Ord: : 7 : 7 Wastage: 0 | 14 Open: 3
Ord: : 4 : 10 Wastage: 0 | 14 Open: 2
Ord: : 6 : 6 Wastage: 2 | 14 Open: 2
Ord: : 3 : 10 Wastage: 1 | 14 Open: 0
Total Wastage: 3 No. waste items: 2 Stock used: 9 No. different orders: 8

Problem 1a with contiguity

Ord: : 4 : 10 Wastage: 0 | 14 Open: 2
Ord: : 4 : 10 Wastage: 0 | 14 Open: 1
Ord: : 3 : 10 Wastage: 1 | 14 Open: 1
Ord: : 3 : 3 : 8 Wastage: 0 | 14 Open: 2
Ord: : 3 : 3 : 8 Wastage: 0 | 14 Open: 0
Ord: : 6 : 6 Wastage: 2 | 14 Open: 0
Ord: : 7 : 7 Wastage: 0 | 14 Open: 1
Ord: : 7 : 7 Wastage: 0 | 14 Open: 0
Ord: : 5 : 9 Wastage: 0 | 14 Open: 0
Total Wastage: 3 No. waste items: 2 Stock used: 9 No. different orders: 8

Problem 4 without contiguity

Ord: 6 Ord: 7 Ord: 7 Wastage: 0 | 20
Ord: 10 Ord: 10 Wastage: 0 | 20
Ord: 9 Ord: 11 Wastage: 0 | 20
Ord: 5 Ord: 9 Ord: 11 Wastage: 0 | 25
Ord: 8 Ord: 12 Wastage: 0 | 20
Ord: 9 Ord: 11 Wastage: 0 | 20
Ord: 6 Ord: 7 Wastage: 0 | 13
Ord: 5 Ord: 8 Wastage: 0 | 13
Ord: 5 Ord: 8 Wastage: 0 | 13
Ord: 6 Ord: 7 Wastage: 0 | 13
Ord: 6 Ord: 8 Ord: 11 Wastage: 0 | 25
Ord: 6 Ord: 7 Wastage: 0 | 13
Ord: 7 Ord: 7 Ord: 11 Wastage: 0 | 25
Ord: 6 Ord: 7 Wastage: 0 | 13
Ord: 5 Ord: 9 Ord: 11 Wastage: 0 | 25
Ord: 5 Ord: 8 Wastage: 0 | 13
Ord: 6 Ord: 7 Wastage: 0 | 13
Ord: 6 Ord: 7 Wastage: 0 | 13
Ord: 6 Ord: 8 Ord: 11 Wastage: 0 | 25
Ord: 5 Ord: 10 Ord: 10 Wastage: 0 | 25
Ord: 6 Ord: 7 Wastage: 0 | 13
Ord: 6 Ord: 7 Wastage: 0 | 13
Ord: 5 Ord: 8 Wastage: 0 | 13
Ord: 6 Ord: 7 Wastage: 0 | 13
Ord: 10 Ord: 10 Wastage: 0 | 20
Ord: 7 Ord: 7 Ord: 11 Wastage: 0 | 25
Total Wastage: 0 No. waste items: 0 Stock used: 26

Problem 4 with contiguity (Solution A)
Generation: 100 Best at: 79 Ave fit: 0.954 Max fit: 0.972 Min fit: 0.878

REACHED MAX GENERATION Best at: 79

Ord: : 5 : 8 Wastage: 0 | 13 Open: 2
Ord: : 5 : 8 Wastage: 0 | 13 Open: 2
Ord: : 5 : 8 Wastage: 0 | 13 Open: 2
Ord: : 5 : 8 Wastage: 0 | 13 Open: 2
Ord: : 5 : 8 Wastage: 0 | 13 Open: 2
Ord: : 5 : 8 Wastage: 0 | 13 Open: 2
Ord: : 5 : 8 Wastage: 0 | 13 Open: 0
Ord: : 10 : 10 Wastage: 0 | 20 Open: 1
Ord: : 10 : 10 Wastage: 0 | 20 Open: 1
Ord: : 10 : 10 Wastage: 0 | 20 Open: 0
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 : 7 Wastage: 0 | 20 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 1
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 1
Ord: : 11 : 12 Wastage: 2 | 25 Open: 1
Ord: : 11 : 11 Wastage: 3 | 25 Open: 1
Ord: : 9 : 11 Wastage: 0 | 20 Open: 2
Ord: : 9 : 11 Wastage: 0 | 20 Open: 2
Ord: : 9 : 11 Wastage: 0 | 20 Open: 2
Ord: : 9 : 11 Wastage: 0 | 20 Open: 0
Total Wastage: 5 No. waste items: 2 Stock used: 29 No. different orders: 8

Problem 4 with contiguity (Solution B)
Generation: 100 Best at: 68 Ave fit: 0.944 Max fit: 0.972 Min fit: 0.856

