Compressed Pattern Databases for Classical Planning

Bachelor's Thesis Presentation

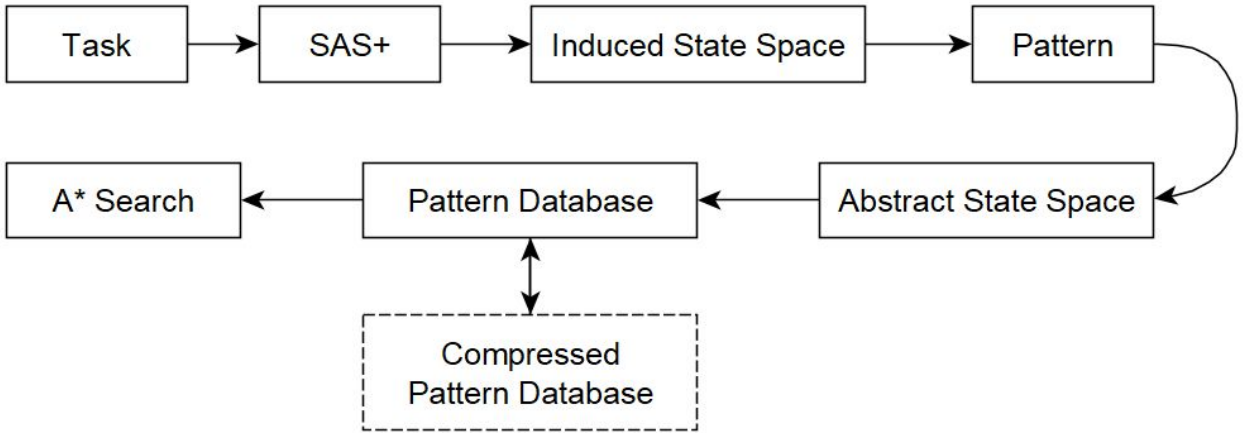
11 December 2020

Pascal Mafli
pascal.mafli@unibas.ch

Supervisor: Dr. Silvan Sievers
Examiner: Prof. Malte Helmert

- Introduction
- Background:
  - State Space, Abstract State Space
  - Pattern Database
  - Pattern Database Compression (min-compression)
  - Cost Partitioning
- Boosting
- Results
- Conclusion

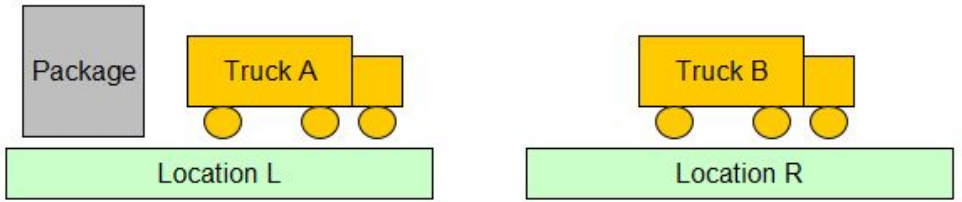
Introduction



Example: Logistics task 1 Package, 2 Trucks

V = { pack, t_A, t_B }
dom(pack) = { L, R, A, B }
dom(t_A) = dom(t_B) = { L, R }
Actions:

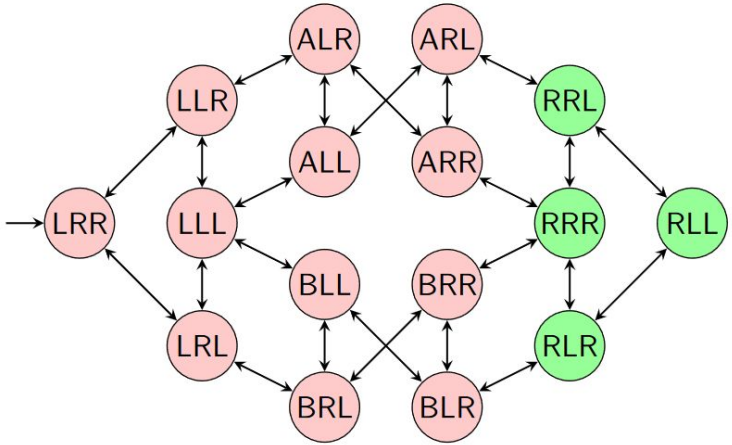
- load/unload package
- move truck



State Space

- Finite set of states
- Finite set of actions
- Cost Function
- Transition relation
- Initial State
- Goal States

