Automatic Move Pruning Revisited

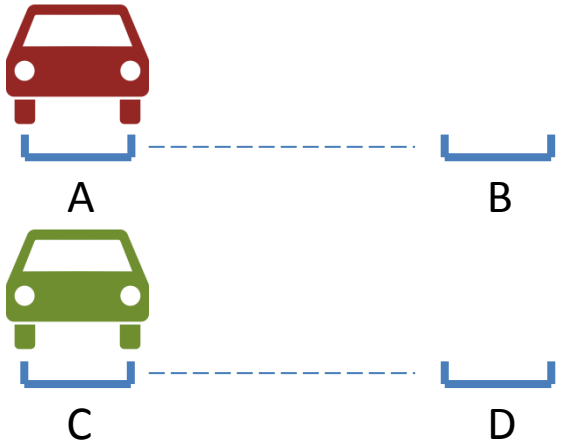
Neil Burch, Robert Holte

SoCS 2012

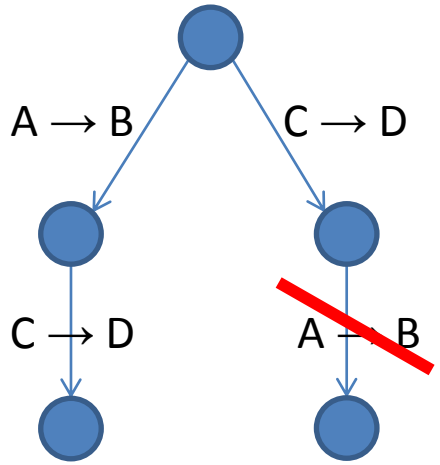
Automatic Pruning

- Problem: Even with almost perfect heuristics, search space can grow exponentially
  → «How good is almost perfect?» (Helmert & Röger, 2008)
- Possible solution: In every search state, ignore – «prune» – some actions
- Preserve optimality by keeping at least one optimal path
- Automatic means domain independant

Example



- Goal: red to B, green to D



- Move sequences:
  [A → B, C → D]
  [C → D, A → B]
- Prune [C → D, A → B]

«Moves»

- Paper considered other state space representation: PSVN
- More powerful than STRIPS, similar to SAS$^+$
- «Moves» are ‹equivalent› to operations in STRIPS
- Example move:

$$a\colon (0, X_1, X_1) \rightarrow (1,0, X_1)$$

Road Map

- Pruning technique to identify redundant move sequences
- Example showing erroneous pruning
- Revised pruning technique

Fundamental Pruning Principle

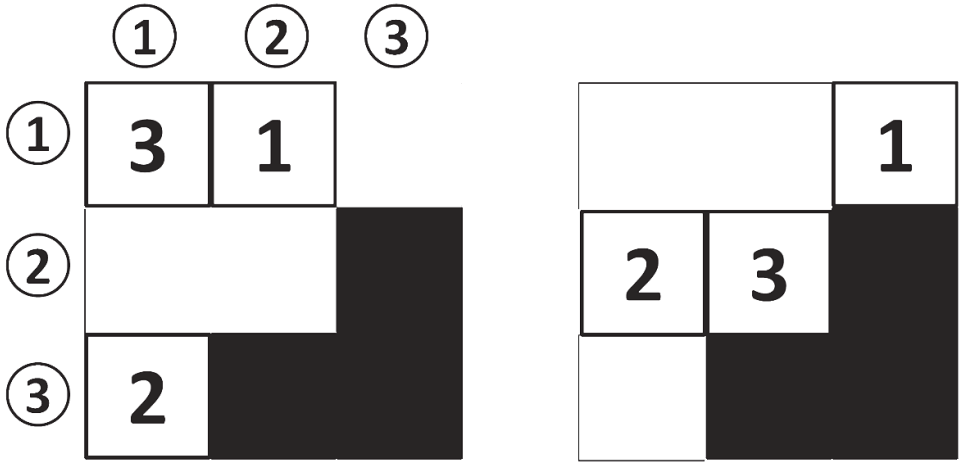
- Taylor and Korf in 1992/1993:
  «We can prune move sequence B if there exists a move sequence A such that
  i. $\text{cost}(B) \geq \text{cost}(A)$,
  ii. every state that satisfies preconditions of B satisfies also A,
  iii. applying A and B leads to the same state.»
- ‹B redundant with A›, denoted as $B \geq A$

Pruning Redundant Move Sequences

- Problem:
  Pruning all redundant move sequences
  eliminates optimality and completeness!
- Why?

