Master's Thesis Presentation

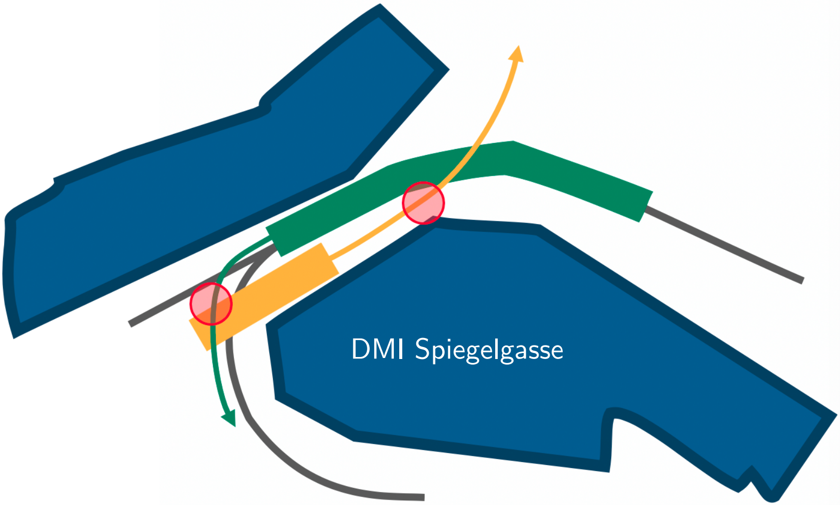
Generalization of Cycle-Covering Heuristics

Clemens Büchner

Department of Mathematics and Computer Science
University of Basel

May 14, 2020

University of Basel


DMI Spiegelgasse

Outline

1. Background
2. Cycle-covering heuristic
3. Experimental results

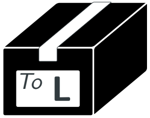
Background

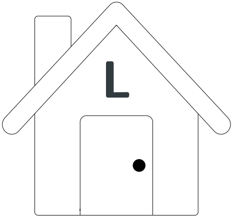
Planning

Simplistic world model for specific problem purposes:

Definition (planning task)

A planning task is a 4-tuple $\mathcal{T} = \langle \mathcal{V}, s_0, \gamma, \mathcal{A} \rangle$ with





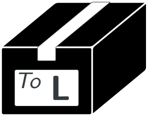
Planning

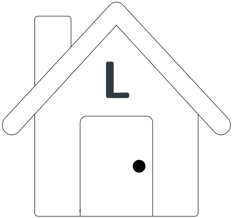
Simplistic world model for specific problem purposes:

Definition (planning task)

A planning task is a 4-tuple $\mathcal{T} = \langle \mathcal{V}, s_0, \gamma, \mathcal{A} \rangle$ with

- a finite set of variables $\mathcal{V}$ to describe each world state,





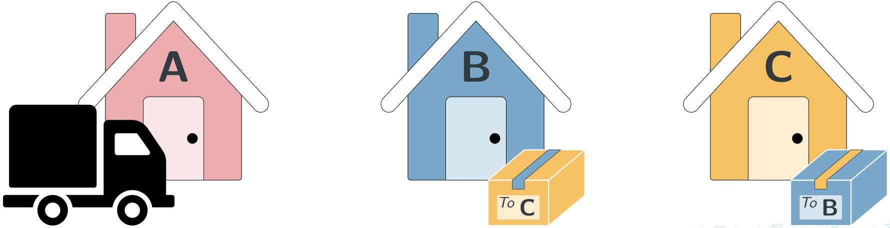
Planning

Simplistic world model for specific problem purposes:

Definition (planning task)

A planning task is a 4-tuple $\mathcal{T} = \langle \mathcal{V}, s_0, \gamma, \mathcal{A} \rangle$ with

- a finite set of variables $\mathcal{V}$ to describe each world state,
- the initial state s_0,



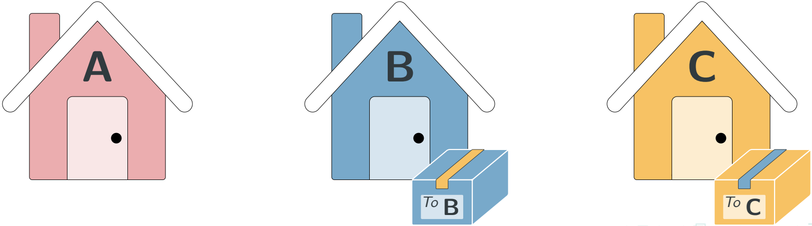
Planning

Simplistic world model for specific problem purposes:

Definition (planning task)

A planning task is a 4-tuple $\mathcal{T} = \langle \mathcal{V}, s_0, \gamma, \mathcal{A} \rangle$ with

- a finite set of variables $\mathcal{V}$ to describe each world state,
- the initial state s_0,
- the goal condition γ, and



