Incremental LM-cut

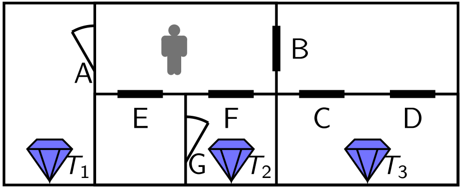
Florian Pommerening and Malte Helmert

Universität Basel
Departement Informatik

14. 06. 2013

Content

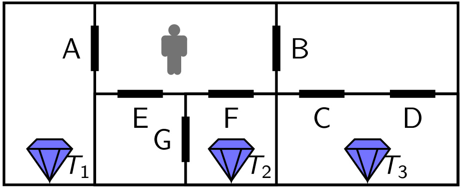
Heuristic Search for STRIPS Planning



- Variables $V = \{\text{A-Is-Open}, \text{B-Is-Open}, \dots\}$
- Initial state $I = \{\text{A-Is-Open}, \text{G-Is-Open}\}$
- Goal state $G = \{\text{Has-}T_1, \text{Has-}T_2, \text{Has-}T_3\}$
- Operators $O = \{o_A, o_B, \dots, p_1, p_2, \dots\}$
- Plan $\pi = \langle p_1, o_F, p_2, o_B, o_C, p_3 \rangle$

- *Heuristic* function
  - Distance estimate
  - Admissibility
- Search methods
  - A* Search
  - IDA* Search

Disjunctive Action Landmarks

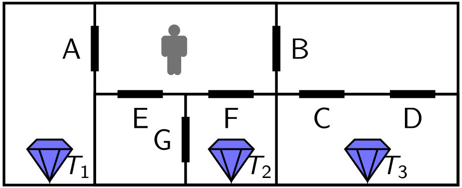


- *Disjunctive action landmarks*
  - Set of operators
  - Every plan contains at least one of them
  - Cost of a landmark: cost of cheapest contained operator
- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}, \ldots$

$h^{\text{LM-cut}}$

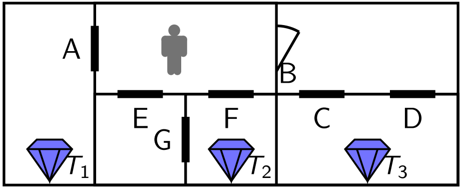
1. Calculate $h^{\max}(s)$
   - Only achieve most expensive subgoal/precondition
   - $h^{\max}(s) = \infty$ task unsolvable
   - $h^{\max}(s) = 0$ stop searching for landmarks
2. Use $h^{\max}$ values to discover new landmark L
3. Reduce cost of each operator in L by L's cost
   - Introduces a cost partitioning
   - Sum of landmark costs is admissible heuristic
4. Repeat

Incremental Computation (Example)



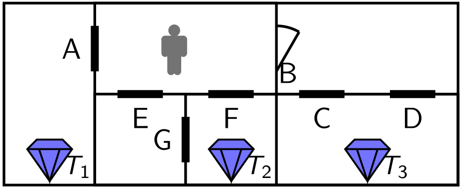
- Discovered landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Landmarks after application of o_B?

Incremental Computation (Example)



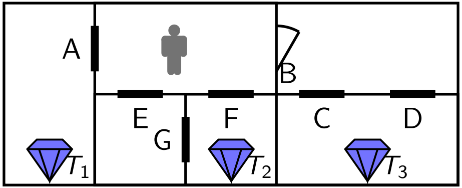
- Discovered landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Landmarks after application of o_B
  - $\{o_A\}$ remains landmark

Incremental Computation (Example)



- Discovered landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Landmarks after application of o_B
  - $\{o_A\}, \cancel{\{o_B, o_C, o_D\}}, \{o_E, o_F\}$ remain landmarks

Incremental Computation (Example)



- Discovered landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Landmarks after application of o_B
  - $\{o_A\}, \cancel{\{o_B, o_C, o_D\}}, \{o_E, o_F\}$ remain landmarks
  - Newly discovered landmark: $\{o_C, o_D\}$

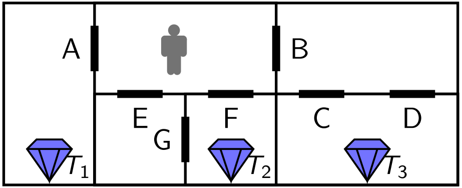
Incremental Computation

- Successor generated by applying operator o
  - All landmarks not containing o are landmarks in successor
  - Discharge landmarks containing o
    - Return landmark's costs to *remaining cost*
    - Can change $h^{\max}$ value
  - Start LM-cut algorithm with set of known landmarks

Theorem

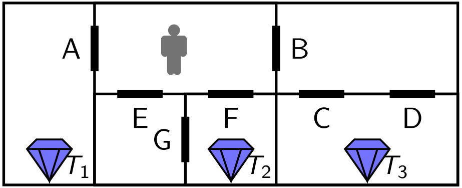
The LM-cut algorithm discovers a new landmark
if the $h^{\max}$ cost of the successor increases.

