Lazy Clause Generation for Bin Packing

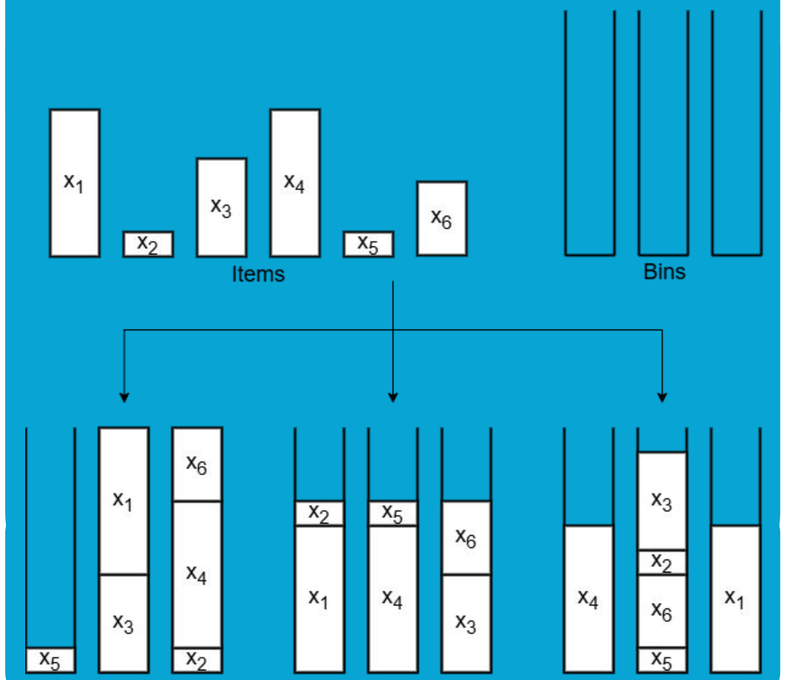
Author:
Melvin de Kloe m.dekloe@student.tudelft.nl

Supervisors:
Maarten Flippo m.l.flippo@student.tudelft.nl
Emir Demirović e.demirovic@tudelft.nl

1 Introduction

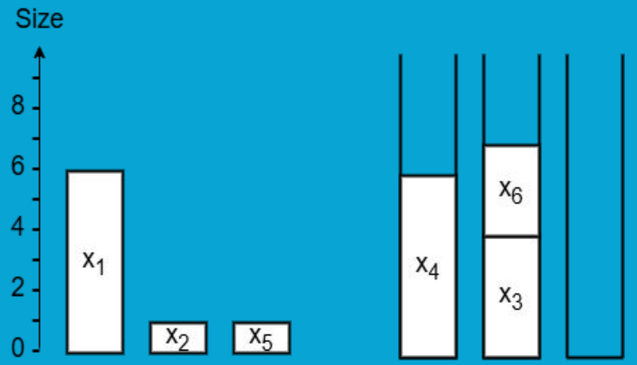
- Bin Packing describes resource allocation
- NP-Complete
- Existing algorithms are not flexible

A bin packing problem and 3 solutions



- Lazy Clause Generation (LCG) is a branch of Constraint Programming (CP)
- Very flexible
- Requires explanations for every step

2 Explanations



- x_1 has to go in the 3rd bin
- Can only use [[var <= val]] or [[var == val]]
- Which explanation to use?
  - $[[x_4 == \text{bin } 1]] \land [[x_3 == \text{bin } 2]] \land [[x_6 == \text{bin } 2]]$
  - $[[\text{load } 1 == 6]] \land [[\text{load } 2 == 7]]$
  - $\neg[[\text{load } 1 <= 4]] \land \neg[[\text{load } 2 <= 4]]$

3 Problem

- CP propagators already exist [1], but not for LCG
- Using these propagators, develop new propagators for LCG, and explain every step they make

References

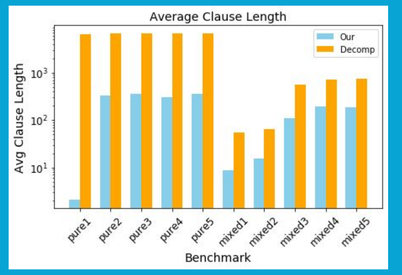
[1] P. Shaw, *A constraint for bin packing*

4 Results

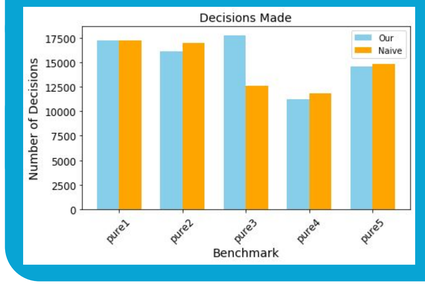
Objective Scores:

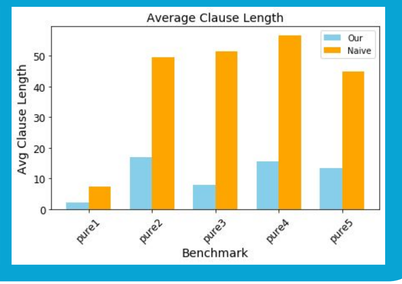
	pure1	pure2	pure3	pure4	pure5	mixed1	mixed2	mixed3	mixed4	mixed5
our	17	19	14	fin	fin	fin	fin	9654	6357	6271
decomp	N/A	N/A	N/A	N/A	N/A	fin	fin	8646	8363	8300

Decomposition Comparison:



Naive Explanations Comparison:





5 Conclusions

- Complete improvement over decomposition in a pure benchmark, but clause quality always improved.
- Marginal difference in decisions made with naive explanations, but clause quality again improved.