A genetic algorithm for stock cutting: an exploration of mapping schemes[1]

Robert Hinterding

Department of Computer and Mathematical Sciences

Kate Juliff

Department of Business Computing

Technical Report

24 COMP3 February 1993

(Draft 1)

Department of Computer and Mathematical Sciences

---

[1] A version of this report has been submitted to the International Conference of Genetic Algorithms 1993.

Abstract

A number of optimisation problems involve the optimal grouping of a finite set of items into a number of categories. Such problems raise interesting issues in mapping solutions to strings in genetic algorithms. We argue that single string representation of solutions as permuted lists for such problems is limiting. This paper describes research into mapping a stock cutting problem where a set number of rods must be cut from a number of stock lengths with minimum wastage. Alternative mapping strategies are described, and the results of a successful implementation using a two-chromosome genetic algorithm with one group-based chromosome and one permuted list chromosome are presented. We conclude that this problem could be successfully implemented in a single-chromosome grouping genetic algorithm, although more complex grouping problems would require multi-chromosomes.

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 and adaptation.

The earliest GAs were developed in the fifties and early sixties, 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. However many applications and hybrid genetic algorithms use a larger alphabet. These strings or chromosomes are evaluated according to fitness. Fitness is the cost of a solution subtracted from some ideal ceiling.

An initial population is built by generating random permutations of a legal encoding. Populations reproduce, producing generations. Reproduction is managed such that the fitter a solution, the more likely the chromosome representing it has of reproducing. 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

that features of the parents will be inherited by the children, mutation ensures that all combinations can exist.

Generations continue to be reproduced for a set number of generations or until the populations consist primarily of similar individuals. If there is little difference between members of a population, the population is said to have *converged*. Ideally, the fittest individual will be found in the last generation.

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. Encoding and decoding solutions

Figure 2 illustrates the encoding of a solution in a simple genetic algorithm to optimise the trivial function $f(x) = x^2$. The string or chromosome has ten genes or elements. Values or alleles are from a binary alphabet. The string represents a blueprint that defines an individual and is encoded as a binary integer. This string or blueprint decodes to a phenotype or solution. The solution represented by the string is 661^2.

