A 2D Strip Packing Problem with One Fixed Variable

A Stockyard Management Problem

Mitchell Metcalfe
mitchell.metcalfe@uon.edu.au

University of Newcastle

March 5, 2012

Summary of a project undertaken as part of an AMSI 2011/2012 Summer Vacation Scholarship, supported by the CSIRO

Abstract

Given a 2D strip of fixed height and a set of rectangles with predetermined widths, heights, and x-positions, a valid packing of the rectangles corresponds to an assignment of a y-position to each rectangle such that all rectangles are on the strip and none overlap. This project investigated conditions determining the existence of such a packing, and methods of finding one. The goal was to prove a conjecture that suggested that a valid packing of rectangles exists as long as a condition on the heights of overlapping rectangles is met. A counterexample to the conjecture was eventually found, after which, methods of conflict minimisation for infeasible problem instances were considered.

1 Introduction

1.1 Stockyard-scheduling

The problem explored in this project is inspired by a problem related to stockyard scheduling. Coal, once delivered to a coal port by train, must be stockpiled on a coal pad for a period of time before being loaded onto ships. The coal pad is a narrow strip of finite length, and thus may only hold a limited amount of coal at any given instant. Each stockpile is allocated some length of space on the pad before it is stored there, taking up the full width and allocated length of the pad until it is loaded onto a ship and the pad space is free again. For our problem, we assume that we know in advance the full schedule of shipments that must occur up until some time horizon. Our task is then to determine where to position each stockpile on the pad such that all stockpiles will fit, and all shipments can be made.

More formally, consider a 2D strip (in a time-space coordinate system) of given height H, where H is the pad length, and of width W equal to the time horizon. A stockpile of size h, which is to be placed on the pad at time x, y units from one end, and is to remain there for w units of time corresponds to a rectangle (x, y, w, h) in time-space coordinates. Thus, our problem can be stated as: Assign a y position to each of n rectangles of predetermined width, height, and x-postion, such that no two rectangles overlap, and all rectangles are contained within the larger rectangle $(0, 0, W, H)$.

1.2 Strip-packing

The problem of placing a given set of axis-aligned rectangles onto a strip, such that the rectangles don't overlap is an example of an *oriented strip-packing problem*. That is, given

a list R of n *oriented* rectangles (i.e. rectangles whose edges are parallel to the coordinate axes, and that can't be rotated) each with a given width w_i and height h_i, assign each rectangle a position (x_i, y_i) (where $x_i, y_i \geq 0$), such that $y_i + h_i \leq H$ for all rectangles i (where H is a given 'strip height'), such that the rectangles do not overlap, and such that $max(x_1, x_2, ..., x_{n-1}, x_n)$ is minimised.

Our problem is similar, except that each x_i is predetermined, and is given along with the widths and heights as part of the problem's input (meaning that we only require a feasible solution, rather than a solution with that minimises the maximum x_i).

1.3 Height-Sum Conjecture

Clearly, for the problem to have a solution, the sum of heights of all blocks that overlap any given x-position must be $\leq H$. The conjecture this project set out to prove was that if a problem instance satisfies this condition, then that problem instance has a solution. I.e. a feasible arrangement of blocks exists if and only if the sum of block heights at every x-position is $\leq H$.

2 Methods

2.1 Integer Program

Since the x-positions and widths of blocks are fixed, if the x-spans of two given blocks do not overlap, then there exists no assignment of y-positions such that the two blocks intersect each other. We consider blocks arranged as such to be *independent*.

Blocks whose x-spans do overlap could intersect each other depending on the choice of y-coordinates. We consider these blocks to be *dependent*.

The following is a formulation of the problem as an integer program. This program was used to solve problem instances for the purpose of visualisation.

The resulting integer program is as follows:

$$
\begin{aligned}
& H = \text{the strip height} \\
& \text{for all blocks } i: && \text{(strip constraints)} \\
& \quad 0 \leq y_i \leq H - h_i \\
& \text{for all dependent blocks } i, j: && \text{(intersection constraints)} \\
& \quad y_i + h_i \leq y_j + b_{ij} H \\
& \quad y_j + h_j \leq y_i + (1 - b_{ij}) H
\end{aligned}
$$

2.2 Random Instance Generator

A C++ program was written that could generate a random problem instance, solve it using Gurobi Optimiser (Gurobi Optimization, 2012), and generate diagrams of the unsolved and solved arrangements of blocks as Postscript. The data generator was written to aid in the visualisation of problem instances.

