Automated Pattern Selection using MiniZinc

S. Edelkamp, M. Martinez, I. Moraru

King's College of London

{*firstname.lastname*}*@kcl.ac.uk*

October 4, 2018

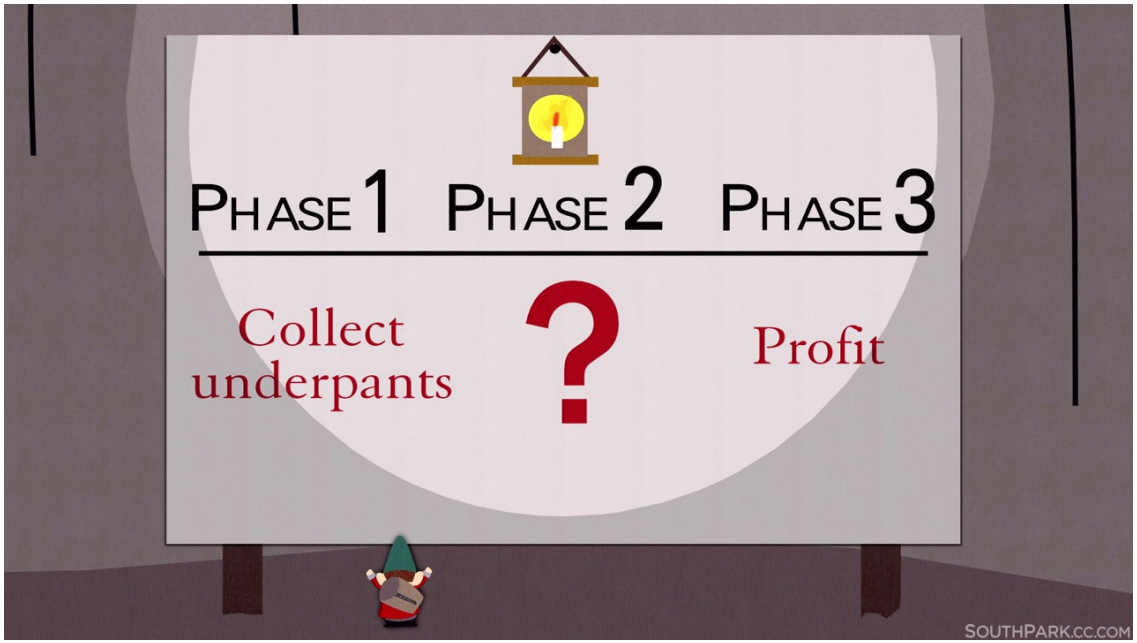
Overview

1. Introduction
   - AI Planning
   - Heuristic Search
   - PDBs
2. Automated Pattern Selection
   - Bin Packing
   - Bin Packing for Pattern Selection
   - MiniZinc
3. Conclusion
   - Results

Introduction

AI Planning

Planning is the discipline that has the task of coming up with a *sequence of actions* that will achieve a goal.



AI Planning Task

Definition (Planning Task)

A planning problem defined in propositional STRIPS is a 4-tuple $\Pi = \langle V, I, G, A \rangle$ where:

- V is a finite set of state variables;
- $I \subseteq V$ is the initial state of the problem;
- $G \subseteq V$ a set of goal variables;
- A is a finite set of actions (each action being composed out of preconditions, add effects, delete effects and a cost.

Bylander, 1994, showed that propositional STRIPS planning is *PSPACE-Complete*.

Heuristic Search

Heuristics are an intuition of how far a node is from the goal.

- They are relaxations of the problems constraints that solve a *relaxed* problem exactly;
- We can do an *informed* search through the state space by selecting the state that is *more promising* than the rest by using this *estimation*.

Definition (Heuristic)

Heuristic h is a function that maps states, $\mathcal{S}$, to positive numbers:

$$h : \mathcal{S} \rightarrow \mathbf{N}_+ \cup \{0, \infty\}$$

Heuristic families

- Delete relaxation - h^+ heuristic (Hofman and Nebel, 2001);
- Abstraction - pattern dababases (Culberson and Schaeffer, 1998; Edelkamp, 2001);
- Critical Paths - h^m heuristic (Haslum et al., 2000);
- Landmarks - landmark-cut heuristic (Helmert and Domshlak, 2009);
- Network Flows - flow heuristic (Van den Briel et al., 2006)

Pattern Databases

Definition

PDBs map the state space of a planning task onto an abstract state space by considering only a subset of the planning variables, the *pattern*.
The pre-computed optimal distance from each abstract state to an abstract goal state is stored in a look-up table, the *database*.

