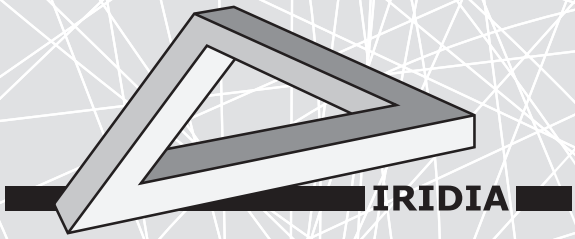


**Université Libre de Bruxelles**

*Institut de Recherches Interdisciplinaires*
*et de Développements en Intelligence Artificielle*

# A Genetic Algorithm for 2D Glass Cutting Problem

William Benfold, Max Manfrin, António Manuel Rodrigues Manso, Salvatore Spinella

**IRIDIA – Technical Report Series**

Technical Report No.
TR/IRIDIA/2005-13

September 2005

**IRIDIA – Technical Report Series**
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2005-13

The information provided is the sole responsibility of the authors and does not necessarily reflect the opinion of the members of IRIDIA. The authors take full responsability for any copyright breaches that may result from publication of this paper in the IRIDIA – Technical Report Series. IRIDIA is not responsible for any use that might be made of data appearing in this publication.

# A Genetic Algorithm for 2D Glass Cutting Problem

William Benfold wb02r@ecs.soton.ac.uk
University of Southampton, Southampton, Great Britain
Max Manfrin mmanfrin@ulb.ac.be
IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
António Manuel Rodrigues Manso manso@ipt.pt
Instituto Politécnico de Tomar, Tomar, Portugal
Salvatore Spinella spins@unical.it
Università della Calabria, Arcavacata di Rende, Italy

August 2003

**Abstract**

A Genetic algorithm is a heuristic approach which applies analogies from nature to tackle complex optimization problems. Existing solutions (the ancestors) are randomly modified to produce offsprings which might replace their parents.

We apply a genetic algorithm to a special case of a 2D bin packing problem, using a permutation-style representation decoded by a modified first-fit heuristic

## 1 Introduction

A glass maker usually starts from a set of desired pieces that have been commisioned to him, and a set of sheets to cut from. Loss is created when a sheet can not be exactly covered by the desired pieces. Loss is constituted by small glass pieces, called scraps, that can not be utilized in any way.

For economical reasons, glass makers require a given list of pieces to be placed over the smallest number of sheets, and the total glass loss to be minimized. There are also several technological constraints, deriving from how the glass sheet is actually cut by automatic or semi-automatic cutting machines (We don't treat these).

## 2 Problem Description

Given sheets with $H$ hight and $W$ witdh and given a set of pieces
$m_1$ with $h_1$ hight and $w_1$ witdh
$m_2$ with $h_2$ hight and $w_2$ witdh
... $m_n$ with $h_n$ hight and $w_n$ witdh

We wish to find a way to cut the pieces from the sheets which minimizes the area of wasted glass.However, there are additional constraints:

- All cuts must be either horizontal or vertical
- All cuts must be "guillotinable"; cuts must run across the whole length of the glass (2)