The data generator creates an integer program of the form described in §2.1 from a generated problem instance, and Gurobi Optimiser is called to solve it. If the problem instance is found to be feasible, the y-positions from the solution are used to generate a diagram of the feasible arrangement.

2.3 Simplification

In an effort to prove the height-sum conjecture, a number of ways of modifying problem instances were identified. Each modification would change the problem instance in such a way that if the unmodified problem instance was feasible, then its modified form would also be feasible, and that a solution to the modified instance can easily be converted into a solution to the original problem instance.

An example of one such problem simplification is the use of the maximal cliques of the interval graph corresponding to a given problem instance to re-scale the x-axis of a problem instance to a minimal, integer form.

2.4 Reduced problem instance

By repeatedly applying the simplifications in §2.3 in the correct order until no more can be applied, a problem instance can be converted into what we call a *reduced form*.

Any solution to a problem instance's reduced form can be converted into a solution to the original problem instance. Also, if the original problem instance is feasible, and the height-sum conjecture is true, then the problem instance's reduced form must also be feasible.

2.5 Limited Cases

At first, only restricted cases of the problem were considered. These included:

1. Reduced problems in which each block spans two or fewer maximal cliques.
2. Reduced problems in which the height of each block is no greater than two.
3. Reduced problems in which each block spans three or fewer maximal cliques.

The first of these was fairly trivial, and it was proved that all problems of that form were feasible. However, the second and third of these cases resisted proof.

2.6 Counter Example

While working on limited cases of reduced problems, and after modifying the data generator to generate reduced problem instances with integer block widths and block heights, a number of counterexamples to the height-sum conjecture were discovered.

Figure 1 illustrates a simple counterexample. The height sum in each maximal clique is equal to H, but no feasible arrangement of blocks exists.

2.6.1 Implications

The existence of a counterexample to the height-sum conjecture proves the conjecture to be false. Additionally, it implies that most of the problem simplifications in §2.3 are invalid (i.e. those based on the height-sum conjecture).

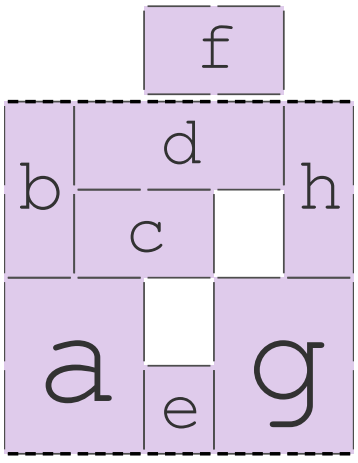

Figure 1: A counterexample to the height-sum conjecture.

2.7 Conflict Minimization

The existence of a counterexample to the height-sum conjecture demonstrates that not all problem instances being considered have a feasible solution. However, finding a way of 'solving' such problems in some sense could have practical benefit.

There are two main senses in which an infeasible problem instance can be solved:

- Blocks may be allowed to overlap, but the 'amount' of overlapping is minimised.
- Some aspect of the problem is changed in a minimal way such that the blocks can be placed without overlapping.

A number of integer programs were written that could solve infeasible problem instances by minimally changing them in a variety of different ways.

3 Future Work

3.1 Feasibility Condition

The height-sum condition considered here fails to indicate whether a given problem instance is feasible, but some other simple test may work. Finding an easy (i.e. polynomial time) test for feasibility would have practical benefit, and is a goal worth pursuing.

3.2 Exact solutions

The integer programs presented in §2.1 and §2.7 provide exact solutions to all feasible problem instances, and additionally, the programs in §2.7 provide block arrangements that cause 'minimal conflict' in some sense. Other methods of finding exact solutions would be worth exploring, in particular, methods that might have a faster running time than the integer programs used here.

4 Acknowledgements

I would like to thank my supervisors, Prof. Natashia Boland, Dr Faramroze Engineer, and Conjoint Prof. Martin Savelsbergh for their guidance and support.

I would also like to thank AMSI for providing me with this opportunity, and the CSIRO for their generous support. Undertaking an AMSI Vacation Research Schoplarship gave me a valuable first taste of what research mathematics is like. I found the opportunity of attending CSIRO's Big Day In conference both enjoyable and exciting. During the conference I was able to meet and talk with a number of students with a wide range of interests and backgrounds. I would reccomend the experience to any interested student of mathematics.

5 Bibliography

Gurobi Optimization, 2012. *Gurobi Optimizer 4.6.* [online] Available at: <http://www.gurobi.com> [Accessed 11 March 2012].