

Clayton School of IT

CSE460 Optimization and Constraints

Assignment 2

Due Friday 6th of October COB

The purpose of this assignment is to practice the application of (stochastic) meta-heuristics to simple problems.

The so-called cutting stock problem is an industrial variant of the two-dimensional bin packing problem: A given set of shapes of fixed sizes have to be cut from a larger sheet of material. The order in which these shapes are produced does not matter and hence the production process is free to allocate them in any way on the (two-dimensional) sheet of material. The objective of the allocation is to minimize the waste of material when the shapes are cut. In other words, we are looking to find the smallest enclosing rectangle in which all given shapes can be packed such that there is minimum unused space (wasted material) between them.

Often the cutting machinery is set up such that all cuts must be *guillotineable*. This means that each cut must be executed in a straight line through the whole sheet, separating it into two smaller sheets. Further guillotineable cuts are then applied to these pieces recursively (See Figure 1).

For our particular instance of the cutting stock problem we make the following simplifying assumptions:

- all items (shapes) are rectangular,
- items can be rotated by 90 degrees when placed on the sheet,
- all cuts must be guillotineable and axis-parallel,
- the sheet of material is always big enough,
- there are no further constraints on the problem,
- the shape of material that is left from the sheet after cutting all items is not taken into consideration.

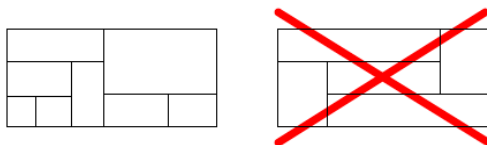


Figure 1: A guillotineable and a non-guillotineable cut. Each of the boxes contains one item. Notice that each of the boxes also contains blank space (wasted material) unless the neighboring items have the same extension in one dimension.

Your task is to design and implement an algorithm that can compute the sequence of cuts automatically.

Warning: It is obviously possible to spend an arbitrary amount of time on this task, trying to achieve world-class performance. This is not the purpose of the assignment. It is allocated to require less than four full days. Please be aware that your solution will be judged mainly by its approach and not by the absolute quality of the results achieved. This means that your submission must show evidence of a well-considered design of the algorithm and proper testing and evaluation of the approach. It is admissible to use information obtained from other sources, e.g. on the internet, to solve this problem, but you must develop your own solution and *all sources used must be properly attributed*. Copying a solution from external material

and not attributing it is an instance of cheating and will lead to your submission being disallowed, i.e. zero marks.

1. Select a suitable optimization method (meta-heuristic) to tackle this problem and write a program that given a set of shapes computes the complete cutting sequence. You are free to use your favourite programming language for this task. The input to your program are the shape descriptions, which are given as plain text in the following form: “ $n, w_1, h_1, w_2, h_2, \dots, w_n, h_n$ ”, where n is the number of shapes in the set, w_i is the width of the i -th item and h_i its height. All numbers are integers. The numbers are separated by a comma and the usual whitespace characters (spaces, tabs, newlines, linebreaks, etc.). You can read this data from a file or from the console input stream. The output of your program should be a suitable description of the cutting sequence, the dimensions of the outermost bounding box and the amount of waste generated by the cutting sequence in this box. **[25 marks]**
2. Extend your program such that it gives a graphical display of your solution, indicating items and cuts (5 marks). It should also give an adequate indication of the progress of the optimization. For example, it could display the best solution found so far during the search and the current best objective value. (2 marks). **[7 marks]**

Hint: If you are not sure how to program graphical output, one of the simplest ways possible is to use TK. The following, rather self-documenting program should illustrate sufficiently how you can solve this task easily without becoming entangled in GUI details. Just replicate the commands in the last two lines to create more items.

```
#include <stdio.h>

/* called with ./a.out | wish
*/

int main() {
    printf("puts stdout {Hello World!}\n");
    printf("frame .menubar \n");
    printf("button .menubar.hello -text Hello -command {puts stdout {Goodbye!} \n exit}\n");
    printf("button .menubar.clear -text Clear -command {.can delete cuts items}\n");
    printf("canvas .can -width 100 -height 100 \n");
    printf("pack .menubar.hello -side right -padx 20 -pady 10 \n");
    printf("pack .menubar.clear -side left -padx 20 -pady 10 \n");
    printf("pack .menubar .can -side top \n");
    printf("eval {.can create rect} {50 50 70 70} {-outline red -fill green -width 3 -tags items} \n");
    printf("eval {.can create line} {0 0 50 50} {-fill blue -width 4 -tags cuts} \n");

    return 0;
}
```

3. Describe the data structure that you are using to represent the cut sequence and the relevant parts of your optimization process. (For instance, if you are using a GA describe genome structure, fitness function, selection function, mutation, cross-over; if you are using SA describe the instance representation, the evaluation function; the cooling schedule and the neighborhood generation function; if you are using ACO pay particular attention to the pheromone model. Document repair functions and local search methods if you are using any. **[8 marks]**

Hint: One possible representation is an operator tree in which the nodes are items and the operators are rotation, and horizontal and vertical composition. A postfix expression such as

$$item_1 \ item_2 \ h \ item_3 \ rot \ v \dots$$

can be used as a convenient representation of this.

4. Evaluate your approach briefly. Design and describe some sets of test data and the results for these. What can you learn from this about the performance of your approach? Do your results suggest any modifications of your algorithm that should be tried? Given that you can only spend a very limited amount of time on this task, your evaluation will have to remain sketchy. Suggest how you would extend your evaluation if you had more time to perform it. [20 marks]

Submission Instructions

The above exercises contribute 60% to your total CSE460 mark.

Submission will be electronic. Submit a single *.tar or *.zip or *.gz file which contains your (commented!) program code in separate plain text files. Additional files for documentation can be submitted where required as plain text, postscript or portable document format (pdf). Send your submission as an attachment by electronic mail to `bernd.meyer@infotech.monash.edu.au`. **Your programs must run on the honspc machines or on fangorn. If your programs do not run on either of these you must make arrangements with me for you to demonstrate your programs prior to submission.** Clearly indicate in the email accompanying your submission on which machine your programs have been tested.

The assignment is due **Friday 6th of October COB**.

Assignments handed in after the due date will attract a late penalty of 10% per day unless special consideration applies or there has been prior agreement in writing from the lecturer. No submission will be accepted later than one week after the due date.