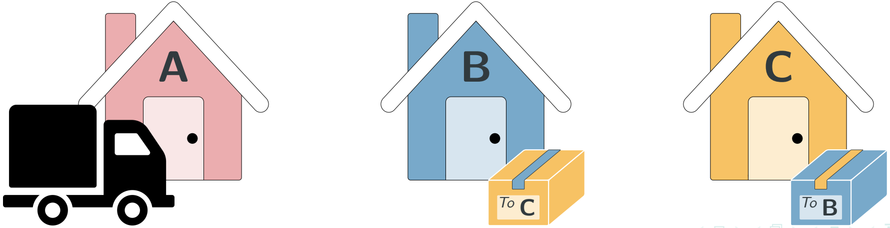
Planning

Simplistic world model for specific problem purposes:

Definition (planning task)

A planning task is a 4-tuple $\mathcal{T} = \langle \mathcal{V}, s_0, \gamma, \mathcal{A} \rangle$ with

- a finite set of variables $\mathcal{V}$ to describe each world state,
- the initial state s_0,
- the goal condition γ, and
- a finite set of actions $\mathcal{A}$ to transition between states.



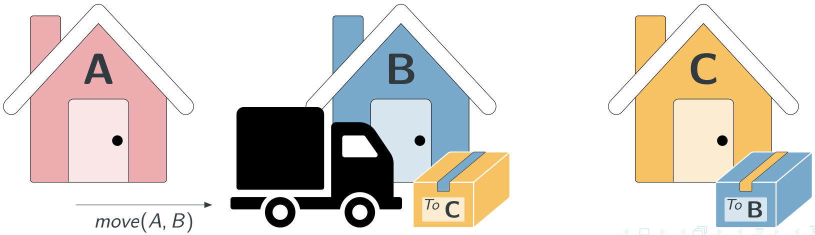
Planning

Simplistic world model for specific problem purposes:

Definition (planning task)

A planning task is a 4-tuple $\mathcal{T} = \langle \mathcal{V}, s_0, \gamma, \mathcal{A} \rangle$ with

- a finite set of variables $\mathcal{V}$ to describe each world state,
- the initial state s_0,
- the goal condition γ, and
- a finite set of actions $\mathcal{A}$ to transition between states.



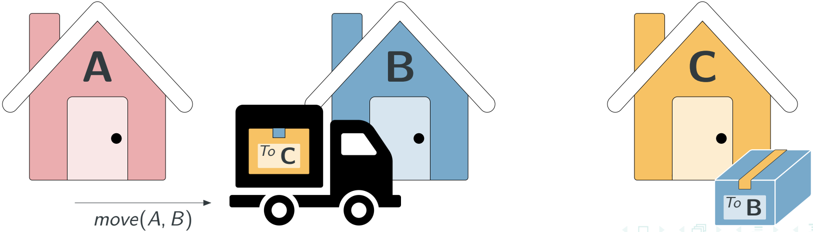
Planning

Simplistic world model for specific problem purposes:

Definition (planning task)

A planning task is a 4-tuple $\mathcal{T} = \langle \mathcal{V}, s_0, \gamma, \mathcal{A} \rangle$ with

- a finite set of variables $\mathcal{V}$ to describe each world state,
- the initial state s_0,
- the goal condition γ, and
- a finite set of actions $\mathcal{A}$ to transition between states.



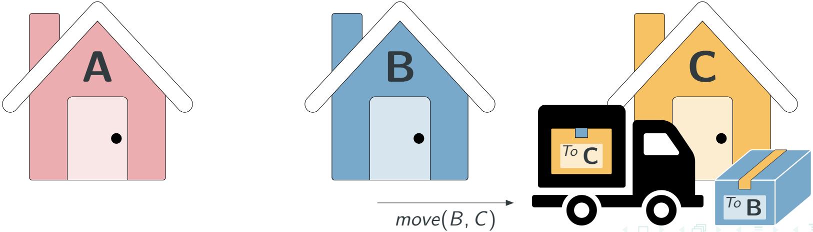
Planning

Simplistic world model for specific problem purposes:

Definition (planning task)

A planning task is a 4-tuple $\mathcal{T} = \langle \mathcal{V}, s_0, \gamma, \mathcal{A} \rangle$ with

- a finite set of variables $\mathcal{V}$ to describe each world state,
- the initial state s_0,
- the goal condition γ, and
- a finite set of actions $\mathcal{A}$ to transition between states.



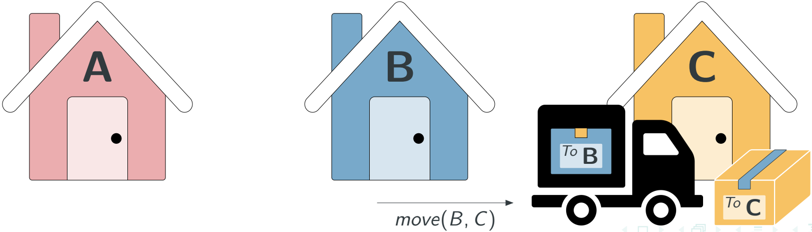
Planning

Simplistic world model for specific problem purposes:

Definition (planning task)

A planning task is a 4-tuple $\mathcal{T} = \langle \mathcal{V}, s_0, \gamma, \mathcal{A} \rangle$ with

- a finite set of variables $\mathcal{V}$ to describe each world state,
- the initial state s_0,
- the goal condition γ, and
- a finite set of actions $\mathcal{A}$ to transition between states.