State { pack $\mapsto$ i, $t_A \mapsto$ j, $t_B \mapsto$ k}
represented as ijk

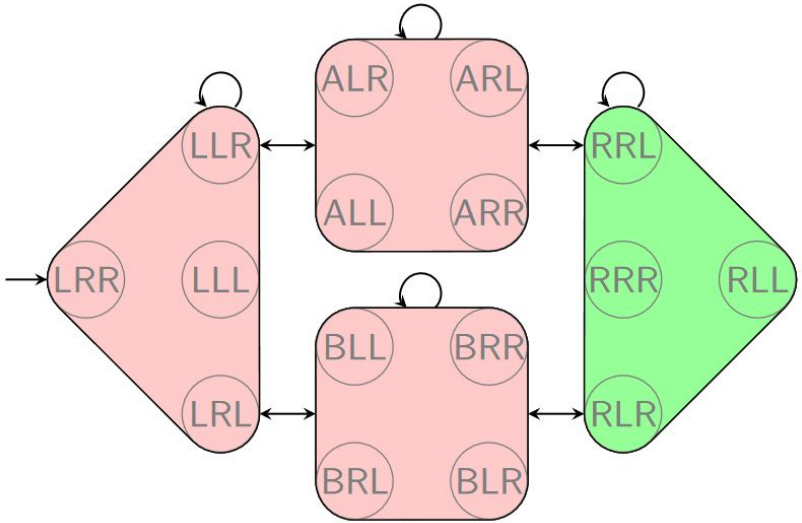


Example taken from the Foundations of Artificial Intelligence lecture (Slideset ai37)

Abstract State Space (1)

- Projection using a pattern p
- p is a subset of all variables

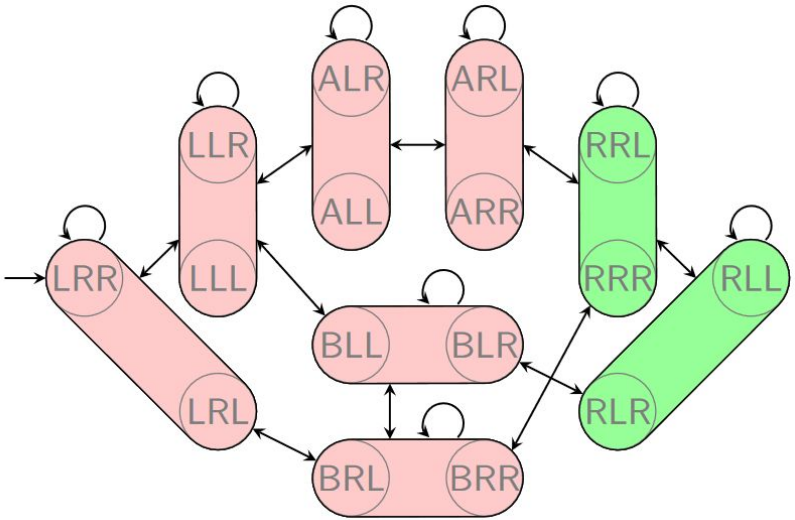
Abstraction induced by projection on p = { pack }



Example taken from the Foundations of Artificial Intelligence lecture (Slideset ai37)

Abstract State Space (2)

Abstraction induced by projection on p = { pack, t_A }



Example taken from the Foundations of Artificial Intelligence lecture (Slideset ai37)

