A population heuristic for constrained two-dimensional non-guillotine cutting

J.E. Beasley

May 2000

j.beasley@ic.ac.uk

http://mscmga.ms.ic.ac.uk/jeb/jeb.html

The Management School

Imperial College

London SW7 2AZ

England

ABSTRACT

In this paper we present a heuristic algorithm for the constrained two-dimensional non-guillotine cutting problem. This is the problem of cutting a number of rectangular pieces from a single large rectangle so as to maximise the value of the pieces cut. In addition the number of pieces of each type that are cut must lie within prescribed limits. Our heuristic algorithm is a population heuristic, where a population of solutions to the problem are progressively evolved. This heuristic is based on a new, nonlinear, formulation of the problem. Computational results are presented for a number of test problems taken from the literature.

Keywords: constrained two-dimensional non-guillotine cutting, population heuristic

1. INTRODUCTION

The constrained two-dimensional non-guillotine cutting problem is the problem of cutting from a single large planar stock rectangle (L_0,W_0), of length L_0 and width W_0, smaller pieces (L_i,W_i) each of length L_i and width W_i $(i=1,...,m)$. Each piece i is of fixed orientation (i.e. cannot be rotated); must be cut with its edges parallel to the edges of the stock rectangle (i.e. orthogonal cuts); and the number of pieces of each type i that are cut must lie between P_i and Q_i $(0 \leq P_i \leq Q_i)$. Each piece i has an associated value v_i and the objective is to maximise the total value of the pieces cut. Without significant loss of generality it is usual to assume that all dimensions (L_i,W_i) $i=0,1,...,m$ are integers.

Figure 1 shows an example cutting pattern. That pattern is a non-guillotine cutting pattern as the smaller pieces cannot be cut from the stock rectangle using guillotine cuts (cuts that run from one edge of a rectangle to the opposite edge, parallel to the other two edges). Figure 2, by contrast, illustrates a pattern that can be cut using a succession of guillotine cuts, i.e. it is a guillotine cutting pattern.

A special case of the two-dimensional non-guillotine cutting problem is the pallet loading problem. This occurs when all pieces are of the same size and the name comes from the practical problem of maximising the number of identically sized boxes that can be placed on a wooden pallet. This special case corresponds to $m=1$ $(v_1=1)$ when box rotation is not allowed, or to $m=2$ but with $L_1=W_2$ and $W_1=L_2$ $(v_1=v_2=1)$ when box rotation is allowed. It is usual in this case to regard the problem as being unconstrained; $P_i=0$, $Q_i \geq (L_0W_0)/(L_iW_i)$ rounded down to the nearest integer, $i=1,...,m$.

Two-dimensional cutting problems, of various types, have been widely considered in the literature. Such problems are also commonly referred to as packing problems since they can be viewed:

(a) **either** as the problem of cutting smaller pieces from a larger stock rectangle; **or**

(b) as the problem of packing smaller pieces into a larger stock rectangle.

Dyckhoff [9] has provided a classification of the various types of cutting problem encountered. Dowsland and Dowsland [8], Haessler and Sweeney [11] and Sweeney and Paternoster [17] give a survey of work that has been done prior to the early 1990's.

Relatively few authors in the literature have considered general two-dimensional non-guillotine cutting problems (constrained or unconstrained). With respect to the unconstrained problem work has been presented by Tsai, Malstrom and Meeks [18] and Arenales and Morabito [1]. Tsai, Malstrom and Meeks [18] presented an integer programming formulation of the problem. Their work contains a number of deficiencies, as Dowsland and Dowsland [8] have commented. Arenales and Morabito [1] presented an approach based upon an AND/OR graph together with branch and bound. Computational results were presented for a number of test problems taken from the literature as well as for a number of randomly generated problems.

With respect to the constrained problem work has been presented by Beasley [3]; Hadjiconstantinou and Christofides [10] and Lai and Chan [13]. Beasley [3] presented a tree search approach based upon lagrangean relaxation of a zero-one integer linear programming formulation of the problem to derive a bound on the optimal solution. He presented computational results for the optimal solution of 12 different test problems. Hadjiconstantinou and Christofides [10] also used lagrangean relaxation to derive a bound on the optimal solution for use in a tree search procedure. Note here that their formulation of the problem as presented in [10] is incorrect, although the bound given is a valid bound. They presented optimal solutions for 7 different test problems, as well as the value of the best feasible solution found for a further 5 test problems which were artificially terminated by a computational time limit constraint. Lai and Chan [13] presented a heuristic algorithm for the case $P_i=0$ $\forall i$ based upon an evolutionary strategy approach. Their representation of the problem encodes the order in which pieces should be cut, each successive piece being cut (having regard to previously cut pieces) with its bottom-left corner as near as possible to the bottom-left corner of the stock rectangle. Computational results were presented for a number of randomly generated test problems with the objective of maximising the total area of the cut pieces (minimise trim loss).