Incremental Computation (Example)



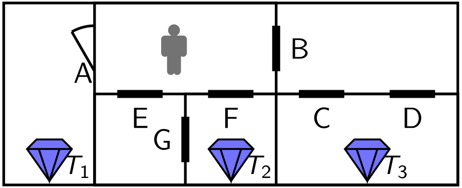
- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$

Incremental Computation (Example)



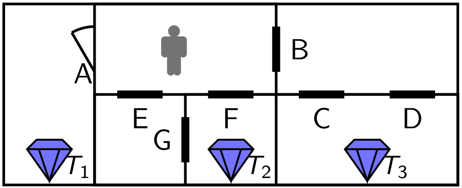
- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door A
- Open door B
- Open door E
- Open door F

Incremental Computation (Example)



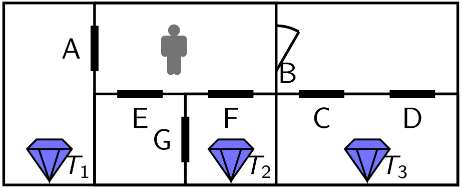
- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door A
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door B
- Open door E
- Open door F

Incremental Computation (Example)



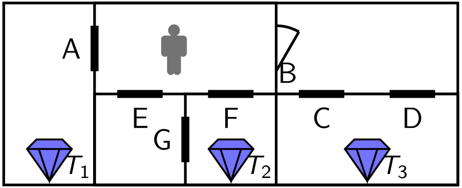
- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door A
  - $\cancel{\{o_A\}}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door B
- Open door E
- Open door F

Incremental Computation (Example)



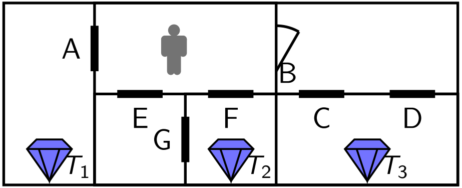
- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door A
  - $\cancel{\{o_A\}}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door B
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door E
- Open door F

Incremental Computation (Example)



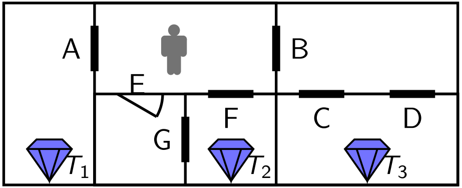
- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door A
  - $\cancel{\{o_A\}}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door B
  - $\{o_A\}, \cancel{\{o_B, o_C, o_D\}}, \{o_E, o_F\}$
- Open door E
- Open door F

Incremental Computation (Example)



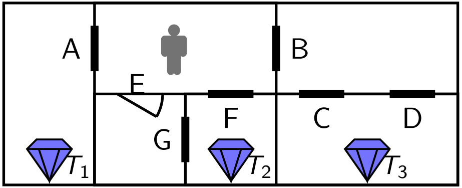
- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door A
  - $\cancel{\{o_A\}}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door B
  - $\{o_A\}, \cancel{\{o_B, o_C, o_D\}}, \{o_E, o_F\}, \{o_C, o_D\}$
- Open door E
- Open door F

Incremental Computation (Example)



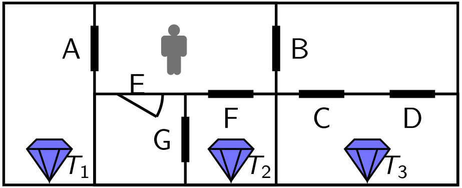
- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door A
  - $\cancel{\{o_A\}}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door B
  - $\{o_A\}, \cancel{\{o_B, o_C, o_D\}}, \{o_E, o_F\}, \{o_C, o_D\}$
- Open door E
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door F

Incremental Computation (Example)