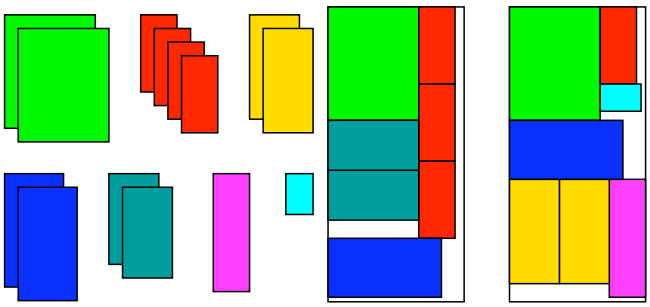

Figure 1: An example of a possible layout

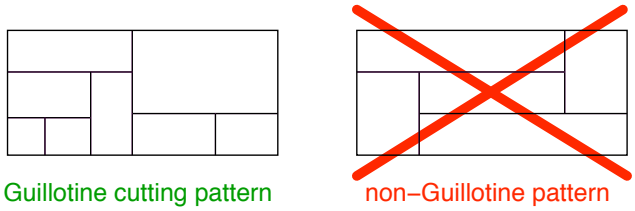


Figure 2: An example of a guillotinable layout

- There are also other industrial constraints which we shall not consider

A solution to this problem takes the form of a *cutting tree*. This is a tree for which each node represents one guillotinable cut, with each branch representing one of the two new subsheets created; an example is shown in 2). In industry, the glass is cut in stages, with each stage producing the cuts for one level of the tree. The depth of the tree is therefore limited to the number of stages used by the glassmaker. For the problems we shall be considering, there are three stages, so our cutting tree can have no more than three levels.

The 2D glass cutting problem is a particular case of a more general problem, namely the 2D Bin Packing Problem.

Lodi et al. [7] proposed the following topology for the four possible cases produced by *orientation* and/or *guillotine cutting* for 2D Bin Packing Problem:

**2BP—O—G** the items are oriented (O) and guillotine cutting (G) is required;

**2BP—R—G** the items may be rotated by 90 degree (R) and guillotine cutting is required;

**2BP—O—F** the items are oriented and cutting is free (F);

**2BP—R—F** the items may be rotated by 90 degree and cutting is free

The problem we shall consider is thus 2BP—R—G. The following references are examples of industrial applications involving the above variants. A problems of trim-loss minimization in a crepe-rubber mill, studied by Schneider [10], induces subproblems of 2BP—O—G; fuzzy two-dimensional cutting stock problems arising in the steel industry, discussed by Vasko et al. [11], are related to 2BP—R—G; the problem of optimally placing articles and advertisemens in newspaper and yellow pages, studied by Lagus et al. [12] falls into the 2BP—O—F case; finally, several applications of 2BP—R—F are considered by Benbtsson [13].