In this paper we present a new, nonlinear, formulation of the constrained two-dimensional non-guillotine cutting problem. Based upon this formulation we present a population heuristic that, computationally, is able to find the optimal solution to standard test problems drawn from the literature very quickly. In the next section we present our formulation of the problem.

2. PROBLEM FORMULATION

2.1 Formulation

As we have a number (Q_i) of identical copies of each piece i we define:

z_{ip} = 1 if the p'th copy ($p=1,...,Q_i$) of piece i is cut from (L_0,W_0)

= 0 otherwise

where the position of any cut piece is taken with reference to the centre of the piece, i.e. let

x_{ip} be the x-coordinate of the centre of the p'th copy of piece i

y_{ip} be the y-coordinate of the centre of the p'th copy of piece i

Figure 3 illustrates the situation diagrammatically. These centre coordinates are limited by:

$$L_i/2 \leq x_{ip} \leq L_0 - L_i/2 \qquad i=1,...,m;\ p=1,...,Q_i \tag{1}$$

$$W_i/2 \leq y_{ip} \leq W_0 - W_i/2 \qquad i=1,...,m;\ p=1,...,Q_i \tag{2}$$

In order to ensure that we cut an appropriate minimum number of pieces of each type we have:

$$z_{ip} = 1 \qquad i=1,...,m;\ P_i>0;\ p=1,...,P_i \tag{3}$$

This constraint simply says that we (arbitrarily) chose to cut the first P_i copies of piece i.

Naturally pieces that are cut from the stock rectangle (L_0,W_0) cannot overlap and considering Figure 3 where we have shown two overlapping pieces (the p'th copy of i and the q'th copy of j) this can be expressed as the condition that we require:

$$|x_{ip} - x_{jq}| \geq (L_i + L_j)/2 \quad \text{or} \quad |y_{ip} - y_{jq}| \geq (W_i + W_j)/2 \tag{4}$$

i.e. that the difference between the centre x-coordinates is sufficient ($\geq (L_i+L_j)/2$) to ensure that the pieces do not overlap, ***or*** that the difference between the centre y-coordinates is sufficient ($\geq (W_i+W_j)/2$) to ensure that the pieces do not overlap. This overlap constraint was first given by Christofides [7] in the context of strip packing. To ease the notation define $\alpha_{ij} = (L_i+L_j)/2$ and $\beta_{ij} = (W_i+W_j)/2$, then the above condition (equation (4)) can be written as:

$$|x_{ip} - x_{jq}| - \alpha_{ij} \geq 0 \quad \text{or} \quad |y_{ip} - y_{jq}| - \beta_{ij} \geq 0 \qquad (5)$$

Hence our constraint to prevent overlap between cut pieces is:

$$\max[|x_{ip} - x_{jq}|-\alpha_{ij},\ |y_{ip} - y_{jq}|-\beta_{ij}]z_{ip}z_{jq} \geq 0 \qquad i=1,...m;\ j=1,...,m;\ p=1,...,Q_i;\ q=1,...,Q_j\ (i\neq j \text{ or } p\neq q) \qquad (6)$$

where the $z_{ip}z_{jq}$ term renders the constraint inactive unless $z_{ip}=z_{jq}=1$ (i.e. unless both the p'th copy of piece i and the q'th copy of piece j are cut).

Our complete formulation of the constrained two-dimensional non-guillotine cutting problem is therefore:

$$\text{maximise} \quad \sum_{i=1}^{m} \sum_{p=1}^{Q_i} v_i z_{ip} \qquad (7)$$

subject to

$$\max[|x_{ip} - x_{jq}|-\alpha_{ij},\ |y_{ip} - y_{jq}|-\beta_{ij}]z_{ip}z_{jq} \geq 0 \qquad i=1,...m;\ j=1,...,m;\ p=1,...,Q_i;\ q=1,...,Q_j\ (i\neq j \text{ or } p\neq q) \qquad (8)$$

$$z_{ip} = 1 \qquad i=1,...,m;\ P_i>0;\ p=1,...,P_i \qquad (9)$$

$$L_i/2 \leq x_{ip} \leq L_0 - L_i/2 \qquad i=1,...,m;\ p=1,...,Q_i \qquad (10)$$

$$W_i/2 \leq y_{ip} \leq W_0 - W_i/2 \qquad i=1,...,m;\ p=1,...,Q_i \qquad (11)$$

$$z_{ip} \in (0,1) \qquad i=1,...,m;\ p=1,...,Q_i \qquad (12)$$

There are several points to note about this formulation:

