

New Heuristic and Interactive Approaches to 2D Rectangular Strip Packing

Neal Lesh, Joe Marks, Adam McMahon, Michael Mitzenmacher

TR2003-18 July 2003

Abstract

In this paper, we consider the two-dimensional rectangular strip packing problem. This problem appears unamenable to standard local search techniques, such as simulated annealing or genetic algorithms. A standard simple heuristic, Bottom-Left-Decreasing (BLD), has been shown to handily beat these more sophisticated search techniques.

We introduce and demonstrate the effectiveness of BLD*, a stochastic search variation of BLD. While BLD places the rectangles in decreasing order of height, width, area, and perimeter, BLD* successively tries random orderings, chosen from a distribution determined by their Kendall-tau distance from one of these fixed orderings. Our experiments on benchmark and randomly generated problems show that BLD* produces significantly better packings than BLD after only 1 minute of computation.

Furthermore, we observe that people seem able to reason about packing problems extremely well. We incorporate our new algorithms in an interactive system that combines the advantages of computer speed and human reasoning. Using the interactive system, we are able to quickly produce significantly better solutions than previous methods on benchmark problems.

To appear in IJCAI workshop on Stochastic Search Algorithms, 2003

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Research Laboratories, Inc.; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Research Laboratories, Inc. All rights reserved.

Submitted April 2003

New Heuristic and Interactive Approaches to 2D Rectangular Strip Packing

N. Lesh¹, J. Marks¹, A. McMahon², M. Mitzenmacher²,

¹ Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA, 02139
lesh@merl.com, marks@merl.com

² University of Miami, Florida.* adam@math.miami.edu

³ Harvard University, Computer Science Department
michaelm@eecs.harvard.edu †

Abstract

In this paper, we consider the two-dimensional rectangular strip packing problem. This problem appears unamenable to standard local search techniques, such as simulated annealing or genetic algorithms. A standard simple heuristic, Bottom-Left-Decreasing (BLD), has been shown to handily beat these more sophisticated search techniques.

In this paper, we introduce and demonstrate the effectiveness of BLD*, a stochastic search variation of BLD. While BLD places the rectangles in decreasing order of height, width, area, and perimeter, BLD* successively tries random orderings, chosen from a distribution determined by their Kendall-tau distance from one of these fixed orderings. Our experiments on benchmark and randomly generated problems show that BLD* produces significantly better packings than BLD after only 1 minute of computation.

Furthermore, we observe that people seem able to reason about packing problems extremely well. We incorporate our new algorithms in an interactive system that combines the advantages of computer speed and human reasoning. Using the interactive system, we are able to quickly produce significantly better solutions than previous methods on benchmark problems.

1 Introduction

Packing problems involve constructing an arrangement of items that minimizes the total space required by the arrangement. In this paper, we specifically consider the two-dimensional (2D) rectangular strip packing problem. The input is a list of n rectangles with their dimensions and a target width W . The goal is to pack the rectangles without overlap into a single rectangle of width W and minimum height H . We further restrict ourselves to the orthogonal variation, where rectangles must be placed parallel to the horizontal and vertical axes. We consider two variations: *fixed orientation* in which the rectangles cannot be rotated, and *variable orientation* in which they can be rotated by 90 degrees. Further, for all our test cases, all dimensions are integers. Like most packing problems, 2D rectangular strip packing (even with these restrictions) is NP-hard.

*This work done while visiting Mitsubishi Electric Research Laboratories.

†Supported in part by NSF CAREER Grant CCR-9983832 and an Alfred P. Sloan Research Fellowship. This work was done while visiting Mitsubishi Electric Research Laboratories.

A common method for packing rectangles is to take an ordered list of rectangles and greedily place them one by one. Perhaps the best studied and most effective such heuristic for the fixed orientation variation is the Bottom-Left (BL) heuristic, where rectangles are sequentially placed first as close to the bottom and then as far to the left as they can fit. For some problems, BL cannot find the optimal packings [2, 4], nor does it perform well in practice when applied to random orderings. However, a very successful approach is to apply BL to the rectangles ordered by decreasing height, width, perimeter, and area and return the best of the four packings that result [9]. We refer to this scheme as Bottom-Left-Decreasing (BLD).