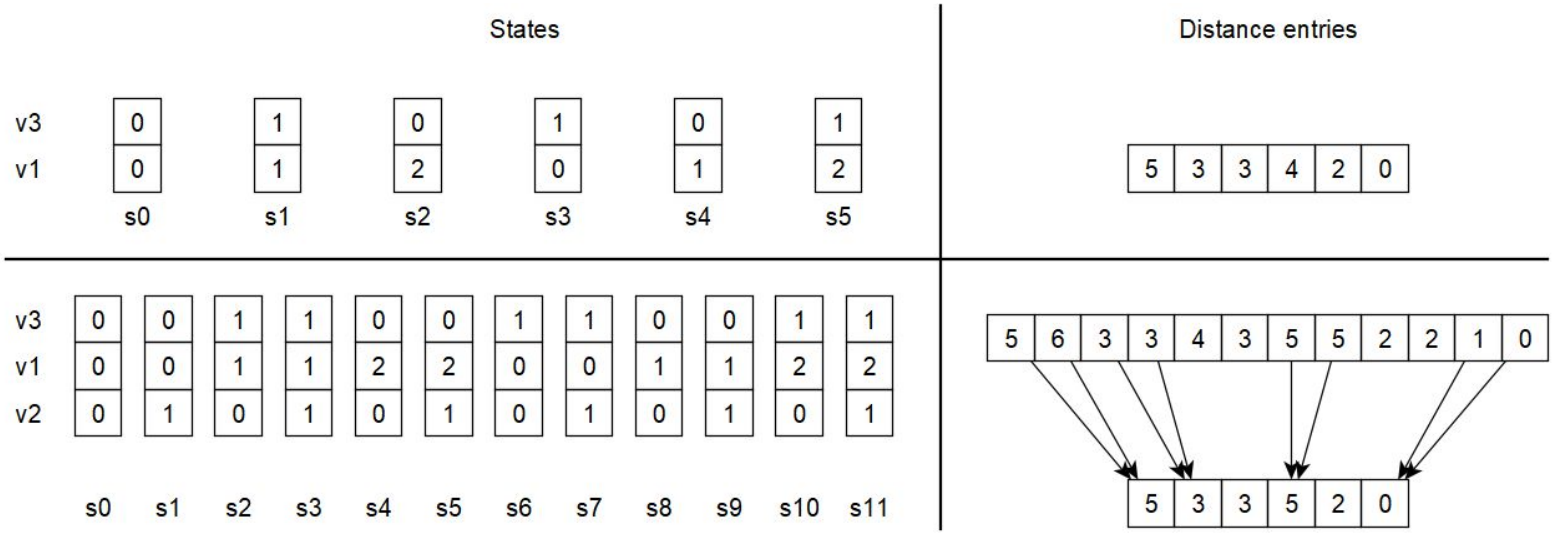
Pattern Database (PDB) Heuristic

Consist of:

- Pattern
- Calculated using the abstract state space induced by the projection
- Shortest distance to a goal state for each abstract state is stored (Dijkstra)
- Ranking/Unranking function to map a concrete state to an abstract state

PDB Compression (min-compression)

Compress a PDB with size M to a PDB with size N



Background - Cost Partitioning

- Way to combine single admissible PDB heuristics to one combined admissible PDB heuristic
- Used in this thesis:
  - Canonical cost partitioning (CAN)
  - (Greedy) Zero-One cost partitioning (gZOCP)
  - Uniform cost partitioning (UCP)
  - Opportunistic uniform cost partitioning (oUCP)
  - Saturated cost partitioning (SCP)

Boosting - Problem

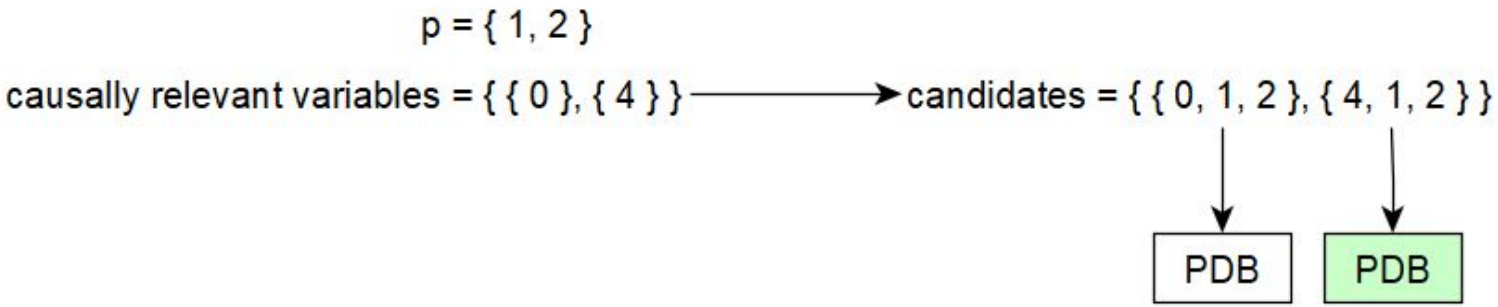
Given PDB P constructed on pattern p

- Find a bigger pattern p’ and calculate PDB P’
- Min-compress PDB P’ to a PDB P’’
- Evaluate P’’
- Use better of the two PDBs