- Combining multiple PDBs can result in better heuristic estimates, than what would be returned by using only one.

Pattern Databases - Limitations

The main *limitation* of PDBs is the amount of memory needed, as during construction the abstract state space may prove to be too large;

- To deal with memory requirements, PDBs have been extended to symbolic search, which represents state sets in the search compactly as decision diagrams;

Another issue is the **automated selection** of the most informative patterns, which remains a combinatorial challenge.

Automated Pattern Selection

Pattern Selection

We identify a pattern database with its abstraction function ϕ.

- The *fitness* f of a pattern database ϕ (represented as a set of pairs $(a, h(a)) \in \phi$) is the average heuristic estimate:

$$f(\phi) = \sum_{(a,h(a))\in\phi} h(a)/|\phi|$$

- For several PDBs and cost partitioning γ, the values are added linearly:

$$f(\phi_1, \ldots, \phi_l) = \sum_{i=1}^{l} \sum_{(a,h_{i,\gamma}(a))\in\phi_i} h_{i,\gamma}(a)/|\phi_i|$$

The **pattern selection problem** is to find a selection of pattern databases that fit into main memory, and optimises f.

Bin Packing Problem

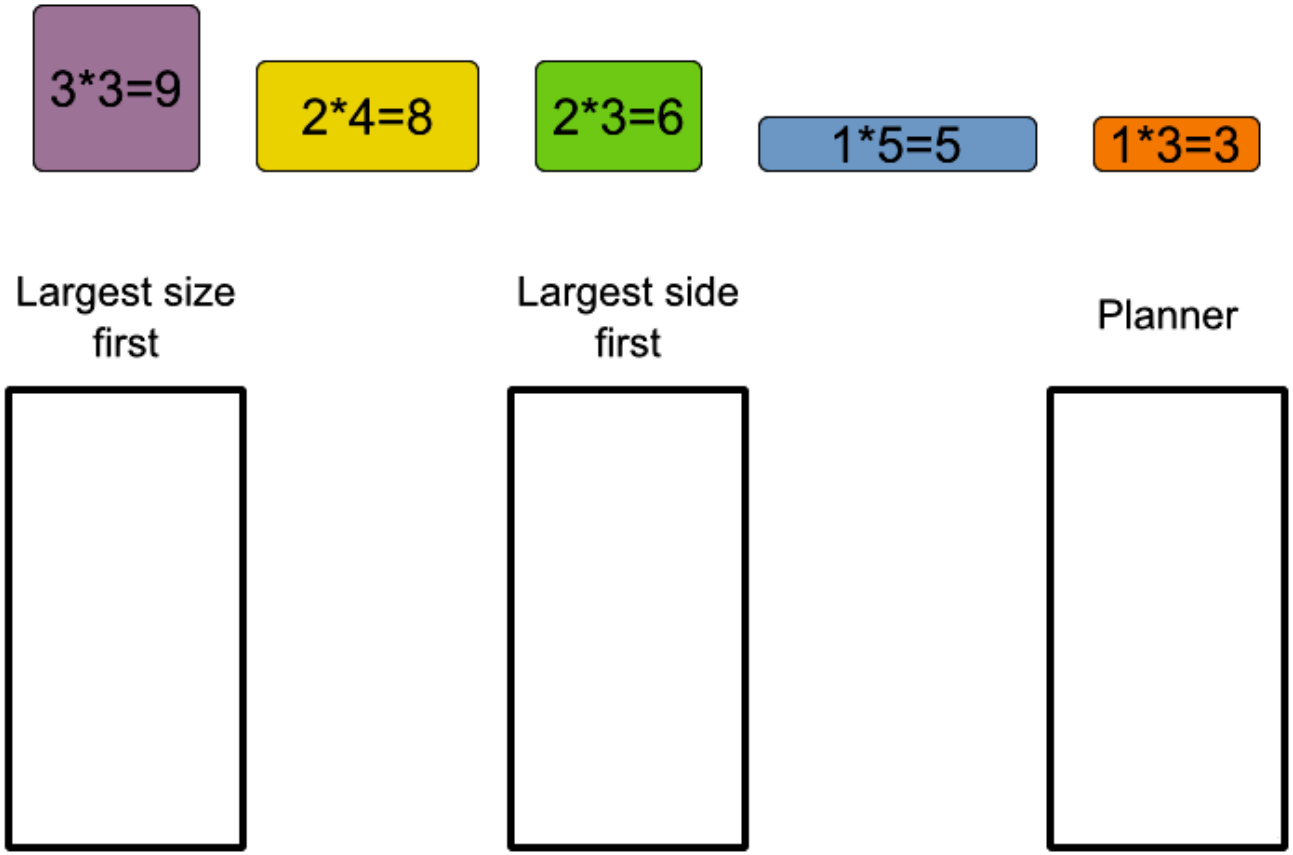
The **Bin Packing Problem (BPP)** is one of the first problems shown to be NP-hard (Garey and Johnson, 1979).

