# Packing Boxes with Score Constraints

Asyl L. Hawa

September 3, 2017

## 1 Example

### 1.1 Creating problem instance

In this example, we will attempt to find a feasible alignment of nine boxes, with score widths ranging between 1mm and 70mm. The minimum score separation constraint, or "threshold", will be set at 70mm, the industry standard.

Initially, we create 18 random values between 1 and 70, which will be our score widths. We also need to add an extra box with two score of widths 70mm, which will be our "dominating scores", giving us a total of $n = 10$ boxes and 20 scores. These dominating scores will eventually be discarded.

We then sort the scores in non-decreasing order, and place them in a vector. The scores will now be addressed by their indices throughout the rest of the algorithm (see Table 1).

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Scores | 3 | 4 | 12 | 22 | 25 | 35 | 36 | 37 | 38 | 45 | 49 | 54 | 54 | 55 | 55 | 64 | 65 | 66 | 70 | 70 |

Table 1: allScores vector.

Since every box consists of two scores, each score is randomly assigned a "mate", so that one score (the smaller score) represents the left-hand side of the box, and the other score represents the right-hand side of the box. The only exceptions to the random allocation of mates are the dominating scores - they must be assigned to one another. We can then assign measurements to each pair of mates to represent the width of each individual box.

| Mates | | Box Width |
|---|---|---|
| 0 | 6 | 314 |
| 1 | 10 | 372 |
| 2 | 16 | 297 |
| 3 | 11 | 220 |
| 4 | 17 | 959 |
| 5 | 9 | 738 |
| 7 | 12 | 622 |
| 8 | 15 | 635 |
| 13 | 14 | 859 |
| 18 | 19 | 0 |

Table 2: Mates.

Next, we create an adjaceny matrix. This adjaceny matrix contains information regarding threshold constraint. If the sum of two scores is greater than or equal to the threshold, the adjaceny matrix contains "1" in the relevant cell. If two scores are mates, the adjaceny matrix contains "2". Else, the adjaceny matrix contains "0".

### 1.2 Matching scores

Once we have created the problem instance, we then have to attempt to match the scores together. If two scores can be matched, it means that the boxes can be placed next to each other, and the total width of the two scores from the boxes is equal to or exceeds the threshold, and can therefore be scored using the knives in the scoring machine.

The matching is performed as follows: starting from the smallest score (0), we attempt to match it with the largest possible score it is adjacent to ( = 1 in the adjaceny matrix), and that has not yet been matched (in this case, 19). We continue until all scores have been matched. If a score cannot be matched due to the largest score being its mate, we match the score with the second largest score available. As we can see in Table 3, score 7 was unable to be matched with the largest possible score, 12, as they are mates. Instead, 7 was matched with the next largest score, 11.

Once the matching algorithm is complete, we then examine the size of the matching list. For the instance to be feasible, all of the scores must be matched. If not all of the scores are matched, i.e. there are less than $n$ matchings in the matching list, then there are not enough matching edges to create a path, and the instance is infeasible.

| Match | |
|---|---|
| 0 | 19 |
| 1 | 18 |
| 2 | 17 |
| 3 | 16 |
| 4 | 15 |
| 5 | 14 |
| 6 | 13 |
| 7 | 11 |
| 8 | 12 |
| 9 | 10 |

Table 3: Match List.

### 1.3 Mate-Induced Structure

Now that we have a list of matched scores, we can now attempt to create an *alternating cycle* which consists of an edge between scores that are mates, followed by an edge between scores that are matched. Therefore each score will either be preceded by a mate and followed by a match, or vice versa. The cycles are easily formed using the mate list and match list.

If the mate-induced structure consists of only one cycle, then the instance is instantly feasible. In this example we have two cycles, and so we must find a way to connect, or "patch", these two cycles together to create a single cycle.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Cycle 1 | 0 | 6 | 13 | 14 | 15 | 9 | 10 | 1 | 18 | 19 |
| Cycle 2 | 2 | 16 | 3 | 11 | 7 | 12 | 8 | 15 | 4 | 17 |

Table 4: Mate-induced structure consisting of two cycles.

### 1.4 FCA

In order to join these two cycles together to create a single cycle, we must find scores from different cycles that are able to be matched, i.e the sum of the scores are greater than or equal to the threshold. If we are able to find such scores, then we simply remove the old matches, and add the new ones. In

this case, there are three possible sets of scores that can be used to connect the cycles together. For simplicity and speed, we choose the first set of scores found. This is depicted in FIGURE.

| | Scores |
|---|---|
| Cycle 1 | 1 - 18 - 2 - 17 |
| Cycle 2 | 4 - 15 - 5 - 14 |
| Cycle 3 | 6 - 13 - 8 - 12 |

Table 5: T-Matrix.