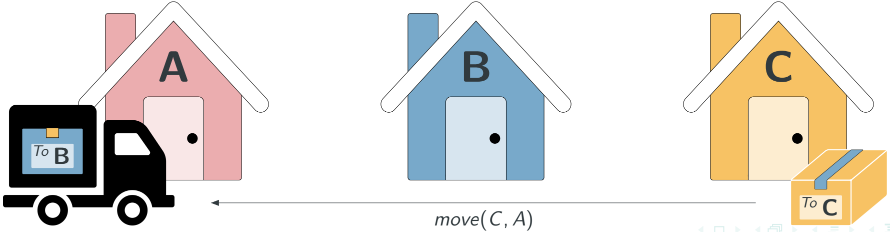
Planning

Simplistic world model for specific problem purposes:

Definition (planning task)

A planning task is a 4-tuple $\mathcal{T} = \langle \mathcal{V}, s_0, \gamma, \mathcal{A} \rangle$ with

- a finite set of variables $\mathcal{V}$ to describe each world state,
- the initial state s_0,
- the goal condition γ, and
- a finite set of actions $\mathcal{A}$ to transition between states.



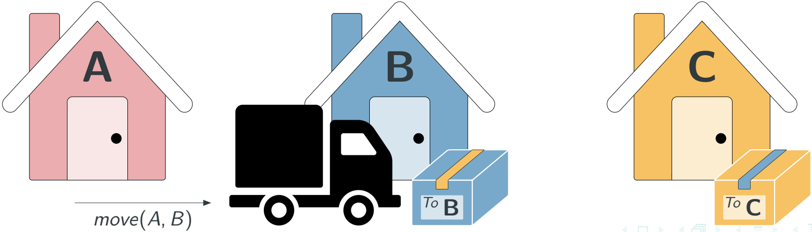
Planning

Simplistic world model for specific problem purposes:

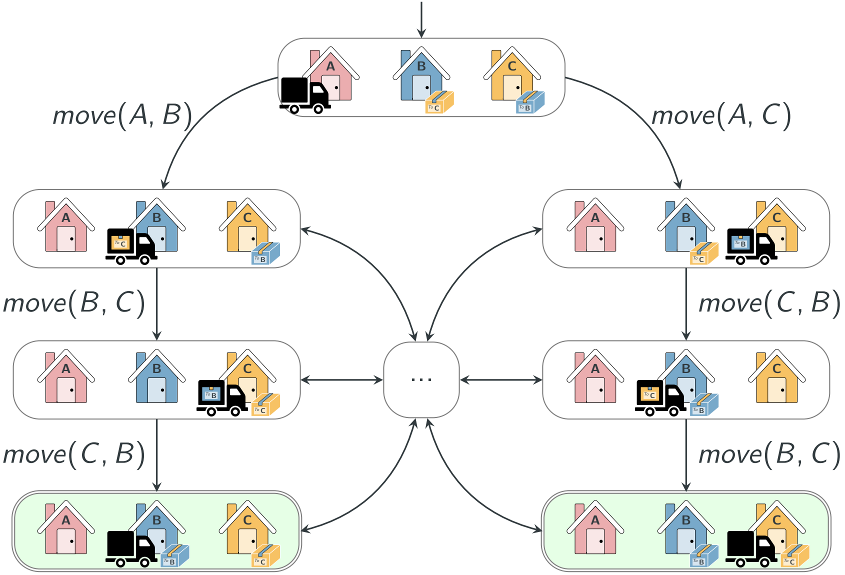
Definition (planning task)

A planning task is a 4-tuple $\mathcal{T} = \langle \mathcal{V}, s_0, \gamma, \mathcal{A} \rangle$ with

- a finite set of variables $\mathcal{V}$ to describe each world state,
- the initial state s_0,
- the goal condition γ, and
- a finite set of actions $\mathcal{A}$ to transition between states.



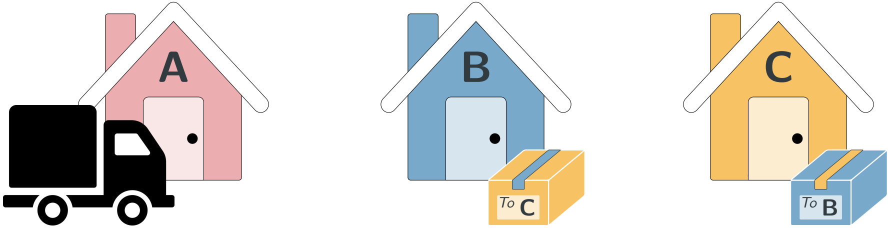
State Space and Heuristic Search



Landmarks

Properties that must hold along all plans:

- move to B to pick up yellow package
- move to C to deliver yellow package
- the blue package induces the same landmarks