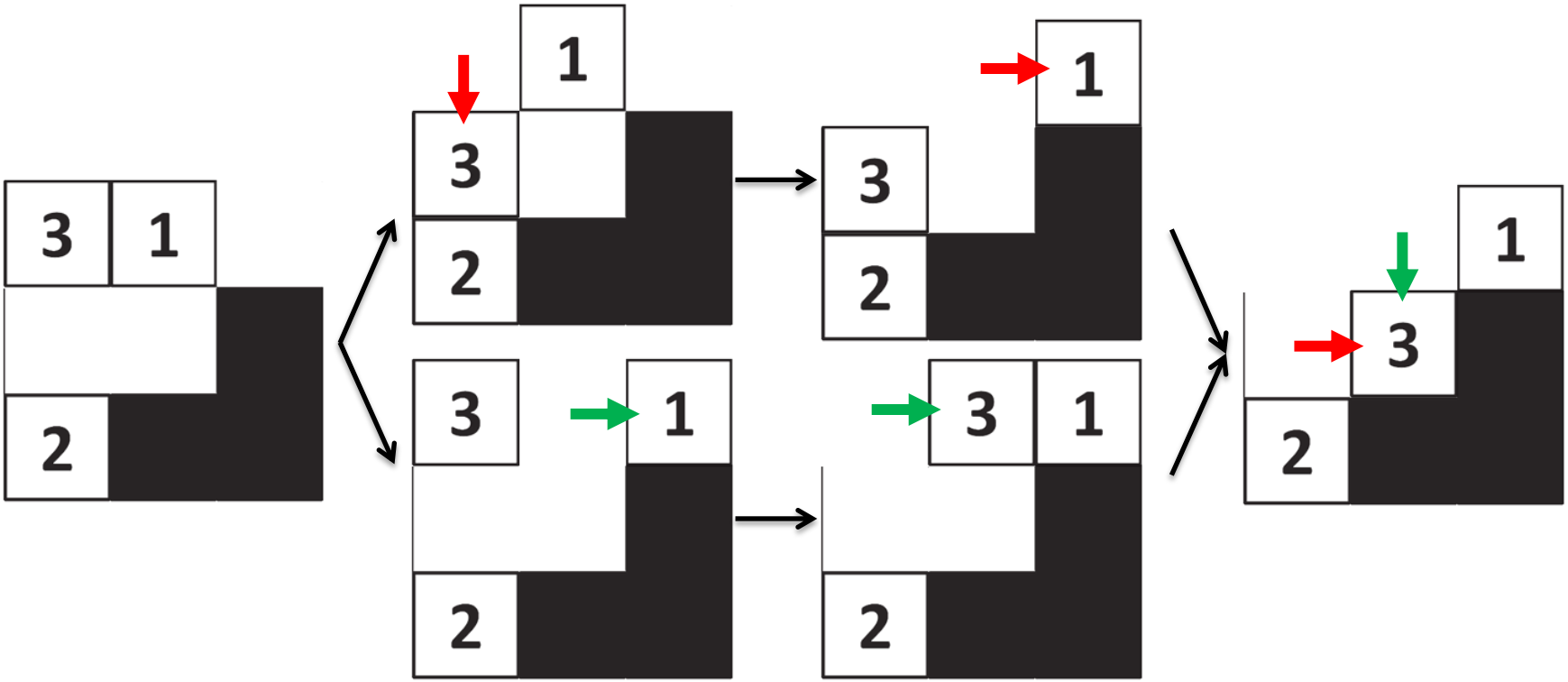
Sliding Tile Example

- Optimal length: 4 moves



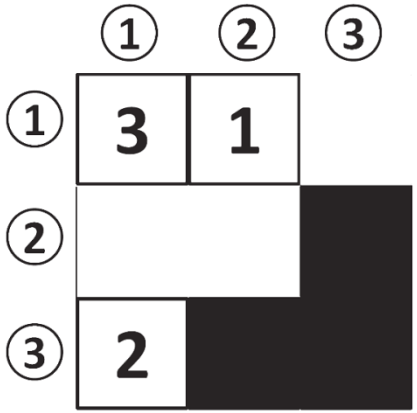
Redundant seq.		Because of
12R – 11D	≡	11D – 12R
11D – 12R – 21R	>	12R – 11R – 12D
11R – 12D – 31U	>	11D – 21R – 31U