A natural alternative approach would be to find good orderings of the rectangles for BL or other similar heuristics, using standard search techniques such as simulated annealing, genetic algorithms, or tabu search. Despite significant efforts in this area, the search space has not proven amenable to such search techniques; for more details see the thesis of Hopper [9].

In this paper, we present a variation of the BLD heuristic called BLD* that considers successive random perturbations of the original four decreasing orderings. We also present an apparently novel generalization of BL, and consequently of BLD and BLD*, for the variable orientation case. Our experiments on both benchmark and randomly generated problems show that BLD* substantially outperforms BLD as well as applying BL to randomly chosen orderings. For example, for the benchmarks taken from Hopper [9] in the case of fixed orientation, BLD* reduces the packing height from an average of 9.4% over optimal by BLD to about 5.24% over optimal after just one minute. We note that improvements of even 1% can be very valuable for industrial applications of this problem, such as glass and steel cutting.

This work was done as part of Human-Guided Search (HuGS) project, an ongoing effort to develop interactive optimization systems. The most successful results have been obtained by allowing users to guide stochastic search algorithms, in particular tabu search [13]. Determining how people can effectively interact with powerful stochastic search algorithms is important because it leverages people's abilities in areas in which they currently outperform computers, such as visual perception and strategic assessment. Furthermore, involving people in the process of optimization can help them understand and trust the produced solutions, as

well as modify them on the fly if the need arises.

For the 2D packing problem, we explored people’s ability to guide our BLD* heuristic. We found that people can reason about this problem extremely well. People can identify particularly well-packed subregions of a given packing and then focus a search algorithm on improving the other parts. People can also devise multi-step repairs to a packing problem to reduce unused space, often producing packings that could not be found by the BL heuristic for any ordering of rectangles. Our experiments on large benchmarks show that interactive use of BLD* can produce solutions 1% closer to optimal in about 15 to 25 minutes than BLD* produces on its own in 2 hours. Thus, 2D packing seems to be a problem for which people and computers can produce better results together than either can alone.

2 Background

Packing problems in general are important in manufacturing settings; for example, one might need n specific rectangular pieces of glass to put together a certain piece of furniture, and the goal is to cut those pieces from the minimum height fixed-width piece of glass. The more general version of the problem allows for irregular shapes, which is required for certain manufacturing problems such as clothing production. However, the rectangular case has many industrial applications [9].

The 2D rectangular strip packing problem has been the subject of a great deal of research, both by the theory community and the operations-research community [6, 8]. One focus has been on approximation algorithms. The Bottom-Left heuristic has been shown to be a 3-approximation when the rectangles are sorted by decreasing width (but the heuristic is not competitive when sorted by decreasing height) [2]. Other early results include algorithms that give an asymptotic $5/4$ -approximation [3] and an absolute $5/2$ -approximation [19]. Recently, Kenyon and Remilia have developed a fully polynomial approximation scheme [12].

Another focus has been on heuristics that lead to good solutions in practice. There are two main lines of research in this area. One line considers simple heuristics such as BLD. Another line focuses on local search methods that take substantially more time but have the potential for better solutions: genetic algorithms, tabu search, hill-climbing, and simulated annealing. The recent thesis of Hopper provides substantial detail of the work in this area [9, 10].

The fixed orientation problem has received much more attention than the variable orientation problem, although some genetic algorithm approaches have allowed reorientation as one of the mutation operations (e.g., [11, 7]). We are unaware of any previous work on adapting the BL algorithm for the variable orientation (as we describe below).

2.1 The Bottom-Left Heuristic

The *Bottom-Left* (BL) heuristic, introduced in [2], is perhaps the most widely used heuristic for placing rectangles. We think of the points in the strip to be packed as being ordered lexicographically, so that point A lies before point B if A is below B or, if A and B have the same height and A is to the left of B . Given a permutation of the rectangles, the Bottom-Left heuristic places the rectangles one by one, with

the lower left corner of each being placed at the first point in the lexicographic ordering where it will fit. There are natural worst-case $O(n^3)$ algorithms for the problem; Chazelle devised an algorithm that requires $O(n^2)$ time and $O(n)$ space in the worst case [5]. In practice the algorithm runs much more quickly, since a rectangle can usually be placed in one of the first open spots available. When all rectangle dimensions are integers, this can be efficiently exploited. Hopper discusses efficient implementations of this heuristic in her thesis work [9].

