Present Wrapping problem

Leonardo Calbi - leonardo.calbi@studio.unibo.it
Alessio Falai - alessio.falai@studio.unibo.it

September 12, 2020

Contents

Foreword

The problem is presented as: given a wrapping paper roll of a certain dimension and a list of presents, decide how to cut off pieces of paper so that all the presents can be wrapped.

Consider that each present is described by the dimensions of the piece of paper needed to wrap it. Moreover, each necessary piece of paper cannot be rotated when cutting off, to respect the direction of the patterns in the paper.

A more general case also requires the following conditions:

- Rotation of the pieces of paper is allowed
- There can be multiple presents of the same dimensions

1 Introduction

The non-overlapment requirement of *PWP* links it to a specialization of the more general rectangle packing problem, in which we have a set of rectangles (our presents) of given dimensions that have to fit into a pre-determined square (the wrapping paper) of a given size.

Observing the assigned problem instances, we assume that the items will perfectly fit into the given container, without any kind of wasted space. This assumption greatly simplifies the problem, by reducing it from a minimization problem to a satisfiability one.

The following sections describe our implementation of different *PWP* solutions using both Constraint Programming and Satisfiability Modulo Theory approaches.

2 Input

Each instance of the problem is defined by:

- `n` ⟵ number of presents to be wrapped
- `w_paper` or `w` ⟵ width of the paper roll
- `h_paper` or `h` ⟵ height of the paper roll
- `presents` or `p` ⟵ list of presents dimensions, in the form $[width, height]$

To better represent equations in the following sections, `presents` is divided in two additional lists, i.e. `presents_xs` or `px` and `presents_ys` or `py`.

3 Constraint Programming

CP models are implemented with the MiniZinc language and models execution is managed by the official MiniZinc Jupyter extension, called iMiniZinc.

Following standard CP model guidelines we proceded by searching for global constraints, since they enable stronger propagation w.r.t user-defined ones, implied constraints, to allow a reduction of the search tree by pruning, channeling

constraints, which can be used to gain a different point of view over the problem, symmetry-breaking constraints, that remove symmetric non-solutions from being analyzed.

In our case-study we tried different approaches, by developing different models. Some of them tend to be faster in a specific subset of instances, w.r.t. the others. In the final model, we tried to put together the different key-points of each model.

In the following subsections each and every tested constriant, along with associated decision variables, will be carefully explained.

Inserire qui roba riguardo variabili/vincoli scartati.

3.1 Decision variables

Bottom-left corners

This is a two-dimensional list of decision variables (`bl_corners` or `b`), where each entry represents the bottom-left corner of a rectangle in the bounding box. Finding a satisfying assignement for this list is the main goal of this project. Moreover, the list is also used to graphically represent every instance solution.

To ease its usage two additional lists were defined (`bl_corners_xs` or `bx` and `bl_corners_ys` or `by`), by channeling over each dimension of the original list.

To reduce the search space, bottom-left corners variables domains are defined as follows:

- `bl_corners`: $0 \dots \max(h, w) - \min(\min(px), \min(py))$
- `bl_corners_xs`: $0 \dots w - \min(px)$
- `bl_corners_ys`: $0 \dots h - \min(py)$

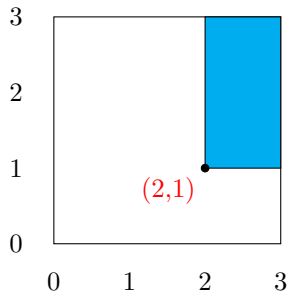

Figure 1: Bottom-left corner example

Top-right corners

As but representing the top-right corner of each rectangle (`tr_corners`). It is used to reduce the number of positions in which a rectangle can fall in, because it must be inside the bounding box.

To reduce the search space, `tr_corners` variables domain is defined as follows:

$$\min(\min(px), \min(py)) \dots \max(h, w)$$

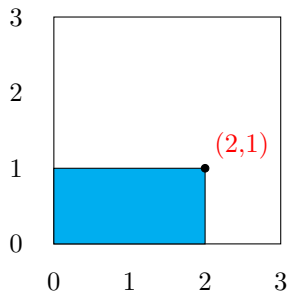

Figure 2: Top-right corner example

Bottom-left corners values

This is a list of decision variables representing a linearization of bottom-left corners (`bl_corners_values`), which uses a one-to-one mapping from each two-dimensional coordinate in the bounding box to an integer value.