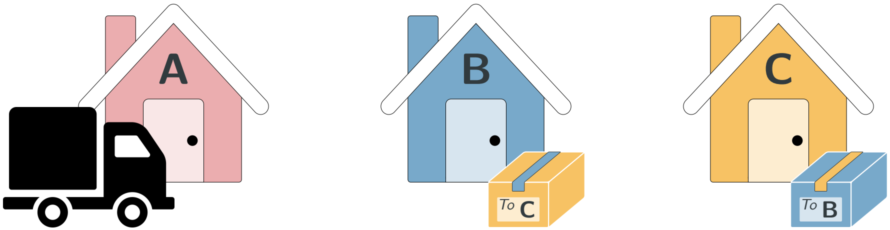


Landmarks

Properties that must hold along all plans:

Definition (disjunctive action landmark)

Let $\mathcal{T} = \langle \mathcal{V}, s_0, G, \mathcal{A} \rangle$ be a planning task and let s be a state of $\mathcal{T}$.
A disjunctive action landmark of s is a non-empty set of actions $\ell \subseteq \mathcal{A}$ such that every s-plan contains an action $a \in \ell$.



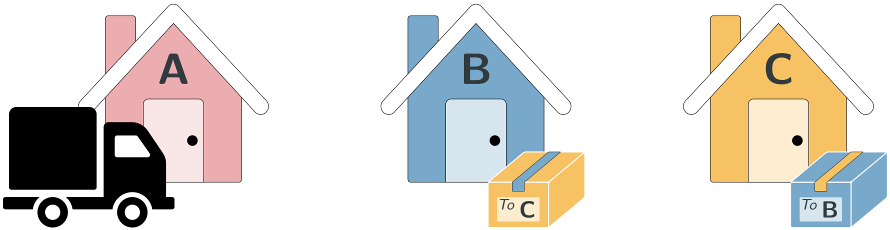
Landmarks

Properties that must hold along all plans:

Definition (disjunctive action landmark)

Let $\mathcal{T} = \langle \mathcal{V}, s_0, G, \mathcal{A} \rangle$ be a planning task and let s be a state of $\mathcal{T}$.
A disjunctive action landmark of s is a non-empty set of actions $\ell \subseteq \mathcal{A}$ such that every s-plan contains an action $a \in \ell$.

- in example: $\{move(A, B), move(C, B)\}$ and $\{move(A, C), move(B, C)\}$



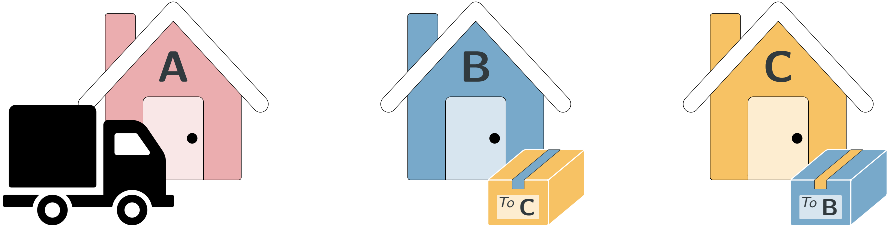
Landmarks

Properties that must hold along all plans:

Definition (disjunctive action landmark)

Let $\mathcal{T} = \langle \mathcal{V}, s_0, G, \mathcal{A} \rangle$ be a planning task and let s be a state of $\mathcal{T}$.
A disjunctive action landmark of s is a non-empty set of actions $\ell \subseteq \mathcal{A}$ such that every s-plan contains an action $a \in \ell$.

- in example: $\{move(A, B), move(C, B)\}$ and $\{move(A, C), move(B, C)\}$
- landmark generation is not the topic of this thesis



Landmark Heuristic h^{LM}

- one action from each landmark must be part of every plan
- minimum hitting set approach
  - cheapest set of actions that hits each landmark
- solve with linear programming

$$\begin{aligned} \min \sum_{a \in \mathcal{A}} \mathrm{Y}_a \cdot cost(a) \quad & \text{s.t.} \\ \mathrm{Y}_a \geq 0 \quad & \text{for all } a \in \mathcal{A} \text{ and} \\ \sum_{a \in \ell} \mathrm{Y}_a \geq 1 \quad & \text{for all } \ell \in \mathcal{L} \end{aligned}$$

- this corresponds to the operator-counting framework
- use objective value as heuristic estimate

Landmark Orderings

Landmark orderings denote dependencies between landmarks.

- natural orderings must hold along all plans
  - e.g., impossible to *unload* package before *loaded*
- reasonable orderings are rather "suggestions"
  - e.g., *move* to the package's origin before its destination

Landmark Orderings

Landmark orderings denote dependencies between landmarks.