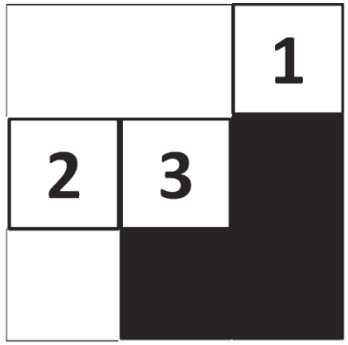
Redundant Move Sequences



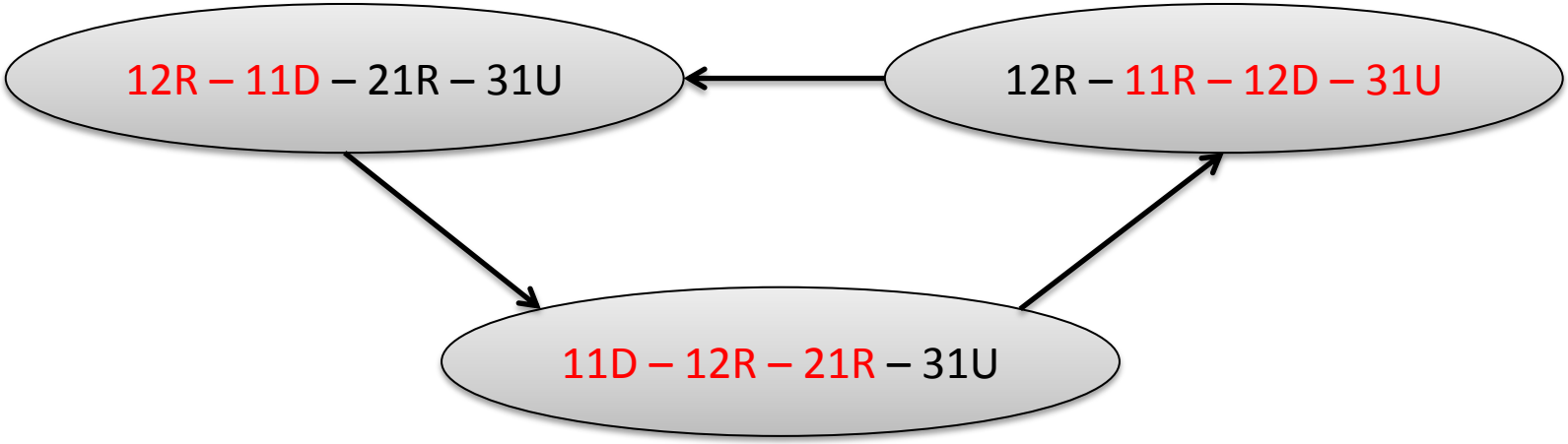
Redundant seq.		Because of
12R – 11D	≡	11D – 12R
11D – 12R – 21R	>	12R – 11R – 12D
11R – 12D – 31U	>	11D – 21R – 31U

Erroneous Pruning





Redundant seq.		Because of
12R – 11D	≡	11D – 12R
11D – 12R – 21R	>	12R – 11R – 12D
11R – 12D – 31U	>	11D – 21R – 31U



First Conclusion

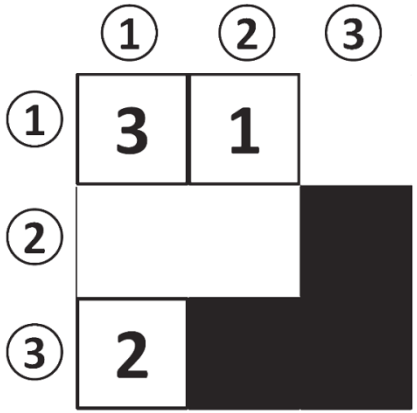
- Redundancy of move sequences correctly identified
- Interaction of multiple redundancies can lead to incorrect behaviour
- Revise technique to leave one least-cost path

