Comparison of Meta-Heuristic Algorithms for Clustering Rectangles

Edmund Burke

The University of Nottingham,
Nottingham NG7 2RD, UK
ekb@cs.nott.ac.uk

Graham Kendall

The University of Nottingham,
Nottingham NG7 2RD, UK
gxk@cs.nott.ac.uk

ABSTRACT

In this paper we consider a simplified version of the stock cutting (two-dimensional bin packing) problem. We compare three meta-heuristic algorithms (genetic algorithm (GA), tabu search (TS) and simulated annealing (SA)) when applied to this problem. The results show that tabu search and simulated annealing produce good quality results. This is not the case with the genetic algorithm.
The problem, and its representation, is fully described along with key test results.

KEYWORDS

Genetic Algorithm, Optimisation, Simulated Annealing, Stock Cutting, Tabu Search, Two Dimensional Bin Packing

INTRODUCTION

The stock cutting problem has been tackled in various ways. Some approaches, using mathematical programming techniques can be used to solve the problem to optimality but computational times are usually excessive. Heuristic techniques have been used since the mid 70's and one approach sorts the shapes into some order (for example, ascending order of area) and then packs the shapes as near to the bottom left of the bin as possible. Meta-heuristic approaches have been applied since the mid 80's. These include GA's, TS and SA.

This paper does not directly address the stock cutting problem. Rather, it takes a simplified problem and investigates three meta-heuristic techniques to solve it. The aim of solving a simplified version of the problem is to ascertain whether or not meta-heuristic techniques offer a sensible approach for solving this type of problem. We know (Blazewick, 1993), (Jain, 1992), (Kröger, 1995), (Parada, 1998) that these techniques have had success in this area but the three techniques have not been rigorously compared on the particular problem that we are interested in. If these approaches show promise on a simplified version of the problem this would indicate that it is worthwhile investigating the same techniques for more difficult problems.

THE PROBLEM

Presented with two rectangles (R_1, R_2) we can make some observations. By placing R_1 and R_2 in contact with each other we can place a bounding box, B_{box}, around them. The area of B_{box}, Area(B_{box}), depends on how the rectangles are placed in relation to one another. The minimum for Area(B_{box}) = Area(R_1) + Area(R_2). However this may not be achievable.

The following figures show these observations.

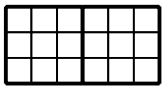

Fig. 1.1

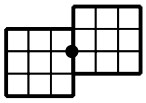

Fig. 1.2

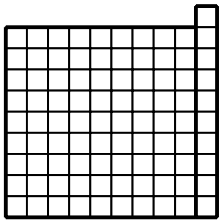

Fig. 1.3

Figure 1.1 shows two rectangles of size 3x3. The sum of their areas is 18, as is the area of their bounding box. Figure 1.2 shows the same two rectangles but now placed in relation to each other so as to give the bounding box an area of 24. The two rectangles (9x9 and 1x10) in Fig. 1.3 are positioned in an optimal way so that the area of the bounding box is minimized, but the area of the bounding box is still greater than the area sum of the two rectangles. In fact, the area of this bounding box will always be greater than the area sum of the rectangles, no matter how they are placed in relation to one another. Our simplified version of the stock cutting problem is to take pairs of rectangles from a given set and combine them with the objective of minimising the sum of their bounding boxes.

REPRESENTATION OF THE PROBLEM

We define variables to represent a rectangle, using cartesian co-ordinates, and also represent points on the edges of the rectangle. For example, to reference a point on the right hand edge of a rectangle we can use a variable related to y (we assume that x = max(x)). We can similarly define variables for points on the top, bottom and left of the rectangle. One further variable defines where another rectangle should be placed in relation to itself. We summarise these variables below

x	:	The width of the rectangle
y	:	The height of the rectangle
Bottom	:	0 >= Bottom <= x-1. Defines a point on the bottom of the rectangle
Left	:	0 >= Left <= y-1. Defines a point on the left of the rectangle
Right	:	0 >= Right <= y-1. Defines a point on the Right of the rectangle
Top	:	0 >= Top <= x-1. Defines a point on the Top of the rectangle
Place	:	A variable indicating if another rectangle should be placed on top, or to the right, of this one.

Interpretation

Given the rectangles $R_{1..}R_{6,}$ we can pair them as follows; R_1 & R_2, R_3 & R_4 and R_5 & R_6 (and in any other permutation). Given these pairings we can decide how any two rectangles can be placed in relation to one another. Assume R_1 and R_2 have the following values.