Perhaps the most natural permutation to choose for the Bottom-Left heuristic is to order the rectangles by decreasing height. This ensures that at the end of the process rectangles of small height, which therefore affect the upper boundary less, are being placed. It has long been known that this heuristic performs very well in practice [6]. It is also natural to try sorting by decreasing width, area, and perimeter, and take the best of the four solutions; while usually decreasing height is best, in some instances these other heuristics perform better. We refer to this as BLD.

2.2 Benchmarks

In this paper, we evaluate our algorithm and interactive system on both a set of structured benchmarks with known optimal packings and on randomly generated test instances without known optimal packings. The former is a set of benchmarks recently developed by Hopper. All instances in this benchmark have perfect packings of dimension 200 by 200. The instances are derived by recursively splitting the initial large rectangle randomly into smaller rectangles; for more details, see [9]. This benchmark set contains problems with size ranging from 17 to 197 rectangles. We use the non-guillotinable instances from this set, collections N1 (17 rectangles) through N7 (197 rectangles), each containing 5 problem instances.

The strengths of this benchmark are that a wide range of algorithms have been tested against it, providing meaningful comparisons; problem sizes vary from the small to the very large; and the optimal solution is known by construction. Furthermore, we were unable to find any other substantial set of benchmarks in the literature. The benchmark problems, however, are highly structured, and because all instances have perfect packings, they yield limited insight on the performance of algorithms when perfect packings are not available. We note that we have developed an exhaustive branch-and-bound algorithm which can quickly solve the N1-N3 problem instances [15]; therefore we tend to focus on the N4-N7 collections to evaluate our heuristic methods.

We also generated eight classes of random test instances that vary in size, *skew*, and target width of the rectangles (two variations for each). For all cases, the dimensions of the rectangles varied from 1 to 50. Each test instance consisted of either 50 or 100 rectangles. We compute the rectangles in one of two ways. Our first method is to randomly choose each dimension uniformly from 1 to 50. Our second method *skews* the rectangles by choosing the width x uniformly at random from 1 to 50. The length is determined by choosing y uniformly at random from 1 to 50 and fixing the length to be either y or $50 - y$, whichever is further from x . We refer to the sum of the area of all the rectangle as the *total area*

of the problem instance. We compute the target width given the target *ratio* of a rectangle that can hold the total area: the target width equals

$$\left\lceil \sqrt{\frac{\text{total area}}{\text{ratio}}} \right\rceil.$$

We generated instances with ratio 1 and 2. For each class, we randomly generated 10 problem instances.

Since we do not know the optimal answer for the randomly generated benchmarks, we evaluate a packing with a given height in terms of its percentage *over ideal*, where the ideal is the nearest integer rounding up from the total area divided by the target width.

3 Orienting rectangles

We modified the BL and BLD heuristics for the variable orientation problem.

Our variation of BL considers both orientations when placing each rectangle. For each orientation, it determines the first point in the bottom-left lexicographic ordering where the rectangle will fit. We experimented with three orientation-decision methods: compare where the bottom-left corner of the rectangle is positioned in each orientation, compare the center of the rectangle, or compare the top-right corner of the rectangle. In each case, we choose the orientation that places the relevant point at the position that is earlier in the bottom-left lexicographic ordering. In the case of ties (which turn out to be very rare), we choose randomly between the two orientations.

Because the rectangles can be reoriented, it does not make sense to order them by decreasing width or height. Instead, we order the rectangles by either their minimum or maximum dimension.

We ran experiments to evaluate the possible combinations of orientation-decision methods and orderings of the rectangles on the 20 instances in the N4 to N7 collections. If more than one ordering is used, then we take the best packing produced. Table 1 shows the average percent over optimal from the various combinations. The results indicate that the most effective orientation-decision method is to choose the orientation that places the top-right corner as early as possible in the lexicographic ordering. The most effective ordering is to sort the rectangles by their minimum dimension.

These results initially seemed counterintuitive. Our current understanding of why sorting by minimum dimension is better than by maximum dimension when the rectangles are reorientable is best expressed by an example: a 50×1 rectangle can be oriented so as to only add at most 1 to the height, and so it is reasonable to place this rectangle towards the end. Similarly, using the top-right corner to decide orientation most closely approximates the objective function being used to evaluate an entire packing.