- natural orderings must hold along all plans
  - e.g., impossible to *unload* package before *loaded*
- reasonable orderings are rather "suggestions"
  - e.g., *move* to the package's origin before its destination

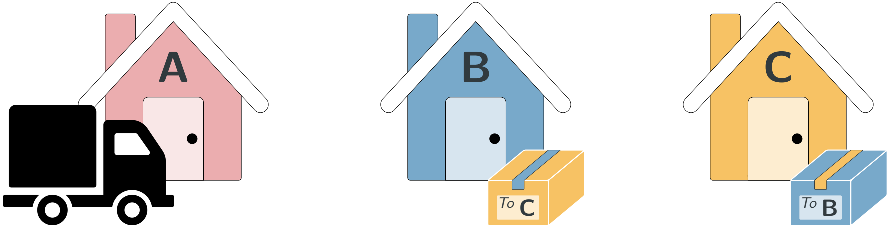
Represent landmarks and orderings in landmark graphs:

$$
\begin{tikzcd}
\{move(A, B), move(C, B)\} \arrow[r, bend left, "r"] & \{move(A, C), move(B, C)\} \arrow[l, bend left, "r"]
\end{tikzcd}
$$

Cycle-Covering

Valuable Information in Landmark Graphs

- cyclical dependencies between landmarks
- sub-goal must be achieved multiple times to resolve cycle
- one landmark per cycle necessary twice in every plan
- again minimum hitting set problem for cycles



Cycle-Covering Heuristic h^{cycle}

Extending the landmark heuristic with cycle constraints:

$$\begin{aligned} \min \sum_{a \in \mathcal{A}} Y_a \cdot cost(a) &\quad \text{s.t.} \\ Y_a \geq 0 &\quad \text{for all } a \in \mathcal{A} \text{ and} \\ \sum_{a \in \ell} Y_a \geq 1 &\quad \text{for all } \ell \in \mathcal{L} \text{ and} \\ \sum_{\ell \in c} \sum_{a \in \ell} Y_a \geq |c| + 1 &\quad \text{for all } c \in \mathcal{C} \end{aligned}$$

with $\mathcal{C}$ the set of cycles in the landmark graph

Heuristics Applied to Running Example

$$\begin{tikzcd} \{move(A,B), move(C,B)\} \arrow[r, bend left, "r"] & \{move(A,C), move(B,C)\} \arrow[l, bend left, "r"] \end{tikzcd}$$

$$\min Y_{A\to B} + Y_{C\to B} + Y_{A\to C} + Y_{B\to C} \quad \text{s.t.}$$

$$\begin{aligned} Y_{A\to B} + Y_{C\to B} \qquad\qquad & \geq 1 \\ Y_{A\to C} + Y_{B\to C} & \geq 1 \end{aligned}$$

Heuristics Applied to Running Example

$$\begin{tikzcd}
\{move(A,B), move(C,B)\} \arrow[r, bend left, "\mathrm{r}"] & \{move(A,C), move(B,C)\} \arrow[l, bend left, "\mathrm{r}"]
\end{tikzcd}$$

$$\min \mathrm{Y}_{A\to B} + \mathrm{Y}_{C\to B} + \mathrm{Y}_{A\to C} + \mathrm{Y}_{B\to C} \quad \text{s.t.}$$

$$\begin{aligned}
\mathrm{Y}_{A\to B} + \mathrm{Y}_{C\to B} \qquad\qquad\qquad\qquad &\geq 1 \\
\mathrm{Y}_{A\to C} + \mathrm{Y}_{B\to C} &\geq 1
\end{aligned}$$

- $h^{\mathrm{LM}}(s_0) = 2 \qquad (\mathrm{Y}_{A\to B} = 1, \mathrm{Y}_{B\to C} = 1)$

Heuristics Applied to Running Example

$$\begin{tikzcd} \{move(A,B), move(C,B)\} \arrow[r, bend left, "r"] & \{move(A,C), move(B,C)\} \arrow[l, bend left, "r"] \end{tikzcd}$$

$$\min \mathrm{Y}_{A\rightarrow B} + \mathrm{Y}_{C\rightarrow B} + \mathrm{Y}_{A\rightarrow C} + \mathrm{Y}_{B\rightarrow C} \quad \text{s.t.}$$

$$\begin{aligned} \mathrm{Y}_{A\rightarrow B} + \mathrm{Y}_{C\rightarrow B} \qquad\qquad\qquad\qquad &\geq 1 \\ \mathrm{Y}_{A\rightarrow C} + \mathrm{Y}_{B\rightarrow C} &\geq 1 \\ \mathrm{Y}_{A\rightarrow B} + \mathrm{Y}_{C\rightarrow B} + \mathrm{Y}_{A\rightarrow C} + \mathrm{Y}_{B\rightarrow C} &\geq 3 \end{aligned}$$

- $h^{\mathrm{LM}}(s_0) = 2 \qquad (\mathrm{Y}_{A\rightarrow B} = 1, \mathrm{Y}_{B\rightarrow C} = 1)$

Heuristics Applied to Running Example

$$\begin{tikzcd}
\{move(A,B), move(C,B)\} \arrow[r, bend left, "r"] & \{move(A,C), move(B,C)\} \arrow[l, bend left, "r"]
\end{tikzcd}$$

$$\min \mathrm{Y}_{A\to B} + \mathrm{Y}_{C\to B} + \mathrm{Y}_{A\to C} + \mathrm{Y}_{B\to C} \quad \text{s.t.}$$

$$\begin{aligned}
\mathrm{Y}_{A\to B} + \mathrm{Y}_{C\to B} \qquad\qquad\qquad\qquad &\geq 1\\
\mathrm{Y}_{A\to C} + \mathrm{Y}_{B\to C} &\geq 1\\
\mathrm{Y}_{A\to B} + \mathrm{Y}_{C\to B} + \mathrm{Y}_{A\to C} + \mathrm{Y}_{B\to C} &\geq 3
\end{aligned}$$