(a) If the p'th copy of piece i is not cut ($z_{ip}=0$) then the values of x_{ip} and y_{ip} are irrelevant as the overlap constraint (equation (8)) is automatically satisfied.

(b) The overlap constraint as presented above (equation (8)) contains redundancy since if we were to write out this overlap constraint in full for any particular example we would find exactly the same mathematical equation appearing twice (e.g for i=3,p=1 and j=7,q=2 we get the same equation as for i=7,p=2 and j=3,q=1). However, we prefer this overlap constraint in the form presented above for later use in our heuristic algorithm.

(c) The formulation given above is valid irrespective of whether the dimensions of the pieces (L_i,W_i) i=0,1,...,m are integers or not. This contrasts with the formulation of the problem presented previously in [3] which was dependent on the pieces having integer dimensions.

(d) The centre coordinates (x_{ip} and y_{jq}) can take fractional values; or if we wish to work with integer centre coordinates then without significant loss of generality simply ensure that all dimensions (L_i,W_i) i=0,1,...,m are integer and even (set $L_i=2*L_i$ and $W_i=2*W_i$ i=0,1,...,m).

Our formulation of the problem is a nonlinear one, involving as it does in equation (8) a maximisation of two modulus terms and the product of two zero-one variables. Defining $M=\sum_{i=1}^{m} Q_i$ then our formulation has 3M variables and $O(M^2)$ constraints (excluding bounds on variables). In the formulation of this problem given previously in Beasley [3] the number of variables and constraints was a function of the size of the stock rectangle (L_0,W_0). The formulation given above does not, in terms of the number of variables and constraints, involve the size of the stock rectangle.

Overall we can say that the formulation presented here is a "more compact" formulation than that given in [3], principally due to the representation of the overlap constraint in equation (8). This compactness has come at a price however, namely the use of a nonlinear overlap constraint. Normally within Operations Research nonlinear terms should be avoided if we are to develop computationally successful algorithms. In this particular case however a combination of the compact representation of the overlap constraint and the solution approach (a population heuristic) adopted does lead, as we shall see below, to a computationally successful heuristic algorithm.

Before presenting our heuristic however we shall briefly indicate how our formulation can be extended to deal with a number of situations encountered in two-dimensional non-guillotine cutting.

2.2 Extensions

In this section we indicate how our formulation can be extended to deal with defects and multiple stock rectangles. Each of these is discussed separately below.

2.2.1 Defects

Defects in the stock rectangle (L_0,W_0), i.e. areas of the stock rectangle which cannot be part of any cut piece, are easily dealt with in our formulation. Suppose, without significant loss of generality, that all defective areas can be decomposed into rectangular shapes, as in Figure 4. In that figure we have one defective area (the shaded portion) which has been decomposed into the two rectangular shapes shown. Each rectangular defective can be regarded as a rectangular piece which must be cut at a pre-specified position.

To illustrate how our formulation can be extended, suppose we regard one of the rectangular defectives in Figure 4 as piece k centred at (a_k,b_k). Set P_k=Q_k=1, so ensuring via equation (9) that the piece is cut, then the only changes necessary to our formulation are:

(a) to replace the bounds (equations (10) and (11)) on the centre coordinates of piece k by x_{k1}=a_k and y_{k1}=b_k; and

(b) not to impose the overlap constraint (equation (8)) for pairs of rectangular defectives (e.g. in Figure 4 we have two overlapping rectangular defectives).

The heuristic we present below is designed such that these changes are easily incorporated.

2.2.2 Multiple stock rectangles

Multiple stock rectangles can be easily dealt with by assembling them, in an arbitrary fashion, into one large rectangular "superstock" separated by defective areas. This is shown diagrammatically in Figure 5. Note here that, as in Figure 5, there is no requirement that the individual stock rectangles be of the same size. We can then simply apply our formulation, amended to deal with defective areas as discussed above, to the "superstock" rectangle.