Definition (Bin Packing)

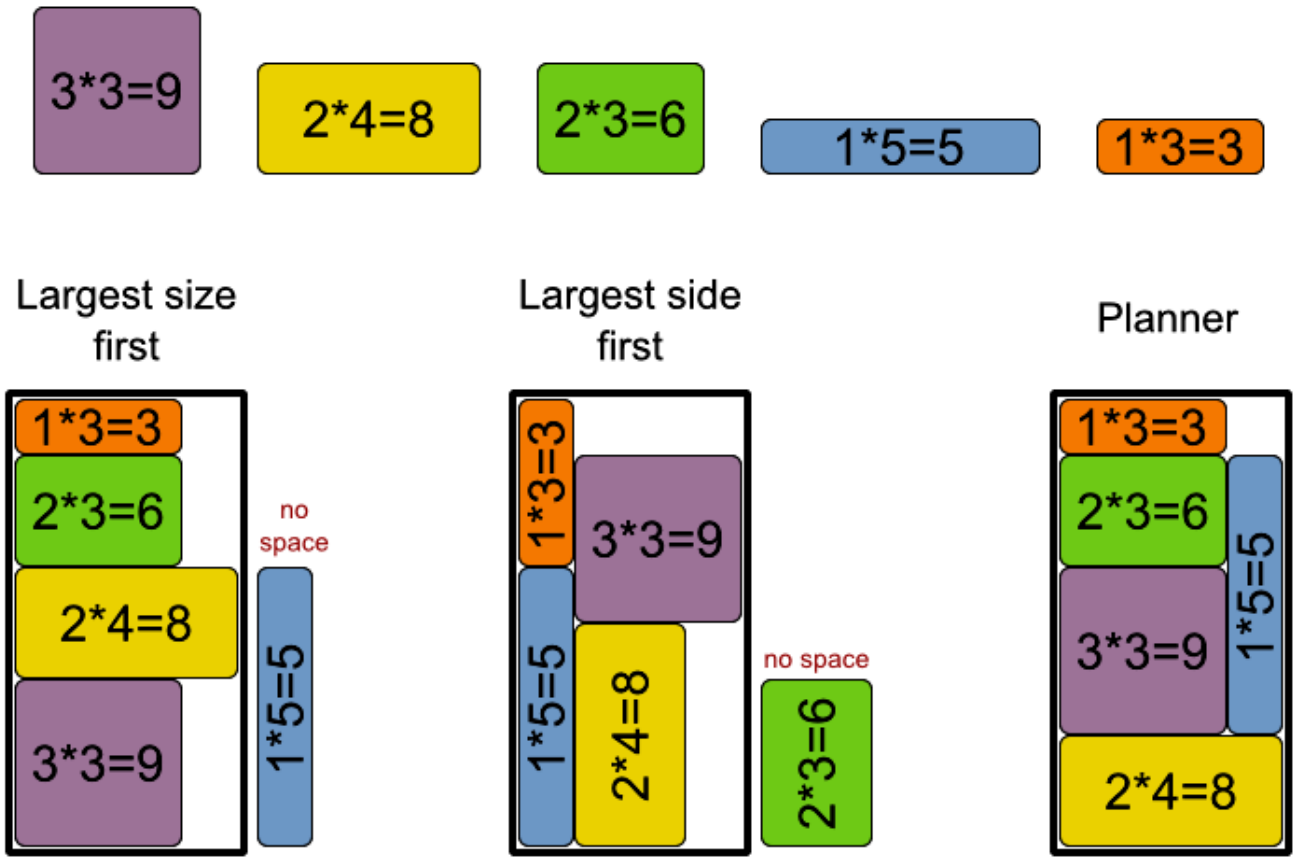
Given objects of integer size $a_1, \ldots, a_n$ and maximum bin size C, the problem is to find the minimum number of bins k so that the established mapping $f : \{1, \ldots, n\} \rightarrow \{1, \ldots, k\}$ of objects to bins maintains