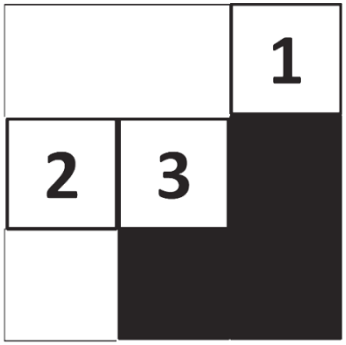
(One) Solution

- Introduce total ordering $\mathcal{O}$ on move sequences:
  $B >_{\mathcal{O}} A$: B is greater than A under $\mathcal{O}$
- Theorem:
  If $B \geq A$ and $B >_{\mathcal{O}} A$,
  then B does not occur in the ‹smallest› least-cost path from s to t for any states s, t.

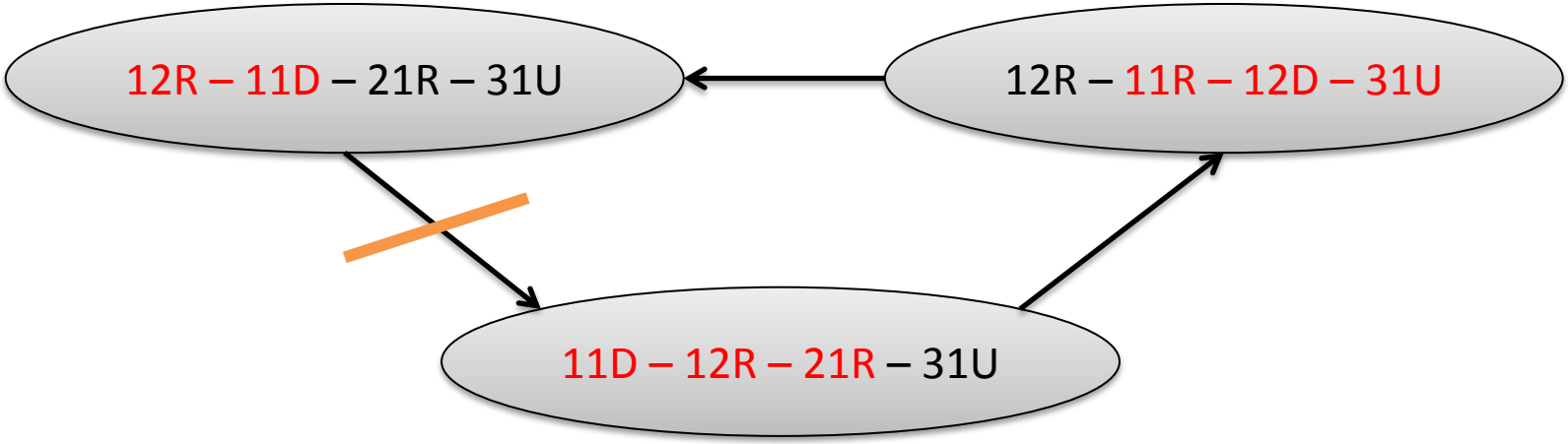
$\rightarrow$Prune move sequences B that are redundant with A and greater than A

