Implementing Symbolic Pattern Databases for Planning

Matthew Fahrni, April 27, 2023

Planning & State Spaces

- State Space $\mathbf{S} = \langle S, A, cost, T, s_0, S_* \rangle$
  - S finite set of states
  - A finite set of actions
  - *cost* representing cost of action
  - $T \subseteq S \times A \times S$ transitions
  - s_0 initial state
  - S_* goal states
- The purpose is to find a sequence of actions $(a_1, ..., a_n), a_i \in A$ to get from the initial state to a goal state
- optimal planner finds sequence of actions with minimal cost

SAS+ Tasks

- State Spaces too big to represent
- instead use SAS+ Task as representation
- SAS+ Planning task is $\Pi = \langle V, I, G, A \rangle$
  - V are the variables, each with a finite domain
  - I is the initial state
  - G are the goal states
  - A are the actions, consisting of
    - the preconditions of the action
    - the effect of the action
    - the cost of the action

Pattern Database

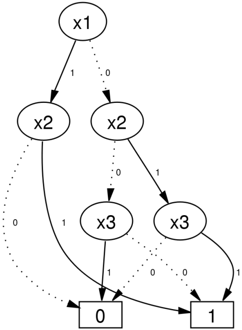
- solve problem for a subset of variables
- save the optimal cost to goal for every abstract state
- use singular or multiple pattern databases as a heuristic in original state space

Symbolic Data Structures

- Many different Decision Diagrams exist
- We focus on Binary Decision Diagrams (BDD) and Algebraic Decision Diagrams (ADD)

Binary and Algebraic Decision Diagram

- directed acyclic graph
- every nonterminal node has exactly two arcs
- BDD has one or two terminal nodes
- ADD has finite number of terminal nodes



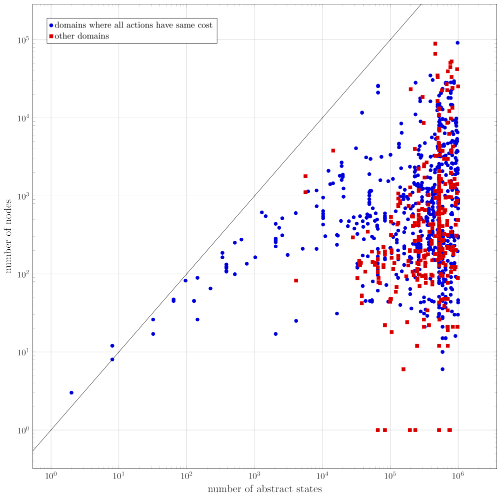
Implementation

- Represent every variable $v_i \in V$ as one or multiple BDD
- Introduce primed variables for every variable
- Represent the actions as Transition Relation (TR), encode them as BDD
- Optionally merge same cost TR
- Use Dijkstra algorithm to calculate the cost of every state
- Save these costs inside an ADD

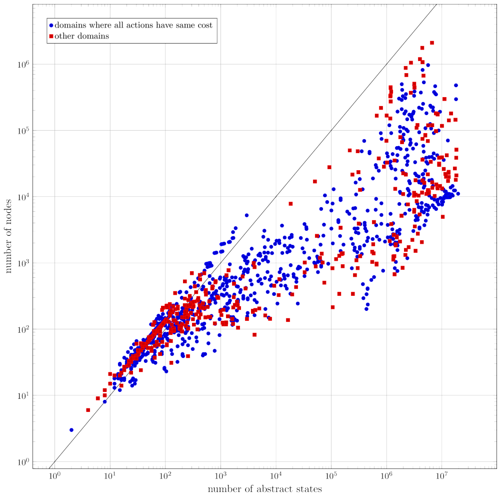
Result

- implemented into Fast Downward (https://www.fast-downward.org)
- using Colorado University Decision Diagram (CUDD) for BDD and ADD
- the analysis was performed using the sciCORE (http://scicore.unibas.ch/) scientific computing center at University of Basel
- results parsed using LAB (https://github.com/aibasel/lab)

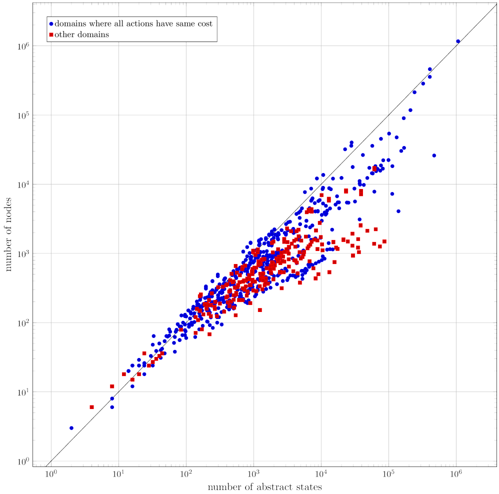
Size Comparison: Greedy Pattern Generator



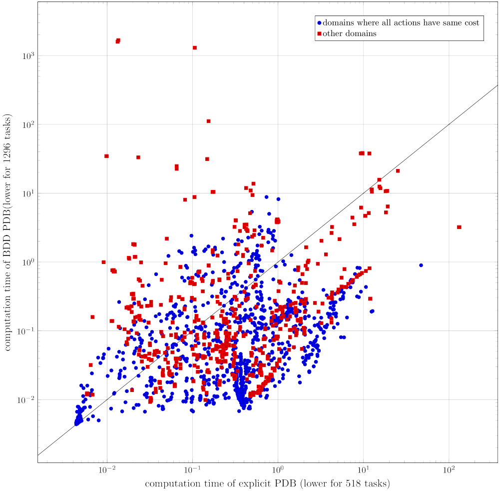
Size Comparison: Hillclimbing Pattern Collection Generator



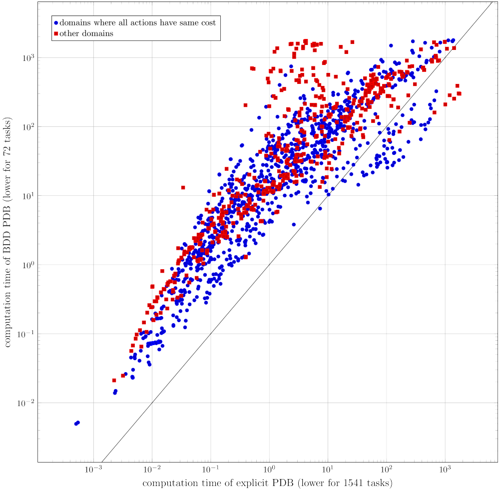
Size Comparison: Systematic Pattern Collection Generator



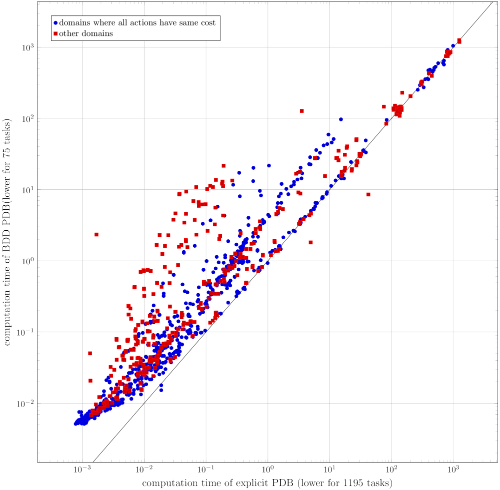
Computation Time Comparison: Greedy Pattern Generator



Computation Time Comparison: Hillclimbing Pattern Collection Generator



Computation Time Comparison: Systematic Pattern Collection Generator



Improvements

Extensions

Conclusion

Thank you
for your attention.