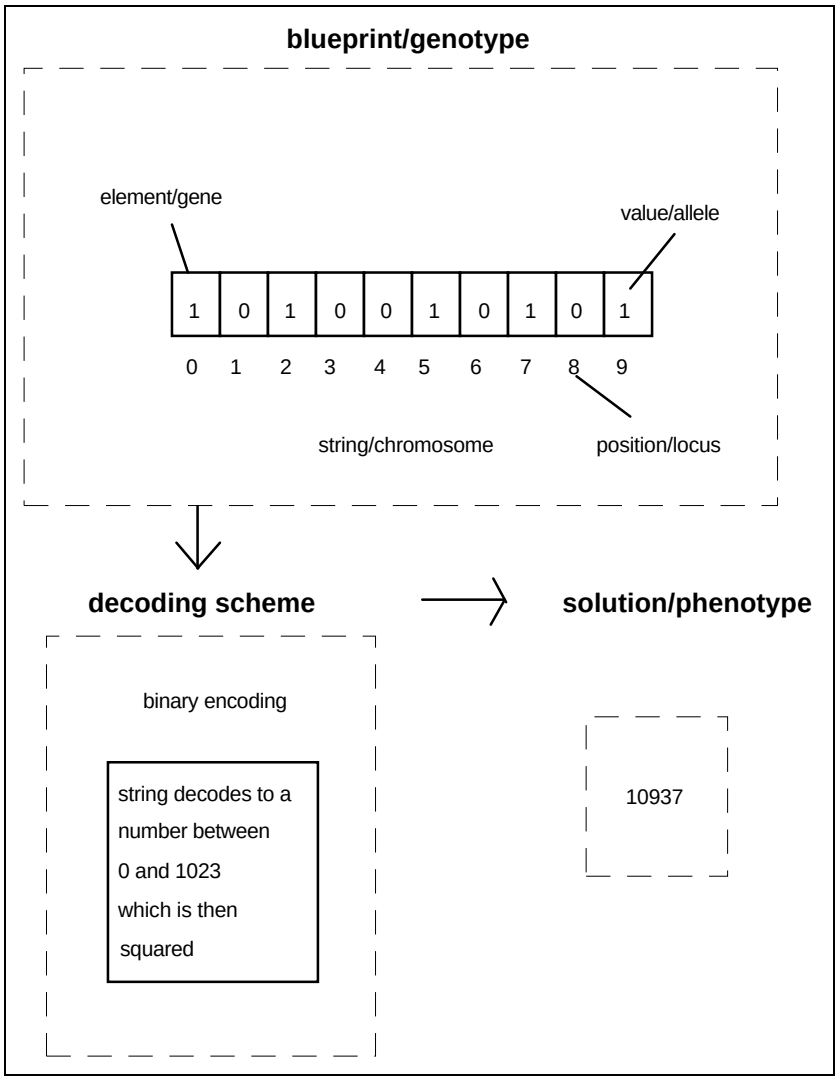

Figure 2 A ten element string decoding to a solution for the function $f(x)=x^2$

The encoding example above is trivial. More complex problems require more complex encoding schemes. For example, in encoding solutions to a stock cutting problem, items to be cut may be represented as a queue in a permuted list, but the problem remains as to how to represent the different sized stock sizes (categories) from which items are to be cut. 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.

More information can be represented by using more than one chromosome per individual [Juliff 92]. Different aspects of a solution are encoded into separate chromosomes and a *decoder* builds solutions, taking specifications from the separate chromosomes.

Falkenauer [Falken 92] uses "grouping" chromosomes to represent "groups" of items. An intelligent *encoder* encodes information into a single chromosome per

individual, such that a single chromosome represents a number of categories, each gene representing a group of items belonging to a single category. Abramson [Abram 91] uses a similar strategy for a time-tabling problem, where classes are grouped into periods, and crossover is carried out separately for each period.

Genetic Algorithms and grouping problems

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 [Falken 91] has defined a sub-set of these problems as *grouping* problems. Such problems require the grouping of items into a sets. An example is classical bin-packing [Garey 79] 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.

A number of mapping schemes were designed and implemented. The most successful was a combination of a multi-chromosome GA with a grouping chromosome representing items and a second chromosome representing stock sizes. Using this representation we developed a genetic algorithm that achieves good results. Problems tested range from 20 to 60 items with three to six different stock sizes. With this particular problem the need for the second chromosome could be eliminated by modifying the encoder so that it, rather than the GA determines stock size for groups.

We conclude that in order to implement GAs for complex grouping problems, mappings other than the traditional permuted lists and variants may be necessary.

3. The problem

The problem consists of cutting a set number of rods (items) from a number of different sized stock lengths such that wastage is minimised. The GA must find which items to cut from which stock lengths. It is therefore necessary for the decoder to know, not only the groups of items to be cut from individual stock lengths, but also the size of the stock length from which each group must be cut. A number of GAs have been implemented for similar problems. Falkenauer [Falken 92] implemented a GA for classical bin-packing where all bins are of the same size. Gemmil [Gemmil 92]

implemented a GA for stock-cutting where the problem is to find the optimal stock-lengths to be stocked. His GA samples the search space of stock-length only, as there is not a set of items to be actually cut.

4. The mappings

4.1. Single chromosome representation

Items to be cut could be represented as a permuted list as in the Travelling Salesperson Problem where each gene may represent a city, the position indicating the order in which the city is to be visited. Using this representation, the item in position 1 would be cut first, then the item in position 2 and so on. The problem with this representation is that the decoder must work out the stock size from which to cut the items, as stock sizes are not represented in the genotype.

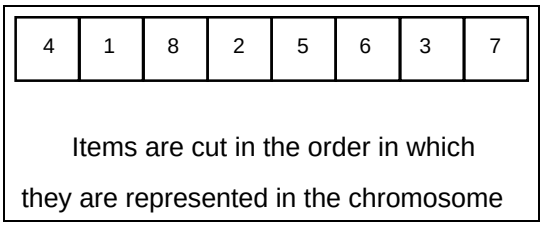

Figure 1 Single chromosome representation

4.2. Two chromosome representation

Because of the limitations of the single chromosome mapping, a number of multi-chromosome schemes were investigated. The item chromosome was kept, with each item represented by a gene, and its position indicating the order in which it was to be cut. A second chromosome was added to represent the groups or categories of stock length.