R_1 = {x=3, y=3, Bottom=0, Left=2, Right=2, Top=1, Place=On Right}
R_2 = {x=3, y=3, Bottom=1, Left=1, Right=0, Top=3, Place=On Top}

We can place these two rectangles in relation to one another using the following reasoning. R_2 (being the second rectangle of the pair) will be placed in relation to R_1. Using the Place variable of R_1 we note that R_2 should be placed to the right of R_1. We use the Right variable of R_1 to define where R_2 should *joined*. We also need a point on R_2 so that the *join* can be completed. This is given by the Left variable of R_2. We use the Left variable as it is the complement of Right. If we had used the Top variable of R_1 we would use the Bottom variable of R_2. The *joined* rectangle is shown in Fig. 1.2. The filled circle represents the join.

COMPARISON OF META-HEURISTIC ALGORITHMS

To compare the meta-heuristic algorithms we presented three rectangle pairing problems to the three meta-heuristic algorithms (GA, SA, TS). The aim was to observe the quality of the solutions produced when the algorithms attempted to minimise the sum of the bounding boxes of the rectangle pairs in the given set. The first problem consisted of twelve rectangles (2 of 9x9, 2 of 8x8, 4 of 4x4 and 4 of 3x3). The optimum solution for this problem is 390. All the algorithms were able to find the optimum for this problem. The second problem consisted of 50 rectangles where the optimum was known to be 20000. A third problem consisted of 100 random rectangles where the optimum was not known. The results for the 50 and 100 rectangle problem are shown below, with comments following. All results are averaged over ten runs.

Table 1 : Pairing Rectangles using Meta-Heuristic (GA, TS, SA) Algorithms

	GA		TS		SA	
	Best	Avg	Best	Avg	Best	Avg
50 Rectangles	22053	22149	20024	20099	20047	20089
100 Rectangles	79586	80356	75269	75949	73845	74003

Crossover, Neighbour and Mutation

In testing the algorithms we used three types of crossover operator for the GA. One was a generic order based crossover. The other two crossover operators were developed specifically for this problem. The neighbour for both SA and TS was a random swap of two rectangles (making allowances for the tabu list in TS). Mutation, for all the algorithms, changed one of the variables in a randomly selected rectangle to a random, allowed, value.

Genetic Algorithm (GA)

Initially we carried out GA runs by testing various parameter combinations. The parameters we varied were the population size (50 and 100), the three types of crossover operator, two types of evaluation (fitness is evaluation and linear normalization) and elitism (0.05 or 0.00). The

permutation of these parameters led to 24 runs being necessary. Each run was carried out fifteen times using three different starting populations (360 tests in all). Some parameters to the GA algorithm remained constant. These were the crossover probability (0.6), the mutation probability (0.05) and the number of generations (100).

The order based crossover did not yield good results. Using the problem specific operators yielded better quality solutions. Linear normalization produced better results than fitness is evaluation. As might be expected a population size of 100 produced better results than a population size of 50. Using elitism, better results were produced than when not using elitism. The results shown above used the best parameter combinations we found. However, even with these combinations, TS and SA outperform the GA.

Tabu Search (TS)

To produce the results shown above, TS was run with a list size of 50% of its population size and a neighbourhood size equal to its population size (that is, when looking for a better neighbour at each iteration it considered *n* neighbours). 5000 iterations were performed. To confirm that the TS list was helping find good quality solutions we ran a test, for the 50 rectangle problem, with the list size set to zero. This yielded a result of 20828. We also tried a list size of 48, which gave a result of 20506. A smaller list size (10) produced an average result of 20632 and increasing the neighbourhood size to 100 (with the list size still at 25) produced similar results to a neighbourhood size of 50.

Simulated Annealing (SA)

To produce the results we used a starting temperature of 10, a decrement of 0.001 and did 100 iterations at each temperature.

CONCLUSIONS

All three meta-heuristic techniques were able to find the optimum solution to a small problem. When presented with larger problems, tabu search and simulated annealing produced good quality solutions whereas a genetic algorithm produced lower quality solutions despite many tests during the small problem to try and ascertain the best combination of parameters. In addition, the GA runs took longer due to the overheads involved in processing a population of chromosomes, rather than a single individual. In conclusion, for this problem the SA and TS algorithms produce similar solutions and both outperform the GA.

REFERENCES

Blazewicz, J., Hawryluk, P., Walkowiak, R. (1993). Using a Tabu Search for Solving Two-Dimensional Irregular Cutting Problem. *Annals O.R.* 41, 313-327

Jain, P., Fenyes, P., Richter, R. (1992). Optimal Blank Nesting Using Simulated Annealing. *Trans. of ASME*, 114, 160-165

Kröger, B. (1995). Guillotineable Bin Packing : A Genetic Approach. *EJOR (Special Issue)*, 84, 645-661

Parada, V., Sepúlveda, M., Solar, M. (1998). Solution for the Constrained Guillotine Cutting Problem by Simulated Annealing. *Computer Ops Res*, 25, 37-47