4 Improving the BLD Heuristic

A natural way to improve the BLD heuristic is to apply BL to other permutation orders. At the expense of more time, more orders besides the four suggested can be tried to attempt to improve the best solution found. One standard technique

sort by				choose by		
min	max	area	perim.	center	bottom left	top right
yes	no	no	no	5.62	15.38	4.43
no	yes	no	no	5.58	6.40	5.98
no	no	yes	no	5.15	8.08	4.60
no	no	no	yes	5.23	5.58	4.70
no	no	yes	yes	5.00	6.28	4.40
no	yes	no	yes	4.83	5.30	4.70
no	yes	yes	no	4.85	5.82	4.53
yes	no	no	yes	4.68	5.58	4.33
yes	no	yes	no	4.85	8.08	4.43
yes	yes	no	no	4.73	6.47	4.38
yes	yes	yes	yes	4.60	6.08	4.23

Table 1: Results of BLD modified for variable orientation. Each number is an average of the 20 problem instances in the N4-N7 benchmark collections.

would be *random-repeat*: permutations are repeatedly chosen uniformly at random, and the best solution found within the desired time bound is used. Random permutations, however, are known to perform poorly [9]. We tried BL on random permutations on the N4 through N7 benchmark collections. After 20 minutes, the average height of the best solution found was 9.6% over the optimal compared to the 6.4% over optimal generated by the BLD heuristic in a matter of seconds.

Instead we suggest the following stochastic variation of BLD, which we call BLD*. Our intuition for why BLD performs so much better than random BL is that the decreasing sorted orders save smaller rectangles for the end. Therefore, BLD* chooses random permutations that are “near” the decreasing sorted orders used by BLD as they will also have this property. There are many possible ways of doing this; indeed, there is a deep theory of distance metrics for rank orderings [16]. BLD* uses the following simple approach: start with a fixed order (say decreasing height), and generate random permutations from this order as follows. Items are selected in order one at a time. For each selection, BLD* goes down the list of previously unaccepted items in order, accepting each item with probability p , until an item is accepted. If the last item is reached and not selected, then we restart at the beginning of the list, again taking an item with probability p . After an item is accepted, the next item is selected, starting again from the beginning of the list. This approach generates permutations that are near decreasing sorted order, preserving the intuition behind the heuristic, while allowing a large number of variations to be tried.

The probability starting from some fixed ordering x of obtaining some other ordering y is proportional to $(1 - p)^{Ken(x,y)}$, where $Ken(x,y)$ is the Kendall-tau distance between the two permutations. This is also known as bubble-sort distance, because it counts the number of swaps bubble-sort would make transforming x to y .

BLD* first tries the four orders used by BLD and then permutes each of these orders in round-robin fashion.

problem	Fixed orientation BLD* score after						Variable orientation BLD* score after					
	0(BLD)	1 m.	2 m.	5 m.	10 m.	20 m.	0(BLD)	1 m.	2 m.	5 m.	10 m.	20 m.
Hopper N1	16.40	7.12	6.40	6.40	5.90	5.40	12.50	4.88	4.90	4.60	4.20	4.10
Hopper N2	12.20	6.70	6.40	6.30	5.50	5.40	9.70	5.00	4.80	4.20	4.10	3.90
Hopper N3	12.40	6.30	6.10	5.60	5.60	5.20	10.00	4.70	4.40	4.00	4.00	3.70
Hopper N4	9.00	5.50	5.30	5.10	4.60	4.30	6.10	3.40	3.20	3.00	2.90	2.80
Hopper N5	7.60	4.80	4.50	4.20	4.20	3.90	6.60	3.10	2.80	2.70	2.60	2.60
Hopper N6	5.40	4.30	4.10	4.00	3.80	3.70	3.60	2.60	2.40	2.30	2.10	1.90
Hopper N7	4.00	2.30	2.30	2.20	2.20	2.10	1.40	1.10	1.10	1.00	0.90	0.90
Hopper N1-N7	9.43	5.24	5.01	4.83	4.54	4.29	7.12	3.50	3.37	3.11	2.97	2.84
random,size=50,r=1	9.82	5.82	5.75	5.29	5.12	4.71	6.74	3.84	3.56	3.45	3.39	3.05
random,size=50,r=1,skew	14.20	8.40	8.25	7.85	7.58	6.99	9.31	5.46	5.17	4.90	4.69	4.36
random,size=50,r=2	8.35	5.49	5.18	4.86	4.66	4.58	6.58	3.83	3.65	3.50	3.30	3.14
random,size=50,r=2,skew	13.72	8.72	8.55	7.80	7.47	7.10	11.08	6.03	6.08	5.49	4.93	4.79
random,size=100,r=1	5.14	3.96	3.97	3.70	3.70	3.62	4.33	2.87	2.82	2.62	2.62	2.46
random,size=100,r=1,skew	10.76	7.60	7.47	7.12	6.98	6.93	7.31	5.29	5.18	5.01	4.79	4.52
random,size=100,r=2	6.02	4.89	4.67	4.59	4.43	4.26	5.70	3.87	3.42	3.25	3.16	3.00
random,size=100,r=2,skew	10.75	8.17	7.68	7.26	6.96	6.67	8.70	6.36	5.95	5.40	5.34	5.01
random, all	9.84	6.63	6.44	6.06	5.86	5.61	7.47	4.74	4.48	4.20	4.03	3.79