4.2.1. Stock sizes represented as a numeric chromosome

Given there are *n* stock sizes then a chromosome of length n may represent the number of lengths of each stock size that will be used for the items. For example a stock length chromosome containing 2,5,8 represents the information that two lengths of stock size #1 will be used, 5 lengths of stock size #2 and 8 lengths of stock-size #3. The decoder takes the first item from the item chromosome and cuts it from stock size #1. It continues taking items from the item chromosome and cuts then from stock size #1 until the two lengths are depleted and them starts cutting from stock size #2, and so on.

The problem with this mapping is that specialised reproduction operators are necessary for the stock size chromosome in order that there is adequate stock represented from which to cut the items.

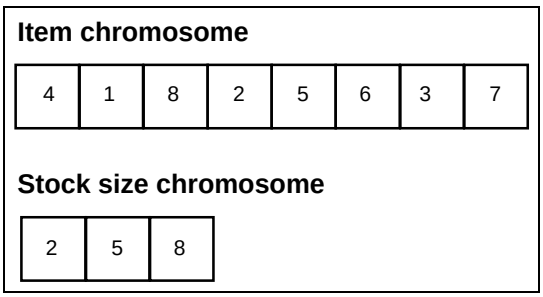

Figure 2 Two chromosome representation with stock size represented as a numeric chromosome

4.2.2. Stock sizes represented as a permuted list with duplicates

Stock sizes can be represented in a chromosome whereby each gene represents a particular stock size for an item. For example, a chromosome containing 4,6,6,2 may represent stock sizes #4, #6, #6 and #2. Each stock size is assigned to an item in the item chromosome by virtue of its position in the chromosome. The *j*th item in the item chromosome will be cut from the *j*th stock size in the stock size chromosome.

Alternatively items may be cut (in the order in which they appear in the item chromosome) from the stock lengths in the stock size chromosome such that the first item is cut from the first stock size represented in the stock size chromosome, if it can fit. If not it is cut from the next stock size and so on.

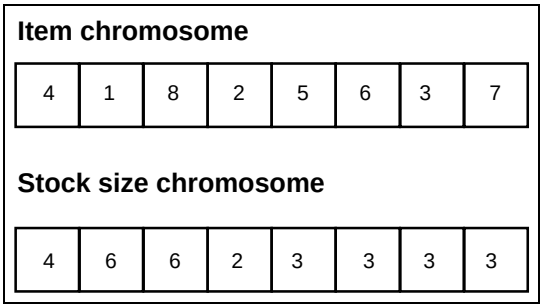

Figure 3 Two chromosome representation with stock size represented as a permuted list with duplicate

4.2.3. Items represented as groups

As the results for GAs using permuted lists to represent items were disappointing, it was decided to try another representation for the item chromosome. Falkenauer [Falken 92] reports problems in using traditional permuted lists for

grouping problems, as traditional reproduction operators have the negative effect of disrupting groups. The item chromosome as implemented in the GAs above, is effectively grouping items together to be cut from different stock sizes.

Falkenauer proposes a grouping representation, whereby the reproduction operators are acting on the *groups* rather than objects themselves. The mapping for items described below is based on Falkenauer's mapping for bin backing. The stock size chromosome is a permuted list with duplicates as described in section 3.2.2.

The item 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, such that the stock length for the group represented by the *ith* gene in chromosome 2 will be of the size represented by the gene in the *ith* position in chromosome 1.

The mapping is illustrated in figure 4. Note that there may be unexpressed material in chromosome 2, as this chromosome is of length *number_of_items* as it is possible that there could be one item per group.

