Heuristics for Large Strip Packing Problems with Guillotine Patterns: an Empirical Study

Christine L. Valenzuela [*] Pearl Y. Wang [†]

[*] Department of Computer Science, Cardiff University
PO Box 916, Cardiff CF24 3XF, United Kingdom
Email: christine@cf.cs.ac.uk

[†] Department of Computer Science MS4A5, George Mason University
Fairfax, VA 22030-4444, USA
Email: pwang@cs.gmu.edu

1 Introduction

In this paper we apply a variety of simple heuristics to strip packing problems with guillotine patterns (also known as slicing floorplans) and compare the results to those obtained using a genetic algorithm (GA). Although the GA performs well on smaller problem sizes, our results confirm that the simple heuristics become more effective and outperform the GA as the problem size increases.

1.1 Preamble

We undertake an empirical study in which the performance of a good genetic algorithm is tested against some well-known bin packing heuristics on a range of two-dimensional bin packing problems. We restrict our study to problems involving *guillotine patterns* which are produced using a series of vertical and horizontal edge–to–edge cuts. Many applications of two-dimensional cutting and packing in the glass, wood, and paper industries, for example, are restricted to guillotine cutting.

The problem under consideration involves the packing of a set of rectangles into a bin of given width and infinite height so that no rectangles are overlapping. The goal of this strip packing problem is to minimize the height of the packing. We compare the performance of our GA, developed for placement and packing problems [9, 10] with the performance of some level-packing algorithms [1] and the Split algorithm [3] on data sets of various sizes with a variety of characteristics.

A suite of data generation programs have been developed in [11] that produce data sets with optimal guillotine packings of zero waste. The software allows a set of basic rectangles to be cut from a large enclosing rectangle of given dimensions, and options are available which allow the user to control various characteristics: the number of pieces in the data set, the maximum and minimum height/width ratios of the pieces, and the ratio of the largest piece to the smallest piece cut.

Several recent comparative studies on strip packing have reported superior performances for meta-heuristic algorithms over simple heuristic approaches (for examples see [4, 5, 7]). We note, however, that in most of these cases, the problem sizes are restricted to 100 rectangles or less. Our interests lie in examining a larger range of problem sizes and types.

2 Simple Heuristic Algorithms for Guillotine Packing

In this section we review the simple heuristic algorithms used for our comparative study. Typically, the sets of rectangles to be packed are first preprocessed by arranging them in order of non–increasing height or width, and then placing them in the bin, one at a time, in a simple deterministic fashion. Two of the heuristic algorithms we use are based on the so-called *level oriented* heuristics which were introduced by Coffmann *et al* [2]. The third algorithm, first described by Golan [3], relies on repeatedly splitting the bin into smaller rectangles (or 'bins'), and packing the pieces, sorted by decreasing width, into ever narrower bins as the algorithm progresses. All three heuristic algorithms produce guillotine patterns.

2.1 The Level Oriented Algorithms

To implement a level oriented algorithm, the items are first preordered by non–increasing height, and then the packing is constructed as a series of *levels*, each rectangle being placed so that its bottom rests on one of these levels. The first level is simply the bottom of the bin. Each subsequent level is defined by a horizontal line drawn through the top of the tallest rectangle on the previous level. In the *Next Fit Decreasing Height (NFDH)* algorithm rectangles are packed left justified on a level until the next rectangle will not fit, in which case it is used to start a new level above the previous one, on which packing proceeds. The run time complexity of NFDH (excluding the sort) is linear, just placing one rectangle after another in sequence. The *First Fit Decreasing Height (FFDH)* algorithm places each rectangle left justified on the first (i.e. lowest) level in which it will fit. If none of the current levels has room, then a new level is started. The run time complexity for FFDH is $O(n \lg n)$ where n is the number of rectangles being packed. Figure 1 illustrates a typical FFDH packing for 50 rectangles.

For the level packing algorithms the asymptotic worst case performance is known to be twice the optimum height for the NFDH algorithm and 1.7 times the optimum height for the FFDH algorithm.

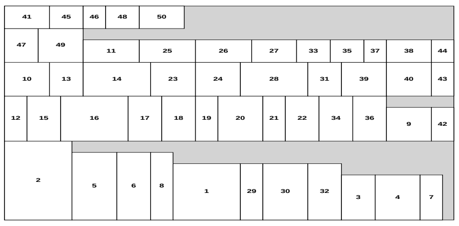

Figure 1: A FFDH packing for 50 rectangles

2.2 The Split Algorithm

The Split Packing algorithm is more complicated than the level oriented algorithms. It packs the pieces in order of non–increasing width. We can imagine that for each piece that is packed the original bin is split into two, and then into two again when the next piece is packed, and so on. As the rectangles to be packed are sequenced according to width, after packing some pieces, those left to be packed are narrower and thus easy to fit into one of newly created bins. If possible we pack pieces side by side with previously packed pieces, when this is not possible we pack pieces on top of previously packed pieces. When pieces are packed next to each other, closed bins are created in which no further pieces can be packed. The worst–case performance of Split pack is three times the optimum height. Full details of the Split algorithm can be obtained from [3]. Figure 2 illustrates a typical Split algorithm packing for 50 rectangles.