$$\sum_{f(a)=i} a \leq C \text{ for all } i \leq k$$

Bin Packing Problem - Example (1/2)



Bin Packing Problem - Example (2/2)



Bin Packing for Pattern Selection

In the PDBs selection, we use a modified BPP:

- Estimate the expected size of the PDB by computing the product *(not the sum)* of the domain sizes, so that:
  - for a maximum bin capacity M, memory imposed, find the minimum number of bins k, so that:
    - the established mapping f of objects to bins maintains[1]

$$\prod_{f(a)=i} a \leq M \text{ for all } i \leq k$$

- In this case, $\prod_{f(a)=i}$ is an upper bound on the number of abstract states needed.

[1] By taking the logs on both sides, we are back to sums, but the sizes become fractional

MiniZinc for Pattern Selection

Our intent for solving the Pattern Selection problem is to be solver-independent, and to apply descriptive models:

- We looked methods that come in the form of a framework, and have choosen to use MiniZinc (Nethercote et al., 2007) a free and open-source constraint modeling language;
- MiniZinc is compiled into the input of a range of different constraint-based solvers including Gurobi (Gurobi Optimization, 2016) and Gecode (Gecode Team, 2006).

MiniZinc Model I

Example (Variables for BPP)

```
int: num_bins; int: num_objects;
array[1..num_objects] of int: object;
int: bin_capacity;
array[1..num_bins, 1..num_objects] of var 0..1: bins;
array[1..num_bins] of var 0..bin_capacity: bin_loads;
var 0..num_bins: num_loaded_bins;
```

MiniZinc Model II

Example (Constraints for BPP)

```
forall(b in 1..num_bins)
 (bin_loads[b] = sum(s in 1..num_objects)
 (object[s]*bins[b,s])) /\
sum(s in 1..num_objects) (object[s]) =
 sum(b in 1..num_bins) (bin_loads[b]) /\
forall(s in 1..num_objects)
(sum(b in 1..num_bins)(bins[b,s]) = 1) /\
decreasing(bin_loads)  /\ bin_loads[1] > 0 /\
num_loaded_bins = sum(b in 1..num_bins)
(if bins_of_load[b] > 0 then 1 else 0 endif);
```

Conclusion

Experimental results

We analyze two planners planners, **Planning-PDBs** and **MiniZinc-PDBs**, that differ only in their Pattern Selection process.

Experimental results

The numbers represent the amount of solved problems on each domain out of a total of 20 problems.

ALT-TAB

MiniZinc for Pattern Selection

- Our first attempt to integrating MiniZinc library version 1.6 resulted to a number of problems failing - as such we did not include it in the IPC submission;
- We have since tried to integrate version 2.2, receiving results but hiting some interesing errors on the way.

Concluding remarks

- Is the Bin Packing Problem the right way for choosing patterns to go into the heuristic?
- How can we take more advantage of the engineering strenghts of PDBs?
- Can we translate the results from Classical planning to some that are more expressive?

Concluding remarks

- Is the Bin Packing Problem the right way for choosing patterns to go into the heuristic?
- How can we take more advantage of the engineering strenghts of PDBs?
- Can we translate the results from Classical planning to some that are more expressive?

- *I still have three years of my PhD, hopefully I can advance in these questions.*

Multumesc pentru atentie!