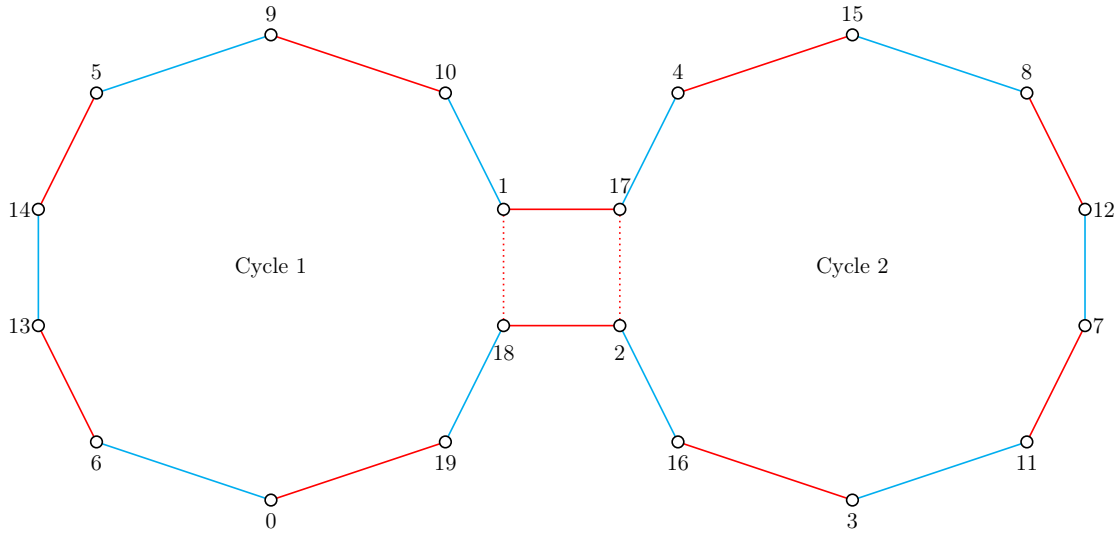


Figure 1: Patching MIS blue = mates, red = matching, explain dotted lines.

Now that the cycle has been formed, we simply remove the dominating scores, and obtain a single path.

```
Complete Path:  0 6 13 14 5 9 10 1 17 4 15 8 12 7 11 3 16 2
Order of Boxes:  1 9 6 -2 -5 -8 -7 -4 -3
Total Length of Path:  5043 millimeters.
```

## 2 Improvements

- Becker used arrays and set a specific size for each one - wastes memory and is inefficient. Instead I used vectors that could be expanded as required.
- Becker's code only evaluates whether a specific instance is feasible or infeasible. My code now outputs a solution to an instance in full, including the following:
  - Outputs final solution with score indices in order.
  - Outputs final solution with score widths in millimeters in order.
  - Outputs box widths in millimeters in order.
  - Outputs box number in order: if the number is positive, then the box's orientation is "regular" (the smallest score width is on the left-hand side), and if the number is negative, then the box's orientation is "rotated" (the smallest score is on the right-hand side).
  - Outputs the total length of the boxes in millimeters.

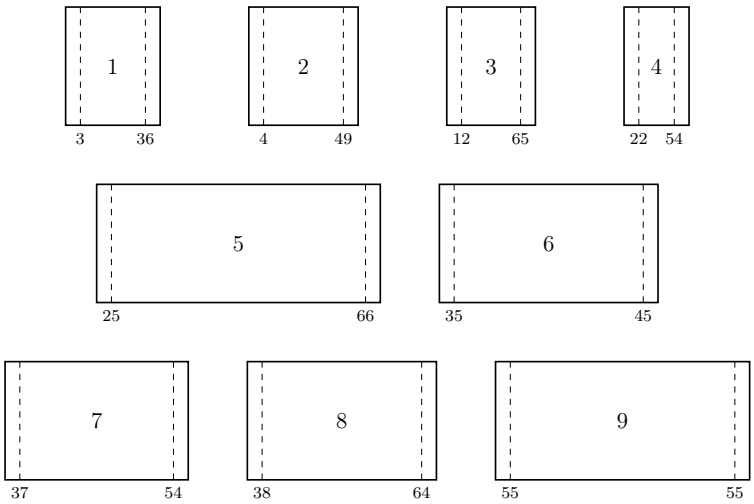


Figure 2: Individual boxes.

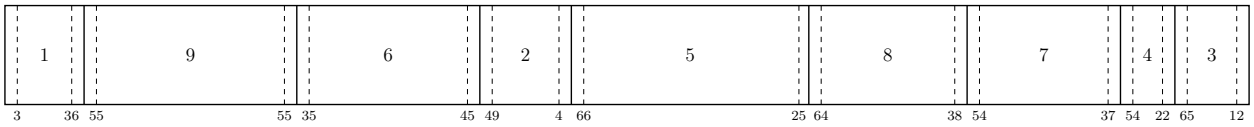


Figure 3: Final feasible alignment of boxes.

- All methods in the algorithm are now in separate functions, which makes the code easier to read and edit.
- The code determines whether only one T-cycle is required to connect all the cycles together, or whether multiple T-cycles are required.
- Box widths have now been added - each box has a width in millimeters.
- User input - users can create a simple text file containing score width, mates and box widths to use with the program.

# 3 Things to find out/do

- What are the industry standard strip lengths?
- Goulimis $n = 10$, is this true? Are there cases where more boxes are needed?
- Current speed of scoring in industry
- What happens before/after scoring in industry?
- Output to csv file?
- Calculate complexity
- Martello and Toth (1990b) MTP algorithm, adapt with additional constraints
- Exemptions on threshold, what if the instance in infeasible due to one box, can that box be held back and attempt to use in the next instance? Or can it still be used, but scoring run slower?

- How many scoring knives are used?
  - If only one set of scoring knives is used, then are the boxes placed on a conveyor belt? Does this affect the time?
  - If there are a specific number of scoring knives used, should we look at how many boxes are on each strip? I.e. if there are 4 scoring knives, then there should be only 4 scores on the strip, even if there is space available left on the strip?