Table 2: Average results of BLD* on the Hopper benchmarks and on randomly generated problems.

4.1 Experimental Results

We ran BLD* on the Hopper benchmarks N1-N7 and on all our random data sets. We used $p = 0.5$ based on a small amount of preliminary investigation of different values. For the fixed orientation problem, we used all four orderings (height, width, area, and perimeter). For the variable orientation problem, we used our modified version of BLD with using only the minimum-dimension ordering and the top-right orientation-decision method.

The results are shown in Table 2. The table shows the results of running BLD and the results of running BLD* at various time increments. The numbers represent the percentage over ideal, as defined above. For all cases, BLD* dramatically improves solutions over BLD even with just one minute of computation. It continues to improve steadily, though improvements taper off with time.

We also ran experiments on the N4-N7 collections to measure how many permutations BLD* considered before improving upon the best solution by BLD; it considered an average of only 15.25 permutations to do so.

5 Interactive Packing

Human guidance has been shown to improve the performance of stochastic optimization algorithms for a variety of problems (e.g., [1, 13, 14] and the papers cited therein). In order for human interaction to be justified for an optimization problem, improvements in solution quality must have high enough value to warrant investing human effort. This is the case for packing problems in which manufacturing costs, and thus potential savings, are high. In order for interaction to be applicable to an optimization problem, there must exist effective visualizations for its problems and solutions. Fortunately, the obvious geometric visualization for packing problems (e.g., see Figure 1) is simple and effective.

In order for human interaction to be beneficial, human reasoning must offer some advantages over the best automatic methods. We have found that people can help overcome many of the limitations of the BLD* heuristic. Peo-

ple can identify particularly well-packed subregions of solutions, and focus BLD* on improving the other parts. Furthermore, people can readily envision multi-step repairs to a packing problem to reduce unused space. These repairs often involve producing solutions that could not be produced by the BLD heuristic.

5.1 Interactive System

We have developed an interactive rectangle-packing system in Java using the Human-Guided Search (HuGS) Toolkit [14]. The toolkit provides a conceptual framework for interactive optimization as well as software for interacting with a search algorithm, logging user behavior, providing history functions including undo and redo, file I/O, and some other GUI functions. We did not however utilize the human-guidable tabu or hill-climbing search algorithms provided in HuGS, as we did not find them effective for this problem in our initial explorations.

In our system, the user is always visualizing a current solution as shown in Figure 1. Given the aspect ratio of a computer monitor, we found it more natural to rotate the problem by 90 degrees, so that there is a fixed height and the goal is to minimize the width of the enclosing rectangle.

The user can manually adjust the current solution by dragging one or more rectangles to a new location. The interface contains buttons which allow the user to cause all the rectangles to be shifted downward or leftward. This basically has the effect of pulling all of the rectangles in one direction until each touches its neighbor or an edge of the possible packing area. These functions also resolve overlaps among rectangles. Additionally, the user can freeze particular rectangles. Frozen rectangles appear in red and will not be moved by the computer. Rectangles that are not frozen appear in green. For the variable orientation problem, the user has the option of reorienting rectangles, manually.

The user can also invoke, monitor, and halt the BLD* heuristic. The user specifies a target region in which to pack rectangles, denoted by a purple rectangular outline. The user can then invoke BLD* by pressing a Start button.