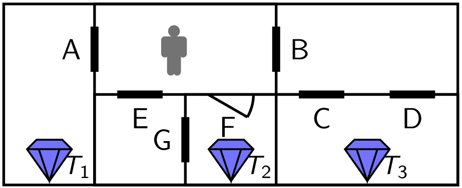
- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door A
  - $\cancel{\{o_A\}}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door B
  - $\{o_A\}, \cancel{\{o_B, o_C, o_D\}}, \{o_E, o_F\}, \{o_C, o_D\}$
- Open door E
  - $\{o_A\}, \{o_B, o_C, o_D\}, \cancel{\{o_E, o_F\}}$
- Open door F

Incremental Computation (Example)



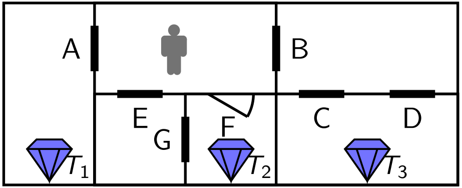
- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door A
  - $\cancel{\{o_A\}}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door B
  - $\{o_A\}, \cancel{\{o_B, o_C, o_D\}}, \{o_E, o_F\}, \{o_C, o_D\}$
- Open door E
  - $\{o_A\}, \{o_B, o_C, o_D\}, \cancel{\{o_E, o_F\}}, \{o_F, o_G\}$
- Open door F

Incremental Computation (Example)



- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door A
  - $\cancel{\{o_A\}}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door B
  - $\{o_A\}, \cancel{\{o_B, o_C, o_D\}}, \{o_E, o_F\}, \{o_C, o_D\}$
- Open door E
  - $\{o_A\}, \{o_B, o_C, o_D\}, \cancel{\{o_E, o_F\}}, \{o_F, o_G\}$
- Open door F
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$

Incremental Computation (Example)



- Landmarks in this state
  - $\{o_A\}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door A
  - $\cancel{\{o_A\}}, \{o_B, o_C, o_D\}, \{o_E, o_F\}$
- Open door B
  - $\{o_A\}, \cancel{\{o_B, o_C, o_D\}}, \{o_E, o_F\}, \{o_C, o_D\}$
- Open door E
  - $\{o_A\}, \{o_B, o_C, o_D\}, \cancel{\{o_E, o_F\}}, \{o_F, o_G\}$
- Open door F
  - $\{o_A\}, \{o_B, o_C, o_D\}, \cancel{\{o_E, o_F\}}$

Content

Benchmarks

- Evaluation
  - 1396 tasks in 44 domains
  - Time limit: 30 min
  - Memory limit: 2 GB
- Measured
  - Coverage (number of solved tasks)
  - Search time
  - Failure reason (timeout or out of memory)

Basic Results

- First idea ($h^{\text{iLM-cut}}$)
  - Regular A* search
  - Store landmarks for all search nodes
  - (Side note: This makes the heuristic consistent)
- Compare with $A^*/h^{\text{LM-cut}}$
  - Speed-ups by up to an order of magnitude
  - More tasks running out of memory
    - Increased from 31 to 526

Coverage

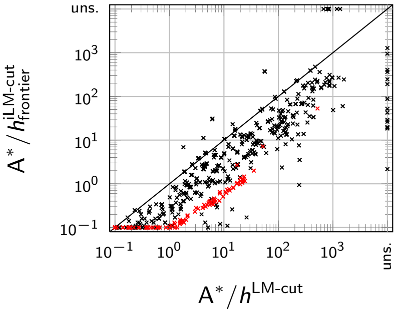
$h^{\text{LM-cut}}$	$h^{\text{iLM-cut}}$
757	762

Removing Landmarks

- Stored landmarks take up too much memory
  - Information can be removed at any time
  - Recompute missing values non-incrementally
- Easy fix ($h^{\text{iLM-cut}}_{\text{frontier}}$)
  - Remove information for closed nodes
  - Only needed again if the node is reopened

Results

- Runtime reduction over baseline
  - 77% (geometric mean)
  - 93% (miconic ×)
- Coverage increased by another 4 tasks



Coverage

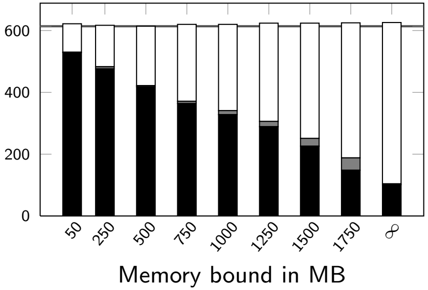
$h^{\text{LM-cut}}$	$h^{\text{iLM-cut}}$	$h^{\text{iLM-cut}}_{\text{frontier}}$
757	762	766