2.3 Summary

In this section we have given our formulation of the constrained two-dimensional non-guillotine cutting problem and have indicated how it can be extended to deal with defects and with multiple stock rectangles. In the next section we briefly introduce population heuristics.

3. POPULATION HEURISTICS

In Operations Research (OR), over the years, a number of standard heuristic algorithms have appeared. In the early years of OR heuristics were especially tailored to the problem under consideration. As OR advanced a number of basic heuristic ideas began to be formalised. These included ideas such as greedy (seek the best possible improvement) and local interchange (e.g. swop two parts of a solution). Further advances came with schemes such as tabu search (accept a change in the solution, for better or worse, unless it is tabu) and simulated annealing (accept a change in the solution, for better or worse, according to some probabilistic criterion). These latter schemes are sometimes called metaheuristics as they provide general guidelines as to the approach to be followed, yet need tailoring for each specific problem under consideration.

All of the above have a common characteristic, namely that they have a single solution to a problem and continually seek to improve this single solution in some way. ***Population heuristics***, by contrast, explicitly work with a population of solutions and combine them together in some way to generate new solutions.

Population heuristics started with the development of genetic algorithms by Holland [12]. As the field advanced (evolved) new terms began to appear in the literature: hybrid genetic algorithms, memetic algorithms, scatter search algorithms, bionomic algorithms. Whilst there are differences between these approaches they can all be classified generically as population heuristics.

A population heuristic (henceforth abbreviated to PH) can be described as an "intelligent" probabilistic search algorithm and is based on the evolutionary process of biological organisms in nature. During the course of evolution, natural populations evolve according to the principles of natural selection and "survival of the fittest". Individuals who are more successful in adapting to their environment will have a better chance of surviving and reproducing, whilst individuals who are less fit will be eliminated. This means that the *genes* from highly fit individuals will spread to an increasing number of individuals in each successive generation. The combination of good characteristics from highly adapted parents may produce even more fit offspring. In this way, species evolve to become increasingly better adapted to their environment.

A PH simulates these processes in a computer by taking an initial population of individuals and applying genetic operators in each reproduction. In optimisation terms, each individual in the population is encoded into a string or *chromosome* which represents a possible *solution* to a given problem. The fitness of an individual is evaluated with respect to a given objective function. Highly fit individuals or *solutions* are given opportunities to reproduce by exchanging pieces of their genetic information, in a *crossover* procedure, with other highly fit individuals. This produces new "offspring" solutions (i.e. *children*), who share some characteristics taken from both parents. Mutation is often applied after crossover by altering some genes in the child strings. The offspring can either replace the whole population (*generational* approach) or replace less fit individuals (*steady-state* approach). This evaluation-selection-reproduction cycle is repeated until a satisfactory solution is found. The basic steps of a simple PH are:

```
Generate an initial population
Evaluate fitness of individuals in the population
repeat
-       Select individuals from the population to be parents
-       Recombine (mate) parents to produce children
-       Mutate the children
-       Evaluate fitness of the children
-       Replace some or all of the population by the children
until
-       You decide to stop whereupon report the best solution encountered
```

Readers wishing to learn more about population heuristics are referred to [2,6,14,15,16].

4. A POPULATION HEURISTIC FOR TWO-DIMENSIONAL CUTTING

In this section we will outline the population heuristic that we have developed for the constrained two-dimensional non-guillotine cutting problem, Below we discuss the elements of our PH relating to:

- representation, how we represent a solution to the problem in our PH;
- fitness, assessing the value of a solution;
- parent selection, choosing who shall have a child;
- crossover, having a child from parents;
- mutation, altering the child;
- unfitness, dealing with the constraints;
- improvement, improving the child; and
- population replacement, placing the child in the population.

4.1 Representation and fitness

The first decision in any PH is how to represent a solution to the problem. For the constrained two-dimensional non-guillotine cutting problem we used a combined binary/real-numbered representation.

The binary part of our representation is simply z_{ip}, which takes the values zero or one and indicates whether the p'th copy of piece i is cut or not.

Letting X_{ip} ($0 \le X_{ip} \le 1$) be the real-numbered representation of the x-coordinate value for the centre of the p'th copy of piece i then X_{ip} represents the proportion of the interval $[L_i/2, L_0-L_i/2]$ that lies before x_{ip}, i.e. $x_{ip} = L_i/2 + X_{ip}*(L_0-L_i)$. For example if $X_{ip}=0.73$, $L_0=10$ and $L_i=4$ then the x-coordinate of the centre of this piece lies 0.73 of the way between 2 ($L_i/2$) and 8 ($L_0-L_i/2$), i.e. $x_{ip} = 2 + 0.73*(10-4) = 6.38$ (rounded to the nearest integer 6 as we chose to work with integer centre coordinates).