- $h^{\mathrm{LM}}(s_0) = 2 \quad (\mathrm{Y}_{A\to B} = 1, \mathrm{Y}_{B\to C} = 1)$
- $h^{\mathrm{cycle}}(s_0) = 3 \quad (\mathrm{Y}_{A\to B} = 1, \mathrm{Y}_{B\to C} = 1, \mathrm{Y}_{C\to B} = 1)$

Ordering-Aware Cycle-Covering Heuristic h^{ord}

- natural orderings are acyclic by definition
- candidates for resolving cycles must have an incoming reasonable ordering

$$
\begin{aligned}
\min Y_a + Y_b + Y_c \quad & \text{s.t.} \\
Y_a \qquad\qquad\quad & \geq 1 \\
Y_b \qquad\quad & \geq 1 \\
Y_c & \geq 1 \\
Y_a + Y_b \qquad\quad & \geq 3 \\
Y_b + Y_c & \geq 3
\end{aligned}
$$

$$
\begin{tikzcd}
\{a\} \arrow[d, bend left, "r"] \\
\{b\} \arrow[u, bend left, "r"] \arrow[d, bend left, "r"] \\
\{c\} \arrow[u, bend left, "n"]
\end{tikzcd}
$$

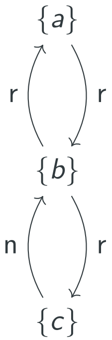
- $h^{cycle}(s) = 4 \quad (Y_a = 1, Y_b = 2, Y_c = 1)$

Ordering-Aware Cycle-Covering Heuristic h^{ord}

- natural orderings are acyclic by definition
- candidates for resolving cycles must have an incoming reasonable ordering

$$\min Y_a + Y_b + Y_c \quad \text{s.t.}$$

$$\begin{aligned}
Y_a \quad\quad\quad\quad\quad &\geq 1 \\
Y_b \quad\quad\quad &\geq 1 \\
Y_c &\geq 1 \\
Y_a + Y_b \quad\quad\quad &\geq 3 \\
\cancel{Y_b +} Y_c &\geq \cancel{3}\ 2
\end{aligned}$$



- $h^{cycle}(s) = 4 \quad (Y_a = 1, Y_b = 2, Y_c = 1)$
- $h^{ord}(s) = 5 \quad (Y_a = 1, Y_b = 2, Y_c = 2)$

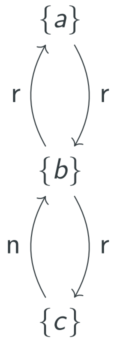
Ordering-Aware Cycle-Covering Heuristic h^{ord}

- natural orderings are acyclic by definition
- candidates for resolving cycles must have an incoming reasonable ordering

Ordering-aware cycle-covering heuristic:

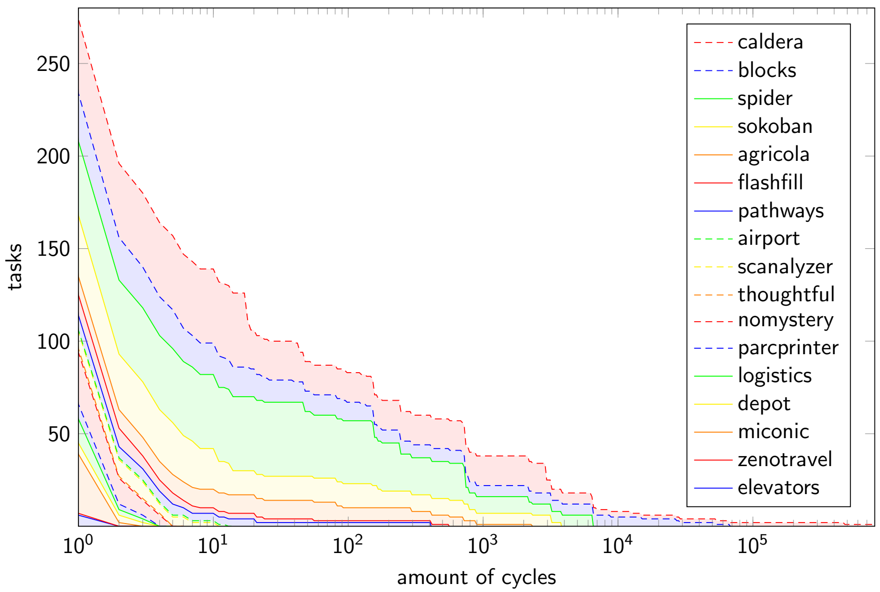
$$\begin{aligned} \min \sum_{a \in \mathcal{A}} Y_a \cdot cost(a) \quad & \text{s.t.} \\ Y_a \geq 0 \quad & \text{for all } a \in \mathcal{A} \text{ and} \\ \sum_{a \in \ell} Y_a \geq 1 \quad & \text{for all } \ell \in \mathcal{L} \text{ and} \\ \sum_{\ell \in c_r} \sum_{a \in \ell} Y_a \geq |c_r| + 1 \quad & \text{for all } c \in \mathcal{C} \end{aligned}$$