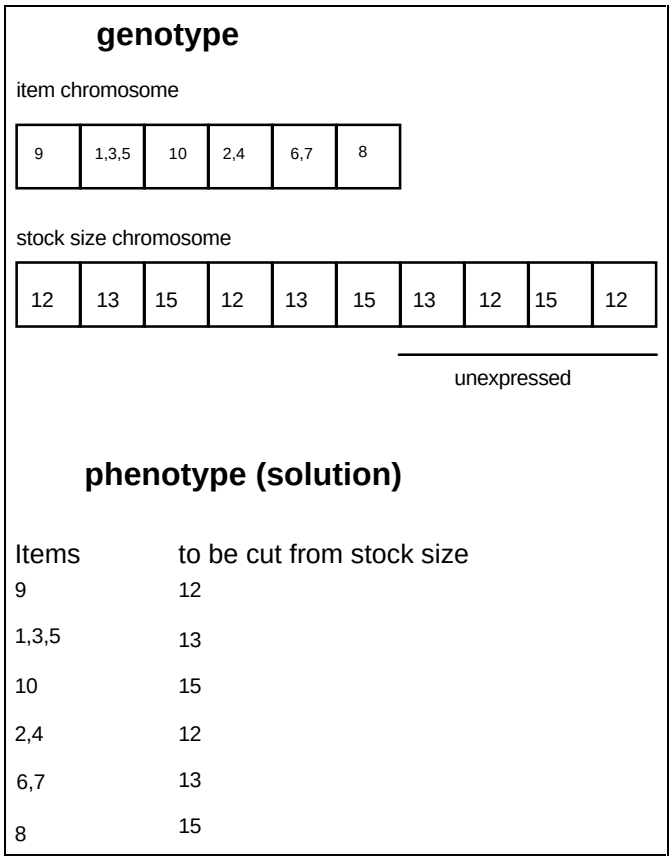

Figure 4 Symbolic representation of mapping

5. Implementation of the multi-chromosome GA with grouping

5.1. 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 randomly chosen stock lengths and a list of items in random order. Using a modified first-fit algorithm it groups the items in the item list into groups to be cut from the stock sizes, to form genes. The stock sizes are the genes for the stock size chromosome, the groups are the genes for the item chromosome.

It would also be possible to build a similar GA where the encoder alone establishes stock lengths, eliminating the need for the stock length chromosome. However, if there were constraints on the stock lengths and items (such as ordering constraints) then the second chromosome would still be necessary.

5.2. The reproduction operators

As the two chromosomes are of different types (permuted list and group) separate operators were implemented for each.

5.2.1. Item chromosome operators

5.2.2. Crossover.

The crossover operator is a modification of Falkenauer's Bin Packing Crossover (BPCX) [Falken 92]. Offspring are essentially copies of one parent, with a segment of the second parent inserted into the copy. Groups containing items that are duplicated in other groups are removed, and new groups are built up with items that are missing because of the removal of groups,. Essentially, offspring contain *groups* from both parents, plus new groups composed of *items* from both parents.

5.2.3. Mutation

Mutation is based on Falkenauer's group mutation [Falken 92]. A number of genes is chosen at random including the gene in the chromosome contributing least to fitness. Items in these genes are regrouped by the encoder.

5.3. Stock size operators

Crossover is standard two-point crossover. Mutation is adapted from standard mutation. When a gene is mutated its value is replaced with a randomly chosen value from the set of legal values (stock sizes).

The decoder takes each gene from the item chromosome, and selects the corresponding gene from the stock size chromosome. The items in the gene are cut from a stock size not less than the stock size represented by the gene in the stock size chromosome.

5.4. The evaluation function

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

$$cost = \frac{\sum_{1,n} \left(waste_i / size_i \right)^2 + \frac{number\_wasted}{n}}{n}$$

where

n = number of groups

$waste_i$ = $stock\_length_i$ - sum of items in $group_i$

number_wasted = number of stock-lengths with wastage

6. Results

The results for implementations of the three GAs with the items represented as permuted lists of *items* were disappointing. The GA search proved to be only marginally better than a random search. The results for the multi-chromosome grouping GA were very good. Figure 5 shows the results of this GA on three different problem sizes. Results are the average of 20 runs.