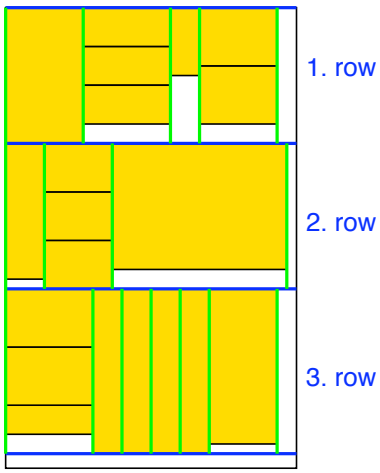


Figure 3: An example of a layout which can be produced using only two stages

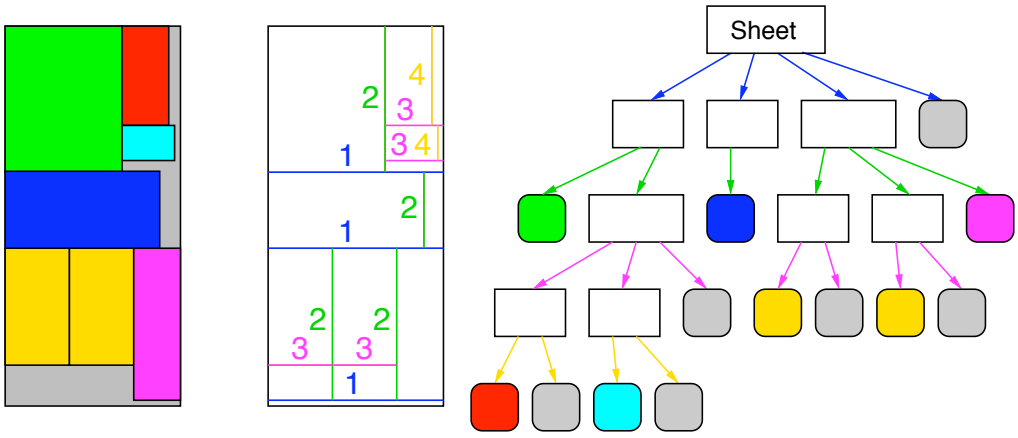


Figure 4: Diagram of a cutting tree

# 3 A Simple Heuristic

## 3.1 The First Fit Heuristic

A very simple way to construct a cutting tree is to use a "First Fit" heuristic. The rectangles are first sorted into height order, with the orientation being chosen so as to maximise height. The first rectangle is placed in the upper left corner of the first sheet. Subsequent rectangles are placed to the right of this until there is insufficient space to add another. The first row is now complete, with a height equal to that of the first rectangle. This process is repeated until no more rows can fit on the sheet, and is then repeated for the other sheets until no rectangles remain.

This heuristic is a "greedy algorithm", which considers only the heights of the rectangles. Within a row, the columns produced by the vertical cuts will each contain only one rectangle; no attempt is made to stack several rectangles together in a single column.

## 3.2 An Improved Heuristic

We present an improvement on the First Fit heuristic, whereby we attempt to fit multiple rectangles into the same column. Note that due to the restriction on the depth of the cutting tree, only rectangles of the same width can be placed in the same column.

When we have completed a row, we do not immediately proceed to the next. Instead, we iterate over all remaining unplaced rectangles, looking for any which have the same width as one of the columns, and are sufficiently small to be placed in the same column *without increasing the height of the row.*

# 4 The GA

## 4.1 The fitness function

The fitness function is derived from the criterion to be optimized and serves as a measure to evaluate individuals. Then, a successive improvement of the population's mean quality can be achived. However since genetic algoritms still are heuristic strategies, it cannot be guarateed that an optimal solution will be found.

The simplest bound for 2D Glass Cutting Problem is the *Continous Lower Bound*

$$L_{\emptyset} = \frac{\sum_{i=1}^{n} w_j h_j}{WH} \tag{1}$$

where

- $n$ is the number of the piece cutted
- $w_j h_j$ is the area of piece $j$
- $WH$ is the total area of the sheet

In our application a secondary objective is the maximization of th unused area in the last sheet, so as to produce a possibly large trim to be used later. The same secondary objective was used for solving 2BP—R—F in [13].

We combine the two objectives by defining our fitness function in terms of the number of sheets used and the fraction of the last sheet which is wasted. The function we use is:

$$\frac{N_{used} - L_{\emptyset}}{N_{used}}$$

where $N_{used}$ is the number of sheets completely used plus the fraction of the available height used on the last sheet. Hence our fitness function should be *minimized* to produce optimal results.

## 4.2 Representation

We use a permutation representation, but with the exception that we use duplicated labels to represent identical elements, rather than assigning every element a unique label. In the chromosome we also include an additional entry for the rotated version of each rectangle. The cutting tree is created by decoding this permutation using the Improved First Fit heuristic described above. Instead of sorting rectangles by height order, we sort them in order according to the chromosome. Thus the chromosome is a queue of rectangles to be placed by the heuristic. We track the number of rectangles of each type that we have placed, so that we do not place a rotation of an element which has already been placed.

## 4.3 Operators

### 4.3.1 Mutation operator

Similar, to nature, the task of mutation in a genetic algorithm is to effect random variations of the genes which are lost in the current population and which cannot be gained if only the existen material is combined. In general, mutation aims at preveservating the genetic diversity within a population of individuals.

We shall use two different mutation operators:

- **Swap**: exchange two randomly chosen elements
- **Move**: extract a random element and reinsert at a randomly chosen postion

### 4.3.2 crossover operator

To simulate the sexual propagation of individuals a recombination operator is included into genetic algorithms. its task is to combine the genetic material of offsprings which inherits certain good characteristics of its ancestors.