The mapping operates as follows:

$$c : (x, y) \mapsto x + (y \cdot m),$$

where $m = \max(h, w)$.

To reduce the search space, `bl_corners_values` variables domain is defined as follows:

$$0 \dots c(w, h)$$

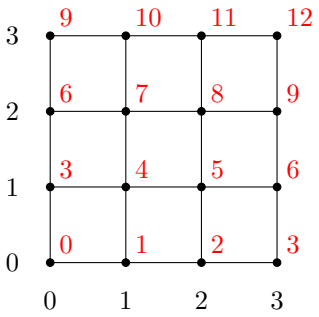

Figure 3: Example of 2D-coordinates linearization in a 3 by 3 box

3.2 Constraints

The constraints described below, divided by scope, are presented at the top of each section with a simple schema depicting their evolution throughout different models. The following is a legend explaining how constraints advancement is achieved:

- `A[x]`: Constraint `A` has been introduced in model number `x`
- `A[x]` → `B[y]`: Constraint `A` was removed in favor of `B`, in model `y`

- `A[x]` → ✗: Constraint `A` has not been carried over to models `x + 1, ...`

Model numbers are related to the organization inside the Jupyter notebook.

3.2.1 Non-overlapment

- Presents cannot overlap [1] → Global `diffn_k` [3]
- Global `all_different` [2]

Presents cannot overlap [1]

The idea behind this simple constraint is, given a rectangle, to avoid the existance of areas of overlap with every other rectangle.

$$\max(bx_i, bx_j) \geq \min(bx_i + px_i, bx_j + px_j)$$
$$\vee$$
$$\max(by_i, by_j) \geq \min(by_i + py_i, by_j + py_j)$$
$$\forall i, j \mid j > i$$

The described constraint has been observed to be efficient enough for relatively small instances of the problem, while already suffering to position rectangles in a 17×17 bounding box. Results are justified by the disjunctive nature of the constraint, which implies an higher burden in the propagation phase.

Global `diffn_k` [3]

The `diffn_k` global constraint is defined by the official MiniZinc documentation [2] as follows:
Constrains k-dimensional boxes to be non-overlapping. For each box i and dimension j, box_posn[i, j] is the base position of the box in dimension j, and box_size[i, j] is the size in that dimension. Boxes whose size is 0 in any dimension still cannot overlap with any other box.

```
constraint diffn_k(bl_corners, presents);
```

Being a global constraint it gives a stronger propagation and a more efficient search w.r.t to Presents cannot overlap [1], allowing us to solve bigger instances, up to a 23×23 bounding box.

It's also notable, as described by [1], that `diffn_k` is an onerous constraint. In [1] it accounts for 30 to 80% of the total running time in an implementation of *PSP (Perfect Square Packing)* problem, which is very much related to *PWP*.

Global `all_different` [2]

The `all_different` global constraint asserts that every variable has a different value assigned to it.

In our models it is used w.r.t `bl_corners_values` to ensure that every present has different `bl_corners`. The choice of the constrained variables is related to their one-dimensional nature, which guarantees compatibility with MiniZinc's implementation of `all_different`.

```
constraint alldifferent(bl_corners_values);
```

3.2.2 Containment

- Reduce presents domains [1]
- Areas summation [4]

Reduce presents domains [1]

Areas summation [4]

3.2.3 Positioning

- Global `count_eq` [2]
- Intervals approach [5] → ✗
- Anchor points [5] → Anchor points [6]→ ✗

Global `count_eq` [2]

Intervals approach [5]

Anchor points [5]

Anchor points [6]

3.2.4 Stacking

- Global `cumulative` [3]
- Column stacking by two [4] → General column stacking [5]

Global `cumulative` [3]

Column stacking by two [4]

General column stacking [5]

3.2.5 Symmetry breaking

- Biggest rectangle in lower left quadrant [4] → Ordering by areas [6]
- In-column ordering by width [4] → ✗

Biggest rectangle in lower left quadrant [4]

Ordering by areas [6]

In-column ordering by width [4]

3.3 Models

Search strategy

LaTeXis a very powerful tool when it comes to typesetting of mathematical equations. The quality of the output is extremely high and hardly matched by

other word processors. It takes little time with LaTeX to learn how to handle even complicated mathematical expressions.

$$\sigma_0 = \frac{\pi}{\sqrt{8}} \frac{1}{\tau_{\rm ff}} \tag{1}$$

$$K = \frac{\sqrt{32}}{\pi} \frac{1}{\delta} \frac{\tau_{\rm ff}}{\tau_{\rm co}} ; \tag{2}$$

LaTeX uses a simple and convenient system for assigning numbered labels to equations and other objects (figures, tables, etc...) and for referring to them. After having edited the source file and rearranged the position of the equations, LaTeX will change labels and references consistently throughout the text (if you did the things right of course...)