In a similar fashion we let Y_{ip} ($0 \le Y_{ip} \le 1$) be the real-numbered representation of the y-coordinate value for the centre of the p'th copy of piece i so that Y_{ip} represents the proportion of the interval $[W_i/2, W_0-W_i/2]$ that lies before y_{ip}, i.e. $y_{ip} = W_i/2 + Y_{ip}*(W_0-$

W_i). For example if Y_{ip}=0.21, W_0=30 and W_i=6 then the y-coordinate of the centre of this piece lies 0.21 of the way between 3 (W_i/2) and 27 (W_0-W_i/2), i.e. y_{ip} = 3 + 0.21*(30-6) = 8.04 (rounded to the nearest integer 8).

In order to clarify our representation we show below a sample complete representation for a problem with m=2, Q_1=3 and Q_2=2:

Piece (i)	1	1	1	2	2
Copy (p)	1	2	3	1	2
z_{ip}	1	0	0	1	0
X_{ip}	0.73	0.34	0.27	0.56	0.49
Y_{ip}	0.21	0.97	0.88	0.94	0.11

In this example the first copy of piece 1 is cut (z_{11}=1) with its centre coordinates at the appropriate integer position revealed by decoding X_{11}=0.73 and Y_{11}=0.21. The only other piece cut is the first copy of piece 2, at the appropriate integer position revealed by decoding X_{21} and Y_{21}.

The representation we have adopted has the property that we ensure that the constraints on the centre coordinate positions (equations (10) and (11)) are automatically satisfied. Note here that if defective rectangles (e.g. piece k, copy 1) are present, as discussed above, we simply decode any values of X_{k1} and Y_{k1} to be the pre-specified centre coordinates of the defective rectangle.

Fitness in PHs relates to assessing the value (worth) of an individual. In our PH for the constrained two-dimensional non-guillotine cutting problem the fitness of an individual was given by the objective function value (equation (7)), which we are trying to maximise.

4.2 Parent selection, crossover and mutation

In our PH we have two parents coming together to have a single child. Our two parents were chosen randomly from the population. In order to produce a child from these two parents we used uniform crossover. This works by considering each element in our representation in turn (i.e. each copy of each piece) and choosing values taken from either the first or second parent at random. Taking our example from before if we have:

	Piece (i)	1	1	1	2	2
	Copy (p)	1	2	3	1	2
Parent 1	z_{ip}	1	0	0	1	0
	X_{ip}	0.73	0.34	0.27	0.56	0.49
	Y_{ip}	0.21	0.97	0.88	0.94	0.11
Parent 2	z_{ip}	0	1	0	1	1
	X_{ip}	0.13	0.45	0.32	0.99	0.34
	Y_{ip}	0.23	0.43	0.85	0.57	0.69

and our random choice of parents is:

piece 1 copy 1 - parent 1
piece 1 copy 2 - parent 2
piece 1 copy 3 - parent 2
piece 2 copy 1 - parent 1
piece 2 copy 2 - parent 2

then the child is:

Piece (i)	1	1	1	2	2
Copy (p)	1	2	3	1	2
z_{ip}	1	1	0	1	1
X_{ip}	0.73	0.45	0.32	0.56	0.34
Y_{ip}	0.21	0.43	0.85	0.94	0.69

Mutation in our PH was applied to only one in ten children. If a child was selected for mutation then a randomly selected piece/copy (piece j, copy q, say) in that child had its z_{jq} value set to zero (i.e. if the piece/copy was being cut then it was no longer cut).

4.3 Unfitness

Unfitness in a PH relates to dealing with the constraints. Considering our formulation of the problem given above (equations (7)-(12)) we have that the constraints on the centre coordinates (equations (10) and (11)) are automatically satisfied by virtue of the representation we have adopted (as remarked previously). This leaves two sets of constraints, equation (9) which ensures that we cut an appropriate minimum number of pieces of each type and equation (8) which is the overlap constraint.

To deal with equation (9) we apply a simple adjustment heuristic to the child. Simply consider each piece i in turn in the child and set $z_{ip}=1\ \forall p \leq P_i$. This simple heuristic will automatically ensure that equation (9) is satisfied by the child.