We shall explore two different crossover operators:

- **Uniform order based**, modified to accept duplicate labels
- **Elitist**:
  - Adopt genes appearing at the same positions in both parents
  - Choose the remainder from alternate parents where possible

Where we are unable to choose genes from alternate parents (because no genes of that type remain to be placed), we arbitrarily choose to insert the lowest-numbered gene which still remains to be placed.

## 4.4 Local Search

We define the neighbourhood of a permutation to be all other permutations which can be reached through a single swap. Because our chromosomes contain duplicated labels, we choose to ignore swaps which involve identical elements. The neighbourhood is still too large to explore exhaustively, so we a attempt a randomized first-improvement search. A swap is applied, and the fitness of the new permutation evaluated. If an improvement has been made, we terminate the search, and overwrite the original permutation with the improved one. Otherwise we undo that swap and apply a different one. If we try 30 swaps without any improvement being found, we terminate the search.

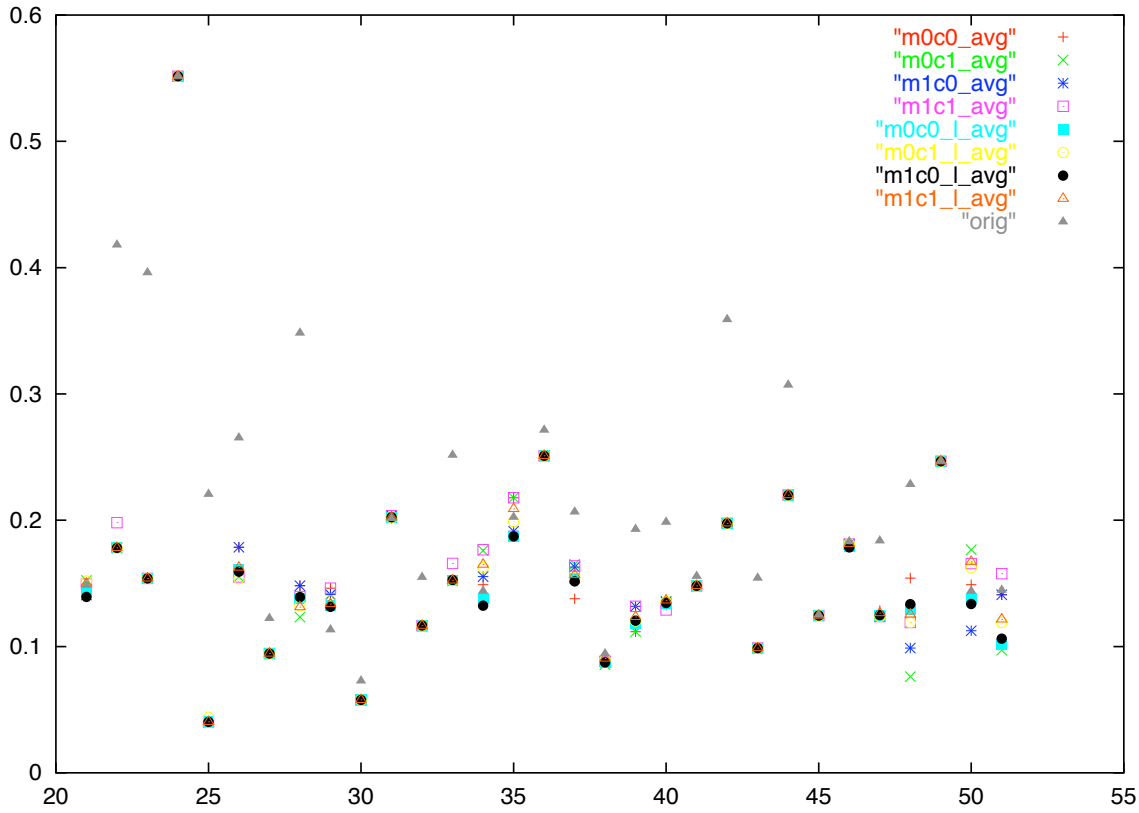


Figure 5: A plot showing average solution quality. The horizontal axis gives the problem number, the vertical axis gives fitness. Lower fitnes is better. The "orig" series shows the results obtained from the simple heuristic.