No More Erroneous Pruning





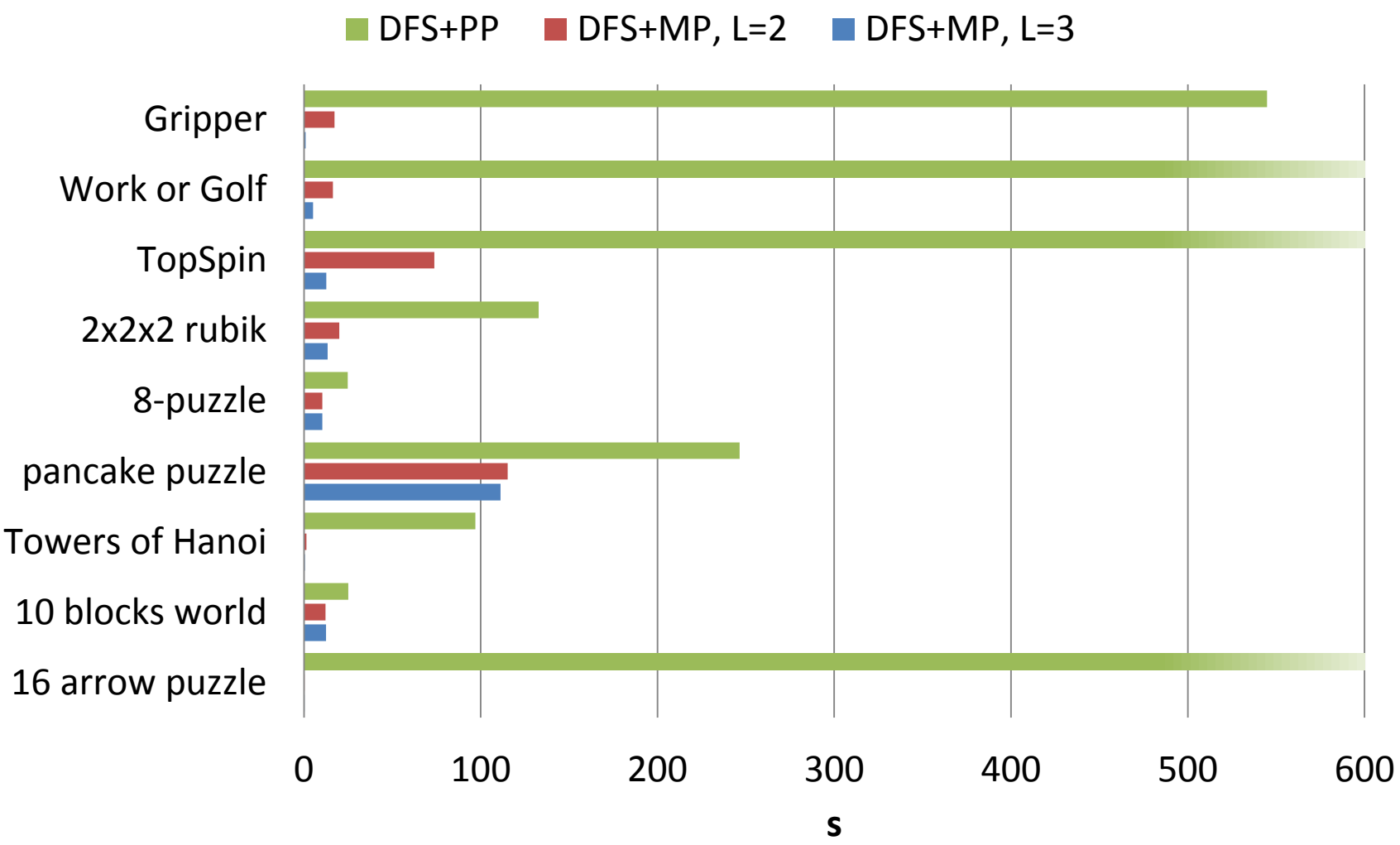
	$\mathcal{O}$	
12R – 11D	<	11D – 12R – 21R
11D – 12R – 21R	>	11R – 12D – 31U
11R – 12D – 31U	>	12R – 11D