Fixed Memory Bounds

- Why stop because of exhausted memory at all?
  - Can always free up memory
  - Remove stored landmarks
- Fixed memory bound ($h^{\text{iLM-cut}}_{\text{bound}}$)
  - Keep track of used memory
  - Remove half of the stored landmarks when hitting the bound
- Dynamic memory bound
  - Technical problems
    - Measure memory pressure and memory requirements accurately
  - Results estimated from fix bounds

Results

- Improvements even for small bounds (50 MB)
- Increasing limit
  - Less timeouts (■)
  - Memory exhausted more often (□)
- Sweet spot for 500 MB



Coverage

$h^{\text{LM-cut}}$	$h^{\text{iLM-cut}}$	$h^{\text{iLM-cut}}_{\text{frontier}}$	$h^{\text{iLM-cut}}_{\text{500 MB}}$	$h^{\text{iLM-cut}}_{\text{dynamic}}$
757	762	766	778	*786*

Local Incremental Computation

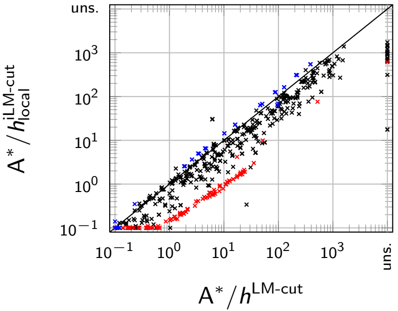
- More generated nodes than expanded nodes
  - Factor 8 in the geometric mean
  - Save work during node generation
  - Additional work during node expansion amortized
- Local incremental computation ($h^{\text{iLM-cut}}_{\text{local}}$)
  - Recompute landmarks for parent node
  - Incremental computation for child nodes
  - Minimal memory overhead

LM-cut Computations

	$h^{\text{LM-cut}}$	$h^{\text{iLM-cut}}_{\text{local}}$
Node generation	Non-incremental	Incremental
Node expansion	0	Non-incremental

Results

- Runtime changes
  - −49% (geometric mean)
  - −89% (miconic ×)
  - +40% (openstacks ×)
- Bad performance of openstacks
  - $h^{\text{LM-cut}}(s) = 1$ for most states s
  - Many 0-cost operators



Coverage

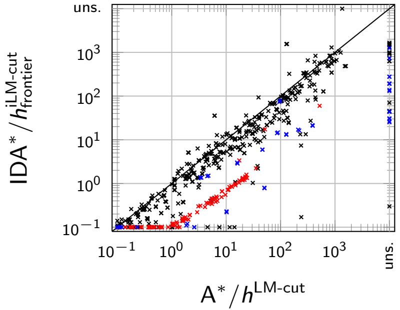
$h^{\text{LM-cut}}$	$h^{\text{iLM-cut}}$	$h^{\text{iLM-cut}}_{\text{frontier}}$	$h^{\text{iLM-cut}}_{\text{500 MB}}$	$h^{\text{iLM-cut}}_{\text{dynamic}}$	$h^{\text{iLM-cut}}_{\text{local}}$
757	762	766	778	*786*	783

IDA*

- IDA* Search
  - Classical solution to memory issues with A*
- Here: Sufficient memory for all search nodes
  - Use unlimited transposition table
  - Perfect duplicate detection
  - Store heuristic values for inner nodes
- Remaining advantage over A*
  - Depth-first expansion order
  - Only store landmarks for current branch

Results

- Best Coverage result
- Skewed by openstacks (×)
  - Better tie-breaking with depth-first order
  - Deeper nodes are preferred
  - Explains 14 tasks



Coverage

A*						IDA*
$h^{\text{LM-cut}}$	$h^{\text{iLM-cut}}$	$h^{\text{iLM-cut}}_{\text{frontier}}$	$h^{\text{iLM-cut}}_{\text{500 MB}}$	$h^{\text{iLM-cut}}_{\text{dynamic}}$	$h^{\text{iLM-cut}}_{\text{local}}$	$h^{\text{iLM-cut}}_{\text{frontier}}$
757	762	766	778	*786*	783	790

Content

Conclusion

- Incremental computation of LM-cut for STRIPS planning
  - Much faster
  - Higher memory requirements
- Dealing with memory limitations
  - Local computation
  - Fixed bounds
  - IDA*
- Not limited to LM-cut
- Necessary conditions
  - Incremental computation is faster
  - Missing information can be computed non-incrementally

Thank you for your attention!
Any questions?