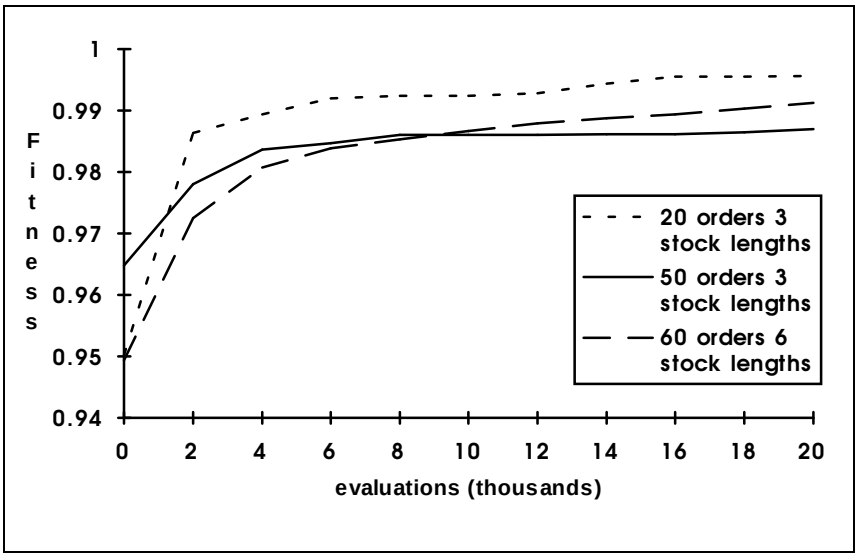

Figure 5 Results of the GA run on three different problems

7. Conclusion

By representing groups of items as single genes for a grouping problem, and ensuring that groups from both parents are maintained in off-spring, it is possible to search for good groups by a genetic search. This representation appears to be superior to a representation where groups span genes (traditional permuted list). It is also possible to represent different aspects of a solution by using more than one chromosome to represent an individual solution.

In order to use groups as genes in a chromosome, it is necessary to implement an intelligent encoder, to find the groups for the initial population and to maintain the groups during reproduction. Although this may result in complex code, the results of such an implementation bear the fruit of this work. Representing different aspects of a solution is somewhat simpler, as the mapping is relatively direct, and the decoder or builder merely builds a solution from the specifications as encoded in the chromosomes.

By using such variants of the traditional GA it is possible to tackle problems that are difficult using standard mappings.

References

[Abram 91a] Abramson, D. *A Parallel Genetic Algorithm for Solving the School Timetabling Problem* Information Technology Technical Report TR-DB-91-02 1991

[Davis 85] Davis, L. *Job-shop Scheduling with Genetic Algorithms,* Proceedings of an International Conference on Genetic Algorithms and their Applications, pp. 136-140, 1985

[Falken 91] Falkenauer, E. *A Genetic Algorithm for Grouping,* Proceedings of the Fifth International Symposium on Applied Stochastic Models and Data Analysis pp. 199-206, 1991

[Falken 92] Falkenauer, E. *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

[Garey 79] Garey Computers and Intractability: A guide to the Theory of NP Completeness, W. H. Freeman 1979

[Gemmil 92] Gemmill, D.D. *Solutions to the assortment problem via the genetic algorithm,* Mathematical Computer Modelling **16** 1 pp 89-94, 1992

[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

[Juliff 91] Juliff, K., *Using a Multi Chromosome Genetic Algorithm to Pack a Truck,* Technical Report (RMIT CS TR 92-2) Department of Computer Science Royal Melbourne Institute of Technology, 1992

[Smith 85] Smith, D. *Bin Packing with adaptive search,* Proceedings of an International Conference on Genetic Algorithms pp.202-206, 1986

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

Appendix

Summary of results for the three different problems.

The orders were randomly generated, and run twenty times.

Problem 1

Stock lengths: 10, 13 15

Number of Items to be cut: 20

item length	3	4	5	6	7	8	9	10
quantity	5	2	1	2	4	2	1	3

Wastage in the solutions

wastage (%)	0	0.8
no. found	10	10

Problem 2

Stock lengths: 10, 13 15

Number of Items to be cut: 50

item length	3	4	5	6	7	8	9	10
quantity	4	8	5	7	8	5	5	8

Wastage in the solutions

wastage (%)	1.2	1.4	2.0	2.3	2.6	2.9	3.2	3.5	4.1	4.6
no. found	1	1	2	3	2	2	2	6	1	1

Problem 3

Stock lengths: 10, 13 15, 20, 22, 25

Number of Items to be cut: 60

item length	3	4	5	6	7	8	9	10
quantity	7	12	6	5	15	6	4	6

Wastage in the solutions

wastage (%)	0	0.03	0.05	0.08	1.1	1.6	1.8	2.4	3.5	5.1	6.2
no. found	2	1	5	3	2	1	2	1	1	1	1