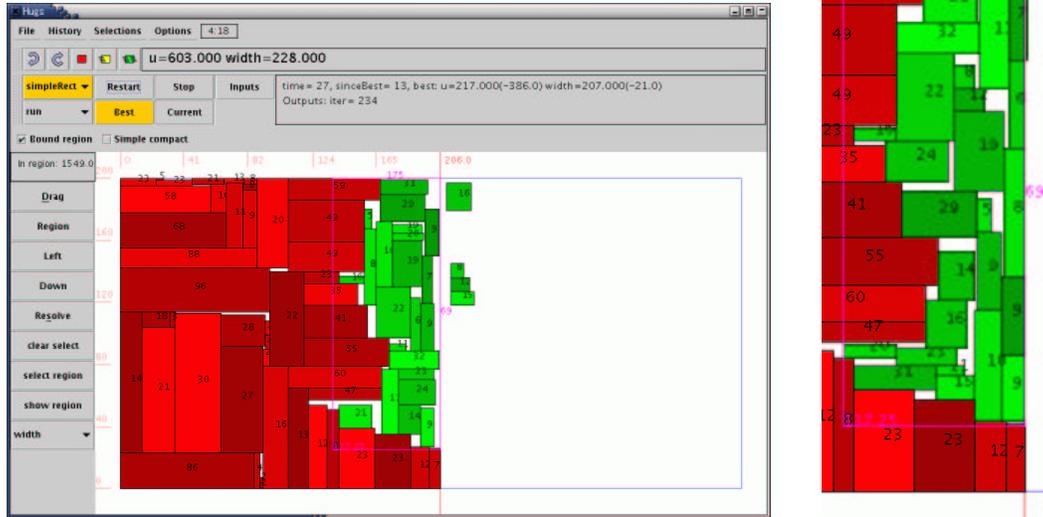


Figure 1: Interactive system: The image on the left is a screen shot of our system in use. The user has selected a region to apply BLD* to and has frozen most of the rectangles (frozen rectangles shown in red/dark gray, unfrozen in green/light gray). The image on the right shows a blowup of the selected portion of the packing, after BLD* has run for a few seconds and the user has pressed the Best button to see the best solution found.

Any frozen rectangles within the region are left where they are. BLD* then tries to fill the region using any rectangles that are not currently frozen. The system works in the background, and uses a text display to indicate the value of the best, i.e., most tightly packed, solution it has found so far. The user can retrieve this solution by pressing the Best button. The user can retrieve the current solution the engine is working on by pressing the Current button. The user can manually modify the currently visualized solution without disturbing the current search. When the search algorithm finds a new best solution, the Best button changes color to alert the user. The user can halt the search algorithm by pressing the Stop button, or reinvoke it by pressing the Start button again.

The user can optionally set a target for the solution she is trying to reach. For example, the user can indicate that the enclosing rectangle should be 200×204 . The system provides some visual cues for how to meet this goal. More importantly, the target solution size affects how solutions are ranked. Rather than using the true objective function (i.e., the size of the enclosing rectangle), the system ranks solutions based on the total area of the rectangles that fall within the target solution size. We found this feature to be extremely useful. For example, the user typically begins a session by having BLD* try to pack the entire target region. Because of our modification, the search algorithm might return, for example, a packing with one rectangle that sticks out of the target region by several units rather than a packing in which many rectangles stick out of the target region by one unit. We usually found the former packings much easier to repair.

5.2 Interaction Experiments

The primary goal of these experiments was to evaluate the hypothesis that interactive use of BLD* can produce superior solutions than BLD* can on its own.

We ran our first set of experiments on the fixed-orientation problem, using the 15 problem instances in the N4-N6 collections in the Hopper benchmark suite. We ran BLD* for 2 hours on each instance. We then performed one trial for each instance in which a user attempted to find a solution 1% closer to optimal than the best solution found by BLD* within 2 hours. The users were two authors of this paper. We were careful that a user had never before seen the particular instances on which they were tested. We logged the users' actions, but the primary measure was how long it took the user to reach their target.

As shown in Table 3, the users were able to reach these targets in about 15 minutes on average. In every case, the target was reached within 30 minutes. While this is not exactly a "head-to-head" comparison, since the users had the target scores to reach, the fact that people were able to improve on the solutions so quickly confirms our hypothesis. Note that because BLD* is a random restart strategy, it is unlikely that running it for another 2 hours would improve the solution much.