REACHED MAX GENERATION Best at: 68

Ord: : 5 : 7 : 8 Wastage: 0 | 20 Open: 3
Ord: : 5 : 7 : 8 Wastage: 0 | 20 Open: 3
Ord: : 5 : 7 : 8 Wastage: 0 | 20 Open: 3
Ord: : 5 : 6 : 6 : 8 Wastage: 0 | 25 Open: 4
Ord: : 5 : 6 : 6 : 8 Wastage: 0 | 25 Open: 4
Ord: : 5 : 6 : 6 : 8 Wastage: 0 | 25 Open: 4
Ord: : 5 : 6 : 6 : 8 Wastage: 0 | 25 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 : 12 Wastage: 0 | 25 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 2
Ord: : 6 : 7 Wastage: 0 | 13 Open: 1
Ord: : 10 : 10 Wastage: 0 | 20 Open: 2
Ord: : 10 : 10 Wastage: 0 | 20 Open: 2
Ord: : 10 : 10 Wastage: 0 | 20 Open: 1
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 2
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 2
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 2
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 1
Ord: : 9 : 11 Wastage: 0 | 20 Open: 2
Ord: : 9 : 11 Wastage: 0 | 20 Open: 2
Ord: : 9 : 11 Wastage: 0 | 20 Open: 2
Ord: : 9 : 11 Wastage: 0 | 20 Open: 0
Total Wastage: 0 No. waste items: 0 Stock used: 22 No. different orders: 8

Note: A comparison of the two solutions (A & B) for the same problem illustrates the effect of contiguity. While both solutions have the same fitness (0.972), A has a wastage of 5 and B has 0 - clearly B is better than A in terms of wastage. However, A has a maximum of 2 items open, compared to 4 of B; thus A is superior in terms of contiguity. This occurs due to the conflicting criteria of wastage and contiguity. It should also be noted that the different numbers of stock used (29 & 22) has little significance as differing stock lengths are used in the solutions.

Problem 4a without contiguity

Ord: : 5 : 9 : 11 Wastage: 0 | 25 Open: 3
Ord: : 7 : 8 : 10 Wastage: 0 | 25 Open: 6
Ord: : 5 : 10 : 10 Wastage: 0 | 25 Open: 6
Ord: : 5 : 6 : 7 : 7 Wastage: 0 | 25 Open: 7
Ord: : 6 : 8 : 11 Wastage: 0 | 25 Open: 7
Ord: : 5 : 9 : 11 Wastage: 0 | 25 Open: 7
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 7
Ord: : 6 : 9 : 10 Wastage: 0 | 25 Open: 7
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 7
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 7
Ord: : 6 : 9 : 10 Wastage: 0 | 25 Open: 6
Ord: : 6 : 7 : 12 Wastage: 0 | 25 Open: 6
Ord: : 6 : 6 : 6 : 7 Wastage: 0 | 25 Open: 6
Ord: : 6 : 8 : 11 Wastage: 0 | 25 Open: 6
Ord: : 5 : 6 : 7 : 7 Wastage: 0 | 25 Open: 6
Ord: : 7 : 8 : 10 Wastage: 0 | 25 Open: 5
Ord: : 6 : 8 : 11 Wastage: 0 | 25 Open: 4
Ord: : 5 : 5 : 7 : 8 Wastage: 0 | 25 Open: 2
Ord: : 6 : 8 Wastage: 11 | 25 Open: 0
Total Wastage: 11 No. waste items: 1 Stock used: 19 No. different orders: 8

Problem 4a with contiguity

Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 2
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 2
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 2
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 2
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 2
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 2
Ord: : 7 : 7 : 11 Wastage: 0 | 25 Open: 2
Ord: : 11 : 12 Wastage: 2 | 25 Open: 1
Ord: : 7 : 9 : 9 Wastage: 0 | 25 Open: 1
Ord: : 5 : 10 : 10 Wastage: 0 | 25 Open: 3
Ord: : 5 : 10 : 10 Wastage: 0 | 25 Open: 3
Ord: : 5 : 10 : 10 Wastage: 0 | 25 Open: 2
Ord: : 8 : 9 Wastage: 8 | 25 Open: 3
Ord: : 8 : 8 : 9 Wastage: 0 | 25 Open: 2
Ord: : 5 : 6 : 6 : 8 Wastage: 0 | 25 Open: 3
Ord: : 5 : 6 : 6 : 8 Wastage: 0 | 25 Open: 3
Ord: : 5 : 6 : 6 : 8 Wastage: 0 | 25 Open: 3
Ord: : 5 : 6 : 6 : 8 Wastage: 0 | 25 Open: 1
Ord: : 6 : 6 : 6 : 6 Wastage: 1 | 25 Open: 0
Total Wastage: 11 No. waste items: 3 Stock used: 19 No. different orders: 8