with $c_r \subseteq c$ the set of landmarks with incoming reasonable orderings

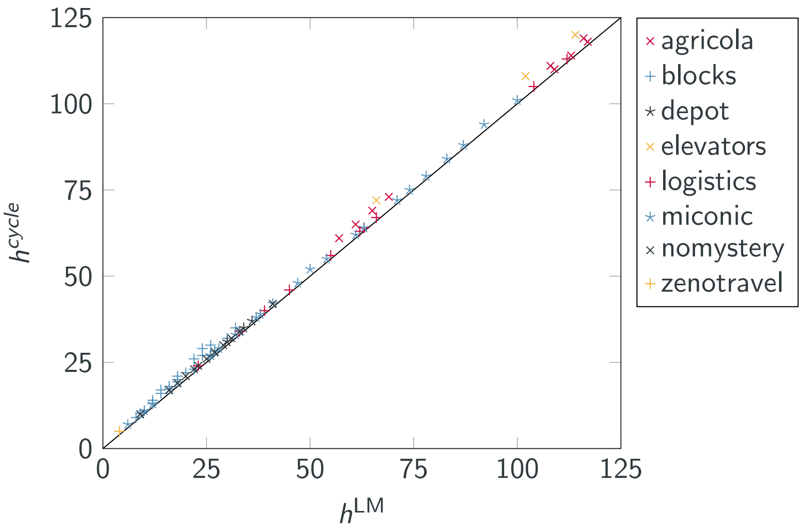


Experimental Evaluation

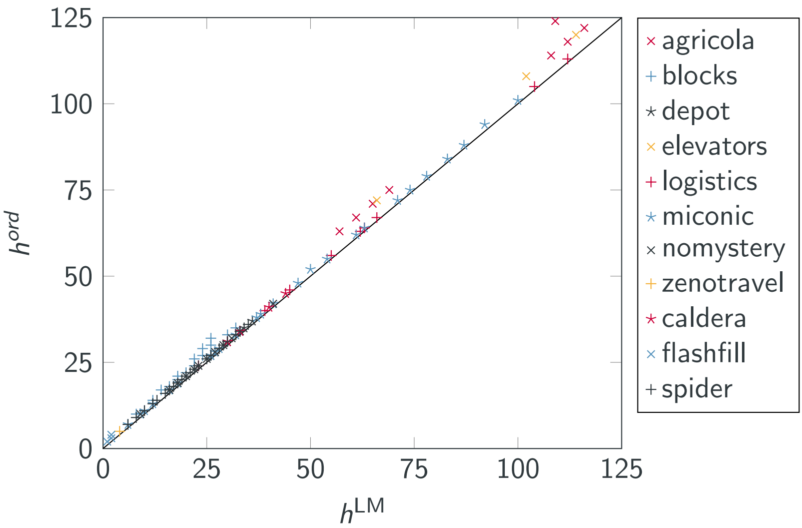
Cycles in Landmark Graphs



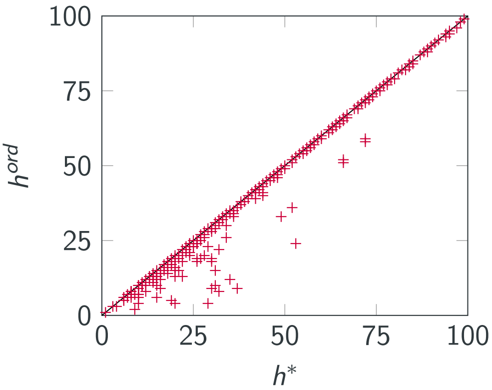
Initial h-Value – h^{LM} vs. h^{cycle}



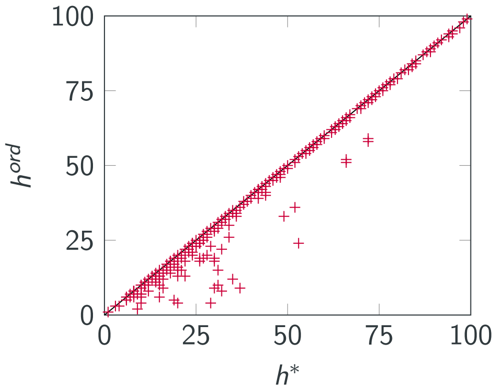
Initial h-Value – h^{LM} vs. h^{ord}

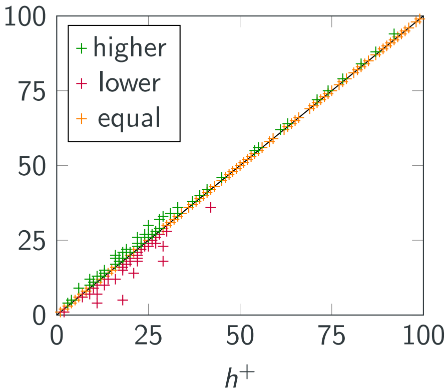


Aiming for Optimality



Aiming for Optimality





Planning with the Cycle-Covering Heuristic

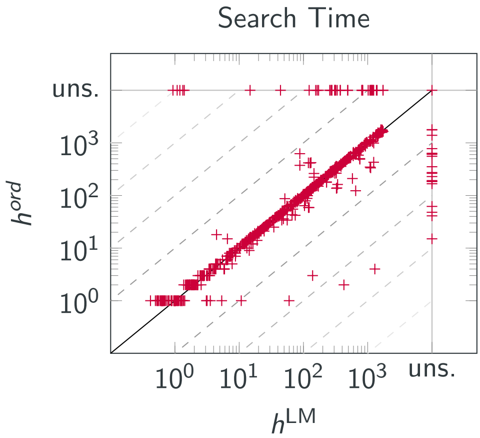
Coverage barely affected

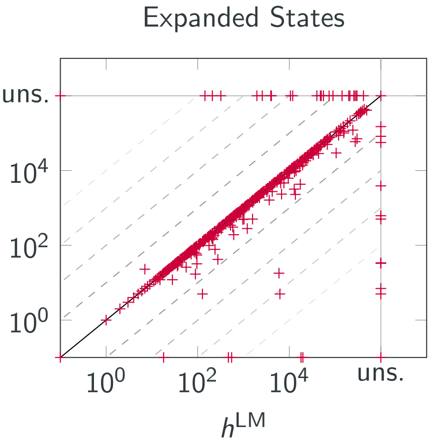
- tasks with many cycles prone to exceed memory limit
- increased complexity of optimization problems

Overall results are inconclusive and vary depending on several factors:

- Which landmark generator is used?
  - various options in Fast Downward
- How to update landmarks in encountered states?
  - recomputing vs. tracking based on previous state