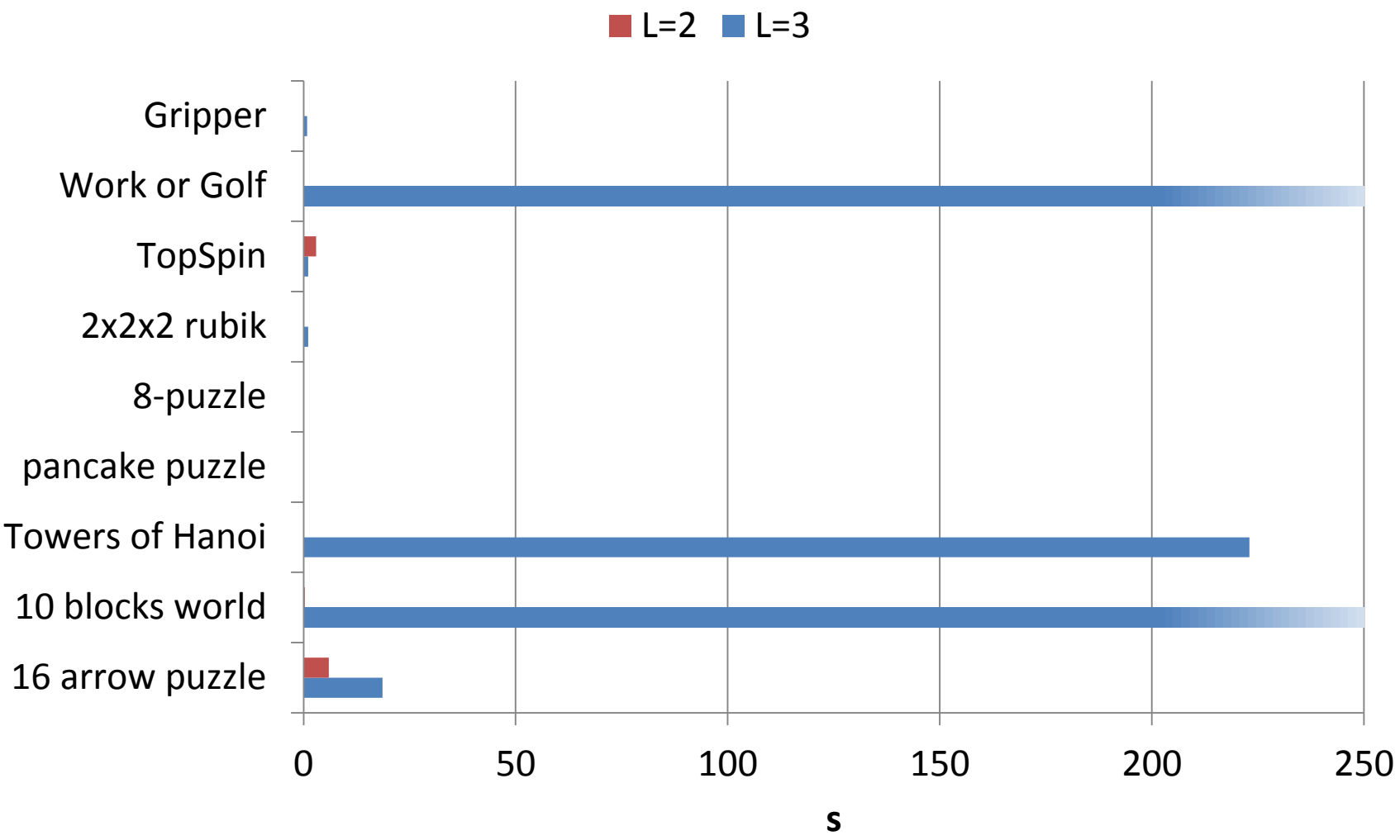
Evaluation

- Tree search (DFS with depth bound)
- Pruning with sequences of length 2 and 3
- Pruning analysis done in advance
- 2.83 GHz Q9550 and 8 GiB RAM

Actual Search Time



Pruning Analysis



Summary

- Introduced move pruning – technique based on identifying redundant move sequences
- Can improve DFS significantly

Future Work

- Investigate «good» orderings to improve technique
- Study effects of move pruning on graph search

Domain	d	DFS+PP	DFS+MP, L=2	DFS+MP, L=3
16 arrow puzzle	15	? >3600s	3,277 0.39s 0.07s	3,277 0.39s 18.58s
10 blocks world	11	352,028 25.02s	352,028 12.23s 5.97s	352,028 12.53s 507s
Towers of Hanoi	10	1,422,419 97.02s	31,673 1.45s 0.30s	9,060 0.49s 223s
Pancake puzzle	9	5,380,481 246.49s	5,380,481 115.22s 0.01s	5,288,231 111.18s 0.02s
8-puzzle	25	368,357 24.77s	368,357 10.40s 0.01s	368,357 10.40s 0.08s
2x2x2 Rubik	6	2,715,477 132.74s	833,111 20.00s 0.02s	515,614 13.35s 1.10s
TopSpin	9	? >3600s	2,165,977 73.80s 0.00s	316,437 12.59s 1.11s
Work or Golf	13	? >3600s	209,501 16.44s 2.98s	58,712 5.14s 904s
Gripper	14	9,794,961 544.85s	590,870 17.22s 0.08s	25,982 0.95s 0.85s