# 5 Experimental Setup

The experiments was carried out on with the follow features

| | |
|---|---|
| Processor | Intel(R) Pentium(R) 4 CPU 2.60GHz |
| Memory | 500 MB |
| Compiler environment | gcc version 3.3 |

# 6 Results

We observe that in many cases there is no significant difference in quality between the solutions produced by each variation of the algorithm. In a large number of cases, the solution quality is identical to that of the First Fit heuristic. Such cases appear to arise when the problem being considered is sufficiently simple that all algorithms, including the simple heuristic, are able to find the global optimum. For instance, there are a large number of problems containing rectangles which occupy more than half of a sheet; it is usually trivial to find a way to place the remaining rectangles without requiring any additional sheets. In addition, there are a few problems where the all of the rectangles fit on a single sheet, and there are so few rectangles that problem could easily be solved by enumerating all possible placements.

The problems where the algorithms differ in performance seem to be the most difficult instances. Such problems usually have a large number of different shapes of rectangle.

# 7 Conclusion

It would appear that both crossover and both mutation operators yield similar results, except on the hardest problems. Similarly, local search does not produce a consistent increase in the quality of solution. There appears to be, when all problems are considered, no particular advantage to be gained from using a particular operator, or from using local search. We do note, however that in almost all cases, the GA produces solutions of quality better than or equal to those of the simple heuristic.

**Acknowledgements**

Whe thank the organizers and teachers of the EvoNet Summer School 2003 for all the advices. In particular we thank Günther Raidl of the Vienna University of Technology, for providing us with the 2DBP framework and the GA library that we have used as starting point for this work. The computational experiments have been executed at the Laboratori CEDI- Facoltá di Ingegneria - Universitá di Parma

# References

[1] F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda, S. Bisotto *Optimizing Area Loss in Flat Glass Cutting.*

[2] E. Hopper and B. C. H. Turton, *An Empirical Investigation of Meta-Heuristic Algorithms for a 2D Packing Problem*, European Journal of Operation Research 128/1,34-57, 2000.

[3] E. Hopper and B. C. H. Turton, *A Review of the Application of Meta-Heuristic Algorithm to 2D Strip Packaging Problems*, Artificial Intelligence Review, vol. 16, 257-300, 2001.

[4] Shian-Miin Hwang, Cheng-Yan Kao, Jorng-Tzong Horng, *On Solving Rectangle Bin Packing Problems Using Genetic Algorihms*, IEEE 1994.

[5] Berthold Kroger, *Guillotineable bin packing: A genetic approach*, European Journal of Operational Research 84 (1995) 645-661.

[6] A. Lodi, S. Martello, D. Vigo, *Recent advances on two-dimensional bin packing problems*, Discrete Applied Mathematics 123 (2002) 379-396.

[7] A. Lodi, S. Martello, D. Vigo, *Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems*, INFORMS J. Comput. 11 (1999) 345-357.

[8] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA 1988.

[9] M.R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completness*, Freeman, San Francisco, CA, 1979.

[10] W. Schneider, *Trim-loss minimization in a crepe-rubber mill; optimal solution versus heuristic in the 2 (3)-dimensional case*, Eur. J. Oper. Res. 34 (1988) 273-281.

[11] F.J. Vasko, F. E. Wolf, K. L. Scott, *A pratical solution to a fuzzy two-dimensional cutting stock problem*, Fuzzy Sets and System 29 (1989) 259-275.

[12] K. Lagus, I Karanta, J. Ylä-Jääski, *Paginating the generalized newspaper: A comparison of simulated anniling and a heuristic method*, Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature, Berlin, 1996, pp 549-603.

[13] B. E. Bengtsson, *Packing recangular Pieces - a heuristic approach*, Comput. J. 25 (1982) 353-357