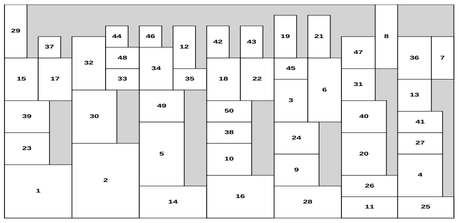

Figure 2: A Split algorithm packing for 50 rectangles

3 Our Genetic Algorithm

Our genetic algorithm (GA) for solving packing and placement problems is based on a normalized postfix representation, which offers a unique encoding for each guillotine pattern and covers the search space efficiently. The postfix representation provides a blueprint for the recursive bottom–up construction of a packing or placement, by combining rectangles together in pairs. Our approach has proven effective for the VLSI floorplanning problem, and we refer the interested reader to our earlier papers [9, 10]. Most of the details given earlier for our representation, decoder, and GA apply equally to the current study. There are a few minor modifications to note, however. Firstly, the GA described in our earlier papers was designed for placing so–called *soft modules*, which are rectangles which have a fixed area but flexible height and width dimensions. The earlier GA includes an area optimization routine which optimizes the height and width dimensions of the soft modules. This routine is not needed for the present study. Secondly, the GA handles the fixed width constraint in the strip packing problem by rejecting packings that are too wide, and repeatedly generates new ones until the width constraint is satisfied. Finally, for the present study we have incorporated a rotation heuristic into our GA. This performs rotations where this is locally effective, and produces better results than the earlier version of our GA.

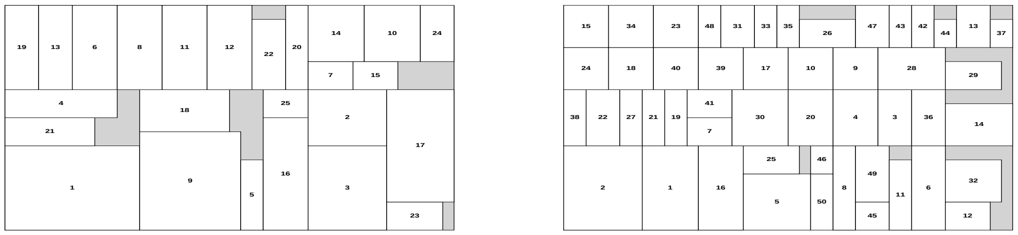

Figure 3: GA packings for 25 and 50 rectangle problems from [6]: width = 40, height = 16

Our current aim is to compare the performance of a good genetic algorithm with the classical strip packing heuristics on a range of data sets of different sizes with different characteristics. Evidence that our GA is a good one is provided by the excellent results we obtained in [10], and also by the results we were able to attain using the 25 and 50 rectangle problems from [6]. Our solutions matched the best obtained by Liu and Teng in a recent paper [8]: a height of 16 is obtained for both problems using a GA and a free orientation approach. We obtained these results despite the use of a much more restrictive search engine – in [8] the non-guillotine Bottom Left heuristic [1] is used; our search space was confined to guillotine patterns. Figure 3 shows some typical packings obtained by our GA for these problems.

3.1 Results

The tables in this section give some results for strip packing on guillotine patterns cut from 100 x 100 enclosing rectangles. Thus the width of the strip is 100 and the optimum height is 100. All our results are quoted in heights.

For each Nice.n data set, the height/width ratio of all n rectangular pieces in the set lies in the range $1/4 \leq H/W \leq 4$ and the maximum area ratio is 7. This value is the limiting ratio of the areas of the largest and smallest rectangles in the data set. For the Path.n (i.e. pathological) data, the H/W ratio is in the range $1/100 \leq H/W \leq 100$ and the maximum area ratio also 100. All the results are rounded to the nearest integer.

Table 1: Results for Heuristics on Nice Data (optimum height is 100)

Problem	NFDH			FFDH			SPLIT		
	Raw	$W \geq H$	$H \geq W$	Raw	$W \geq H$	$H \geq W$	Raw	$W \geq H$	$H \geq W$
Nice.25	147	133	137	145	118	132	162	138	143
Nice.50	127	120	127	125	119	126	139	134	142
Nice.100	117	112	118	115	111	116	145	137	137
Nice.200	122	110	117	121	108	117	141	139	138
Nice.500	109	108	108	108	107	107	140	139	141
Nice.1000	110	105	108	109	105	107	138	140	140

Table 2: Results for Heuristics on Pathological Data (optimum height is 100)