In order to deal with the overlap constraint (equation (8)) we associate with each individual in our PH two numbers: **(unfitness,fitness)** where fitness is our adopted objective function value (which we are trying to maximise) and unfitness is a measure of constraint violation.

Considering equation (8) we have that a convenient measure of constraint violation is given by:

$$\sum_{\substack{i=1 \\ i\neq j \text{ or } p\neq q}}^{m} \sum_{j=1}^{m} \sum_{p=1}^{Q_i} \sum_{q=1}^{Q_j} \max[0, -\max[|x_{ip} - x_{jq}|-\alpha_{ij}, |y_{ip} - y_{jq}|-\beta_{ij}]z_{ip}z_{jq}] \qquad (13)$$

which will be zero if the solution is feasible, nonzero (strictly positive) otherwise. This quantity is the *unfitness* associated with any PH solution and is a measure of constraint infeasibility. Informally the higher the unfitness value the more infeasible a solution.

Note here that if defective rectangles are present (as discussed above) we simply amend our definition of unfitness to exclude from equation (13) any contribution from pairs of defective rectangles (since these may overlap by definition, see Figure 4).

This use of (unfitness,fitness) allows us to design our PH to evolve feasible solutions (i.e. solutions with unfitness zero) through an appropriate population replacement scheme (see below).

4.4 Improvement

Once we have our child we may be able to improve it (reduce its unfitness if it is infeasible). In order to do this we take the centre coordinates of each piece/copy (piece i, copy p, say) that is cut (i.e. has z_{ip}=1) and examine moving the piece/copy both left/right and up/down. There are four possible simple moves for the centre (x_{ip},y_{ip}) of the p'th copy of piece i:

(a) $x_{ip}=x_{ip}-1$ (move left)

(b) $y_{ip}=y_{ip}-1$ (move down)

(c) $x_{ip}=x_{ip}+1$ (move right)

(d) $y_{ip}=y_{ip}+1$ (move up).

With respect to the first two of these moves if they reduce (or leave unchanged) the unfitness then the move is made. The last two of these moves are only made if they reduce the unfitness. The reason for this distinction here is to prevent cycling and to seek a "bottom-left" cutting pattern. These moves are repeated for each cut piece/copy in turn until no improvement in the unfitness can be made.

Note here that some moves may not be possible (e.g. if $x_{ip}=L_i/2$ then it is not possible to move the piece/copy left). Note too here that a naive evaluation of the unfitness (equation (13)) for the child after a move would require $O(M^2)$ operations. However, by recording the contribution to unfitness made by each piece/copy, it is possible to re-evaluate unfitness after a move in just O(M) operations.

After the above improvement procedure the child may be feasible (have unfitness zero, and be a bottom-left cutting pattern). Such a child may improve upon the best feasible solution found previously. If this is so we examine the child further to see if this new improved feasible solution can be further improved by cutting, from any "empty spaces", pieces that are currently uncut. However, in order not to perturb the convergence of our PH, we do not alter the child to reflect any such pieces which are cut.

4.5 Population replacement

We used a steady-state population replacement scheme that makes use of the two numbers (unfitness,fitness) associated with each PH solution. Suppose that we have produced a child in our PH. This child will have certain (unfitness,fitness) values. For the purposes of illustration we shall assume that this child has unfitness > 0, i.e. is not feasible.

In order to explain our population replacement scheme suppose we draw a graph with fitness as the vertical axis and unfitness as the horizontal axis, as in Figure 6. Now plot on this graph:

(a) the PH child; and

(b) each member of the population, as each population member will also have (unfitness,fitness) values. For the purposes of illustration these population members are shown as square boxes in Figure 6, scattered around the child.

Considering Figure 6 it is clear that the presence of the child has naturally divided the population into four groups corresponding to each of the four quadrants created by drawing horizonal and vertical lines through the child. These four groups are labelled G1 to G4 in Figure 6. Informally it is clear that:

(a) our child is superior to any member of the population that is in group G1, as all members of that group are more infeasible (have a higher unfitness) and are worse in objective function terms (have a lower fitness)

(b) our child is inferior to any member of the population that is in group G4, as all members of that group are less infeasible (have a lower unfitness) and are better in objective function terms (have a higher fitness)

(c) our child is superior with respect to one measure (fitness or unfitness), but inferior with respect to the other, when compared to members of groups G2 or G3.

Now as we have a child we need to add this child to the population. To keep the population size constant (the standard PH assumption) we need to choose a member from the population to kill (i.e. to replace the chosen member by the child).