Examples of text containing mathematical expressions and equations:

M_r	mass internal to the radius r
m	mass of the zone
r_0	unperturbed zone radius
ρ_0	unperturbed density in the zone
T_0	unperturbed temperature in the zone
L_{r0}	unperturbed luminosity
$E_{\rm th}$	thermal energy of the zone

$$\tau_{\rm co} = \frac{E_{\rm th}}{L_{r0}} , \tag{3}$$

$$\tau_{\rm ff} = \sqrt{\frac{3\pi}{32G} \frac{4\pi r_0^3}{3M_{\rm r}}} , \tag{4}$$

$$\nabla_{\rm ad} = \left(\frac{\partial \ln T}{\partial \ln P} \right)_S , \ \chi_T = \left(\frac{\partial \ln P}{\partial \ln T} \right)_\rho , \ \kappa_T = \left(\frac{\partial \ln \kappa}{\partial \ln T} \right)_T$$

$$\frac{\pi^2}{8} \frac{1}{\tau_{\rm ff}^2} (3\Gamma_1 - 4) > 0 \tag{5}$$

$$\frac{\pi^2}{\tau_{\rm co}\tau_{\rm ff}^2} \Gamma_1 \nabla_{\rm ad} \left[\frac{1 - 3/4\chi_\rho}{\chi_T} (\kappa_T - 4) + \kappa_P + 1 \right] > 0 \tag{6}$$

$$\frac{\pi^2}{4} \frac{3}{\tau_{\rm co}\tau_{\rm ff}^2} \Gamma_1^2 \nabla_{\rm ad} \left[4\nabla_{\rm ad} - (\nabla_{\rm ad}\kappa_T + \kappa_P) - \frac{4}{3\Gamma_1} \right] > 0 \tag{7}$$

3.4 Results

In LaTeX You can enter text `verbatim`: that means that LaTeX will print it exactly as you enter it in the source file. The output resembles closely the one from old typewriters and it is usually good to print out portions of computer code:

```
PROGRAM area
REAL base, height, area
PRINT *,'Enter the values for the base and height of a triangle.'
```

```
READ *, base, height
area = (1.0/2.0) * base * height
PRINT *,'The area of a triangle with base ', base
PRINT *,'and height ', height,' is ', area
STOP
END
```

Note: In `verbatim` mode you can easily end up outside the margins, as in the example above: pay attention to that!

3.5 Lists

Example of a list with numbered items:

1. Planets, asteroids, moons ...
2. Stars, galaxies, quasars

Example of a list with unnumbered items:

- Planets, asteroids, moons ...
- Stars, galaxies, quasars

4 Results

In this section you present your findings and results.

5 Tables and figures

Figures demonstrate and prove conclusions. They should convince the reader, preferably at first glance. Figures should be self-explanatory. The legends should have a well-defined meaning. The lettering and the thickness of lines and symbols should be large enough to remain recognizable after printing.

The figure captions should contain all the information needed to understand the data presented and references to the text of the paper should be minimized.

Figure 4: Tables and figures are floating objects, LaTeX will place them where it thinks it's best (whatever that means...)

Tables should be self-explanatory. The table headings should contain the essential information needed to understand the data presented. Details should not clutter the header and are better added as explanatory footnotes.

Table 1: Example of table caption: opacity sources.

Source	$T/[\mathrm{K}]$
Yorke 1979, Yorke 1980a	≤ 1700
Krügel 1971	$1700 \leq T \leq 5000$
Cox & Stewart 1969	$5000 \leq$

6 Discussion

In this section you analyse and discuss your results. This section is paramount as it gives indication about the hability of the author to interpret the results and critically discuss his or her findings.

7 Conclusions

Here you summarize the essential aspects and findings of your work and analysis.

Finally, remember to include a section with the bibliography. It is very important to cite the sources you used for your study and for writing the report.

References

[1] Mikael Östlund. "Implementation and Evaluation of a Sweep-Based Propagator for Diffn in Gecode". In: 2017.

[2] Monash University. *MiniZinc*. URL: `https://www.minizinc.org/`.