We then ran a second set of experiments for the variable-orientation problem. In these experiments, the users employed our variation of BLD* that orients the rectangle and could also manually orient them. As in the first experiments, we measured how long it took the users to find solution 1% closer to optimal than the best solution found by BLD* within 2 hours. We ran these experiments on the N4 and N5 collections, with the same test subjects as the first experiments. Thus, in this experiment, the test subjects had

dataset	number of rectangles	percent over optimal by BLD* in two hours	time for users to find packing 1% closer to optimal
N4	49	2.9%	1.9% in 26 min., 21 sec.
N5	73	2.6%	1.6% in 19 min., 59 sec.

Table 4: Interaction experiment results with variable orientation: The second column shows the average percentage over optimal achieved by BLD* in two hours. The third column shows the average time it took interactive use of BLD* to achieve a solution another 1% closer to optimal. The values are averaged over the 2 trials each of five problem instances in the corresponding collection.

7 Acknowledgments

We would like to thank Victor Milenkovic for useful discussions and for sending us an excellent summer intern!

References

- [1] D. Anderson, E. Anderson, N. Lesh, J. Marks, B. Mir-tich, D. Ratajczak, and K. Ryall, Human-guided simple search. In *Proceedings of AAAI 2000*, 209–216, 2000.
- [2] B. S. Baker, E. G. Coffman, Jr., and R. L. Rivest. Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9:846-855, 1980.
- [3] B. S. Baker, D. J. Brown, and H. P. Katseff. A $5/4$ algorithm for two-dimensional packing. *Journal of Algorithms*, 2:348-368, 1981.
- [4] D. J. Brown. An improved BL lower bound. *Information Processing Letters*, 11:37-39, 1980.
- [5] B. Chazelle. The Bottom-Left Bin-Packing Heuristic: An Efficient Implementation. *IEEE Transactions on Computers*, 32(8):697-707, 1983.
- [6] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin-packing : an updated survey. In: G. Ausiello, M. Lucertini, and P. Serafini, editors , *Algorithm Design for Computer Systems Design*, pages 49-106, Springer-Verlag, 1984.
- [7] Dagli, C. H. and Poshyanonda, P. New approaches to nesting rectangular patterns. *Journal of Intelligent Manufacturing* 8, pp. 177-190, 1997
- [8] H. Dyckhoff. Typology of cutting and packing problems. *European Journal of Operational Research*, 44, 145-159, 1990.
- [9] E. Hopper. Two-Dimensional Packing Utilising Evolutionary Algorithms and other Meta-Heuristic Methods, PhD Thesis, Cardiff University, UK. 2000.
- [10] 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(1):34-57,2000.
- [11] Hwang S.M., Cheng Y.K., and Horng J. T., On solving rectangle bin packing problems using genetic algorithms. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Part 2 (of 3), pp. 1583-1590.
- [12] C. Kenyon and E. Remilia. Approximate Strip-Packing. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 31-36, 1996.
- [13] G. Klau, N. Lesh, J. Marks, and M. Mitzenmacher. Human-Guided Tabu Search. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pp. 41-47, 2002.
- [14] G. Klau, N. Lesh, J. Marks, M. Mitzenmacher, and G.T. Schafer. The HuGS platform: A toolkit for interactive optimization. In *Proceedings of Advanced Visual Interfaces*, pp. 324-330, 2002.
- [15] N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher. New Exhaustive, Heuristic, and Interactive Approaches to 2D Rectangular Strip Packing MERL Technical Report TR2003-05, 2003.
- [16] J. I. Marden. *Analyzing and Modeling Rank Data*, Chapman & Hall, New York, New York, 1995.
- [17] K. Ratanapan and C. H. Dagli. An object-based evolutionary algorithm for solving rectangular piece nesting problems. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 989-994, 1997.
- [18] K. Ratanapan and C. H. Dagli. An object-based evolutionary algorithm: the nesting solution. In *Proceedings of the International Conference on Evolutionary Computation*, pp. 581-586, 1998.
- [19] D. Sleator. A 2.5 times optimal algorithm for packing in two dimensions. *Information Processing Letters*, 10:37-40, 1980.