We use our four groups (G1 to G4) in order to establish in which "area" of the population we look to find a member to kill. In our population replacement scheme we:

(a) first look in group G1. As stated above any population member in this group is worse than the child on both measures (fitness and unfitness). Hence we should plainly choose a member of the population from this group to kill if we can do so.

(b) it may be however that G1 is empty (i.e. contains no population members). If this is the case we look in G2, members of this group are inferior to the child in terms of unfitness.

(c) if both G1 and G2 are empty we look in G3, members in this group are inferior to the child in terms of fitness.

(d) finally if G1, G2 and G3 are all empty we look in G4. Members of this group are, as mentioned above, superior to the child on both measures (fitness and unfitness). It might appear strange therefore to choose a member of this group to kill. However, computational experience with this population replacement scheme has been that unless we perturb the population by placing the child in the population the PH fails to make progress.

Once the appropriate group (G1, G2, G3 or G4) has been identified as outlined above we kill a randomly selected member of the group.

There is one complication that must be mentioned here, namely that a child could be a duplicate of a member of the population. By duplicate we mean have exactly the same numeric representation, or have a representation that in terms of the underlying cutting pattern is equivalent. For example consider:

	Piece (i)	1	1	1	2	2
	Copy (p)	1	2	3	1	2
Child	z_{ip}	0	1	0	1	0
	X_{ip}	0.23	0.73	0.23	0.56	0.49
	Y_{ip}	0.18	0.21	0.88	0.94	0.11
Population member	z_{ip}	1	0	0	0	1
	X_{ip}	0.73	0.34	0.55	0.23	0.56
	Y_{ip}	0.21	0.97	0.83	0.48	0.94

Here the child is a duplicate of this population member since the cut pieces associated with each (and their positions) are identical. In order to prevent the population coming to consist of copies of the same cutting pattern any child which was a duplicate of any member of the population was rejected and was not allowed to enter the population.

To assist in duplicate identification, as well as in crossover, the copies of each piece i in the representation were sorted by decreasing z_{ip} values, ties broken firstly by increasing X_{ip} values and secondly by increasing Y_{ip} values. For the example given above the child and population member after such sorting would be:

	Piece (i)	1	1	1	2	2
	Copy (p)	1	2	3	1	2
Child	z_{ip}	1	0	0	1	0
	X_{ip}	0.73	0.23	0.23	0.56	0.49
	Y_{ip}	0.21	0.18	0.88	0.94	0.11
Population member	z_{ip}	1	0	0	1	0
	X_{ip}	0.73	0.34	0.55	0.56	0.23
	Y_{ip}	0.21	0.97	0.83	0.94	0.48

It can be seen here how this sorting helps to more clearly identify duplicates.

4.6 Summary

Our complete PH for the constrained two-dimensional non-guillotine cutting problem can be summarised as follows:

(a) Generate an initial population. In the computational results given below we used a randomly generated initial population with a population size of 100.

(b) Evaluate fitness and unfitness of individuals in the population.

repeat

(c) Select individuals from the population to be parents, parents were randomly selected.

(d) Recombine (mate) parents to produce children, parents were combined using uniform crossover to produce a single child.

(e) Mutate the child, one in ten children were mutated with a single randomly chosen piece/copy being no longer cut.

(f) Evaluate the fitness and unfitness of the child and improve the child using the heuristic improvement scheme.

(g) Replace one member of the population by the child using the population replacement scheme discussed, unless the child is a duplicate of a member of the population in which case reject it.

until

(h) You decide to stop whereupon report the best solution encountered.

In the computational results reported below we repeatedly applied this PH in the following manner:

(1) from an initial population apply the procedure given above until 5000 children have been generated; and then

(2) restart the PH from scratch using a new initial population.

5. COMPUTATIONAL RESULTS

Our population heuristic was coded in FORTRAN and run on a Silicon Graphics O2 workstation (R10000 chip, 225MHz, 128MB main memory). In order to test the effectiveness of our PH we solved the 12 test problems given in Beasley [3] which are publically available from OR-Library [4,5], email the message *ngcutinfo* to *o.rlibrary@ic.ac.uk* or see *http://mscmga.ms.ic.ac.uk/jeb/orlib/ngcutinfo.html.* As the optimal solution for each of these test problems is known (see [3]) we adopted the approach of terminating our PH when it discovered the optimal solution.