Boosting - Finding expanded pattern

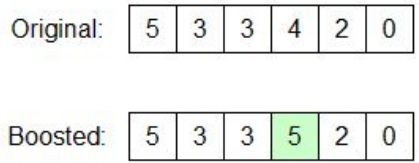
- Causally relevant variables (current pattern) are possible candidates
- Randomwalk
  - Add variables to the pattern until the number of abstract states reach a certain size
  - Calculate PDB P’ and evaluate
- Hillclimbing
  - Calculate PDBs for all candidate patterns (pattern $\cup$ { candidate variable })
  - Evaluate all PDBs and keep best for next iteration
  - Proceed until termination condition is reached

Boosting - Hillclimbing



Boosting - Evaluation

- Compare the distance entries of both PDBs P and P'' (boosted)
  - If the boosted PDB has a higher value in an entry, the heuristic is better



Experiments

- Experiments performed on sciCORE cluster, using tasks from the IPC optimal track collection
- Time limit 30 minutes, memory limit 3.5 GiB RAM, search algorithm A^*
- All Cost Partitioning methods compared using baseline (not-boosted) and boosted approach

Metrics used:

- Coverage (Cov): Number of solved tasks. Larger is better.
- Exp. until last jump (Exp): Expansions during search, excluding last f-layer. Smaller is better.
- Out of Memory (OOM): Sum of the tasks resulting in Out of Memory. Smaller is better.
- Out of Time (OOT): Sum of the tasks resulting in Out of Time. Smaller is better.
- Total time (Time): Overall time (geometric mean). Smaller is better.

Results (1)

Canonical cost partitioning:

	Baseline	HC1	HC1L	HC2	HC2L	HC3	HC3L	RND	RNDL
Cov	919	910	923	909	924	900	923	**928**	**928**
OOM	685	634	683	**616**	680	635	681	674	675
OOT	204	264	**202**	283	204	273	204	206	205
Exp	1021M	879M	918M	858M	915M	841M	915M	**802M**	802M
Time	**2.79**	4.67	3.37	5.07	3.39	5.22	3.42	5.73	5.60

Legend:
Hillclimbing: HC, Parameters: #Iterations, Limited. HC1L = Hillclimbing 1 iteration, Limited abstract states
Randomwalk: RND, Parameter: Limited. RNDL = Randomwalk, Limited abstract states

Results (2)

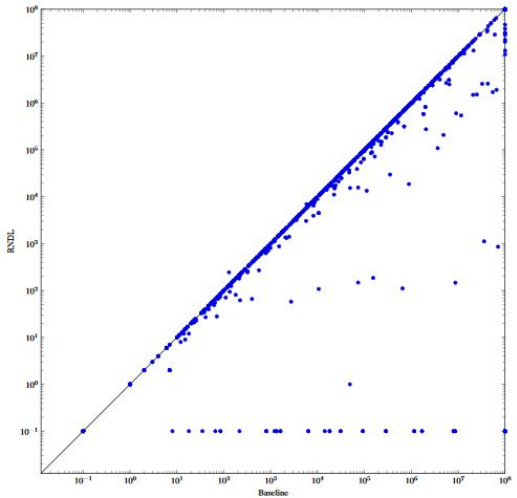
Uniform cost partitioning:

	Baseline	HC1L	HC2L	HC3L	RNDL
Cov	**860**	858	858	858	851
OOM	738	723	722	722	**715**
OOT	**210**	227	228	228	242
Exp	1567M	1545M	1536M	**1535M**	1548M
Time	**3.41**	5.05	5.05	5.11	12.10

Legend:
Hillclimbing: HC, Parameters: #Iterations, Limited. HC1L = Hillclimbing 1 iteration, Limited abstract states
Randomwalk: RND, Parameter: Limited. RNDL = Randomwalk, Limited abstract states

Results (3)

Saturated cost partitioning:
Expansions (x-axis: Baseline, y-axis: Randomwalk limited)



- Big reduction on expansions
- Some tasks have 0 expansions until last jump
  - woodworking-opt08-strips
  - woodworking-opt14-strips

Conclusion

- Boosting has a positive effect on the coverage in some cost partitioning methods
  - Biggest improvement: Canonical CP 919 solved tasks to 928 tasks (+9 coverage)
- The additional computation for the boosting is time consuming
  - For example SCP SYS(2): Baseline 0.89 (geom. mean) to 8.75 (geom. mean) for Randomwalk
- For UCP and oUCP there was a negative impact for boosted PDBs

Further topics

- Orders of heuristics for certain cost partitioning methods (oUCP, SCP, ZOCP)
- Analysis of domains, where boosting had a big impact
  - woodwork-opt08-strips and woodwork-opt14-strips (for SCP)
- Finding more performant boosting methods

Questions?