Problem	NFDH			FFDH			SPLIT		
	Raw	$W \geq H$	$H \geq W$	Raw	$W \geq H$	$H \geq W$	Raw	$W \geq H$	$H \geq W$
Path.25	171	132	160	158	120	159	182	136	176
Path.50	143	153	152	131	136	146	184	154	158
Path.100	157	120	161	154	109	155	174	137	181
Path.200	145	128	139	140	116	138	151	138	155
Path.500	143	112	141	141	105	141	154	141	158
Path.1000	131	110	129	129	107	128	147	137	139

Tables 1 and 2 are concerned with the level algorithms and the Split algorithm only, and cover the Nice.n and the Path.n data sets, respectively. The Raw data column gives results for sets of rectangles sorted in decreasing height for the level algorithms and decreasing width for the Split algorithm. The column headed '$W \geq H$' refers to preprocessing of the sets of rectangles so that tall rectangles are rotated through 90^o. The column headed '$H \geq W$' indicates that the preprocessing rotates wide rectangles through 90^o. Following this preprocessing, the level and Split algorithms sort the rectangles on decreasing width or height as before.

From Tables 1 and 2 we can see that, of the three heuristics, FFDH performs the best. Furthermore, it is clear that preprocessing the data by rotating rectangles to ensure $W \geq H$ improves the results significantly for both NFDH and FFDH. In addition, we can observe that the level heuristics (NFDH and FFDH) become more effective as the problem size increases, but the Split algorithm performs relatively poorly throughout the range of problem instances under test. Run times for NFDH, FFDH and the Split algorithm take less than one second for the data sets on a Pentium III computer.

Table 3 shows the results of our GA compared to the results obtained from the best simple heuristic algorithm, which is the FFDH algorithm, with $W \geq H$. The GA is run until there has been no improvement to the best solution for 100 generations. Where feasible, 5 replicated runs have been carried out for the GA and the solutions have been averaged. Single runs have been carried out for the 500 rectangle problems and for Nice.200 because of its lengthy run time. We have not attempted to run the GA on the 1,000 rectangle problems. Run times for the GA, using a population size of 1,000, take only a few minutes for the smaller problems of up to 100 rectangles, but are unpredictable because of our stopping condition and can take several hours or even days for the larger problems.

Table 3: Comparison of GA with Results from FFDH (optimum height is 100)

Problem	GA: mean 5 runs	GA: best	FFDH: $W \geq H$
Nice.25	108	106	118
Nice.50	108	105	119
Nice.100	111	108	111
Nice.200	–	108	108
Nice.500	–	108	107
Nice.1000	–	–	105
Path.25	109	106	120
Path.50	108	106	136
Path.100	112	109	109
Path.200	123	112	116
Path.500	–	136	105
Path.1000	–	–	107

From Table 3 it is clear that the GA produces superior solutions for problems of about 100 rectangles or less, but for larger problems the FFDH heuristic outperforms the GA.

References

[1] E.G. Coffman Jr., M.R. Garey, and D.S. Johnson. Approximation Algorithms for Bin Packing – An Updated Survey. In G. Ausiello, N. Lucertini, and P. Serafini, editors, *Algorithm Design for Computer Systems Design*, pages 49–106. Springer-Verlag, Vienna, 1984.

[2] E.G. Coffman Jr., M.R. Garey, D.S. Johnson, and R.E. Tarjan. Performance bounds for level-oriented two-dimensional packing a lgorithms. *SIAM Journal of Computing*, 9:808–826, 1980.

[3] I. Golan. Performance Bounds for Orthogonal, Oriented Two-Dimensional Packing Algorithms. *SIAM J. Comput.*, 10(3):571–582, 1981.

[4] E. Hopper and B.C.H. Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, 128:34–57, 2001.

[5] S.M. Hwang, C.Y. Kao, and J.T. Horng. On solving rectangle bin packing problems using genetic algorithms. In *Proceedings of the 1994 IEEE International Conference on Systems, Man and Cybernetics*, pages 1583–1590, 1994.

[6] Stefan Jakobs. On Genetic Algorithms for the Packing of Polygons. *European Journal of Operational Research*, 88:165–181, 1996.

[7] Berhold Kröger. Guillotineable bin packing: A genetic approach. *European Journal of Operational Research*, 84:645–661, 1995.

[8] Dequan Liu and Hongfei Teng. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European Journal of Operational Research*, 112:413–420, 1999.

[9] C. L. Valenzuela and P.Y. Wang. VLSI Placement and Area Optimization Using a Genetic Algorithm to Breed Normalized Postfix Expressions. Under review.

[10] C. L. Valenzuela and P.Y. Wang. A Genetic Algorithm for VLSI Floorplanning. In *Parallel Problem Solving from Nature – PPSN VI*, Lecture Notes in Computer Science 1917, pages 671–680, 2000.

[11] P.Y. Wang and C.L. Valenzuela. Data set generation for rectangular placement problems. *European Journal for Operational Research*, 2001. To appear.