Our results are shown in Table 1. In that table we show, for each of the 12 problems, the size of the problem, the number of children generated before the optimal solution was discovered and the total computation time (in seconds). It is clear from Table 1 that for the vast majority of the problems the optimal solution is discovered very quickly.

One aspect of our overall heuristic that is of interest is whether the PH is itself adding value. We could argue that the effectiveness of our heuristic is solely due to our child improvement scheme. To investigate this issue we randomly generated child solutions (each child being feasible in terms of the total area of cut pieces being $\leq L_0W_0$) and subjected each such child to our improvement scheme. We did this for each of our test problems and set a time limit for each problem equal to 10 multiplied by the time given in Table 1 for our PH. The results were that for only four problems (problems 4, 5, 7 and 10) was the optimal solution discovered. This is a clear indication that the success of our PH is due to more than just our improvement heuristic.

6. CONCLUSIONS

In this paper we have presented a new nonlinear formulation for the constrained two-dimensional non-guillotine cutting problem. Based upon this formulation we presented a population heuristic for the problem. Computational results indicated that this heuristic could effectively solve standard test problems taken from the literature in a matter of seconds.

REFERENCES

[1] M. Arenales and R. Morabito, "An AND/OR-graph approach to the solution of two-dimensional non-guillotine cutting problems", *European Journal of Operational Research* 84 (1995) 599-617.

[2] T. Bäck, D.B. Fogel and Z. Michalewicz (eds), *Handbook of evolutionary computation*. Oxford University Press (1997).

[3] J.E. Beasley, "An exact two-dimensional non-guillotine cutting tree search procedure", *Operations Research* 33 (1985) 49-64.

[4] J.E. Beasley, "OR-Library: distributing test problems by electronic mail", *Journal of the Operational Research Society* 41 (1990) 1069-1072.

[5] J.E. Beasley, "Obtaining test problems via Internet", *Journal of Global Optimization* 8 (1996) 429-433.

[6] J.E. Beasley, "Population heuristics", forthcoming in *Handbook of applied optimization*, P.M. Pardalos and M.G.C. Resende (eds). Oxford University Press (2000).

[7] N. Christofides, "Optimal cutting of two-dimensional rectangular plates", *CAD74 Proceedings* (1974) 1-10.

[8] K.A. Dowsland and W.B. Dowsland, "Packing problems", *European Journal of Operational Research* 56 (1992) 2-14.

[9] H. Dyckhoff, "A typology of cutting and packing problems", *European Journal of Operational Research* 44 (1990) 145-159.

[10] E. Hadjiconstantinou and N. Christofides, "An exact algorithm for general, orthogonal, two-dimensional knapsack problems", *European Journal of Operational Research* 83 (1995) 39-56.

[11] R.W. Haessler and P.E. Sweeney, "Cutting stock problems and solution procedures", *European Journal of Operational Research* 54 (1991) 141-150.

[12] J.H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press (1975).

[13] K.K. Lai and W.M. Chan, "An evolutionary algorithm for the rectangular cutting stock problem", *International Journal of Industrial Engineering* 4 (1997) 130-139.

[14] M. Mitchell, *An introduction to genetic algorithms*. MIT Press (1996).

[15] C.R. Reeves, "Genetic algorithms", in C.R. Reeves (ed), *Modern heuristic techniques for combinatorial problems*. Blackwell Scientific Publications, Oxford

(1993) 151-196.

[16] C.R. Reeves, "Genetic algorithms for the Operations Researcher", *INFORMS Journal on Computing* 9 (1997) 231-250.

[17] P.E. Sweeney and E.R. Paternoster, "Cutting and packing problems: a categorized, application-orientated research bibliography", *Journal of the Operational Research Society* 43 (1992) 691-706.

[18] R.D. Tsai, E.M. Malstrom and H.D. Meeks, "A two-dimensional palletizing procedure for warehouse loading operations", *IIE Transactions* 20 (1988) 418-425.

Problem	Problem size			Number of children generated	Total time (seconds)
	(L_0,W_0)	m	M		
1	(10,10)	5	10	358	0.04
2		7	17	6019	0.52
3		10	21	5471	0.64
4	(15,10)	5	7	1	0.02
5		7	14	625	0.07
6		10	15	5693	0.51
7	(20,20)	5	8	1	0.01
8		7	13	35037	3.31
9		10	18	65613	7.20
10	(30,30)	5	13	3	0.02
11		7	15	5245	0.56
12		10	22	10604	1.36
Average				11223	1.19

Table 1: Computational results