Planning with the Cycle-Covering Heuristic





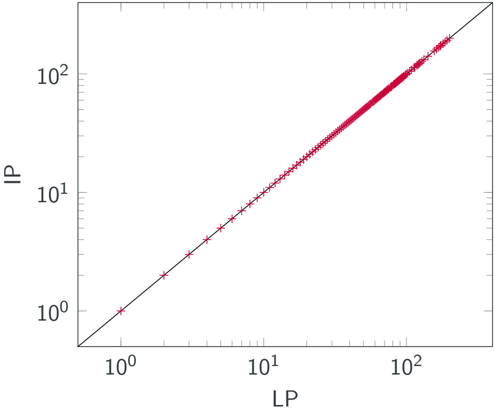
Summary

- cyclical dependencies between landmarks contain valuable information
- cycle-covering heuristic dominates minimum hitting set landmark heuristic for the same landmark graph
- considering ordering types improves cycle-covering heuristic
- increased heuristic accuracy in practice
- does not (yet) pay off in coverage

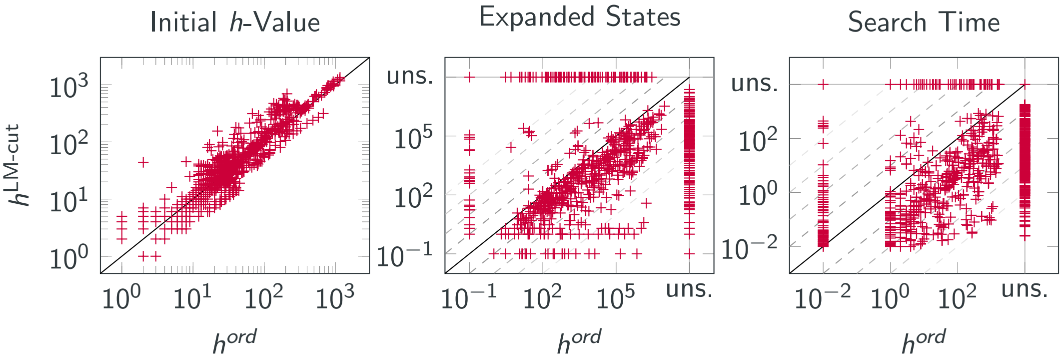
Appendix

Initial h-Values – LP vs. IP

- IP solutions identical to LP-relaxation
- possible explanation: totally unimodular matrices
  - all squared sub-matrices have $\det \in \{-1, 0, 1\}$
  - LP solutions are integral
- but not generally the case
  - Counterexample with LP $\neq$ IP



Comparison to LM-Cut

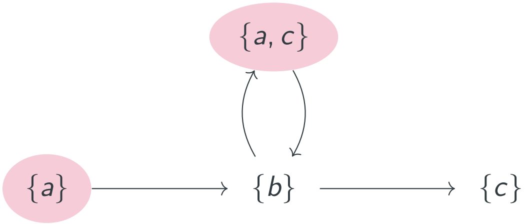


Decomposing Cycle-Covering from Landmark Hitting Set

$$
\begin{tikzcd}
 & \{a, c\} \arrow[d, bend left] & \\
\{a\} \arrow[r] & \{b\} \arrow[r] \arrow[u, bend left] & \{c\}
\end{tikzcd}
$$

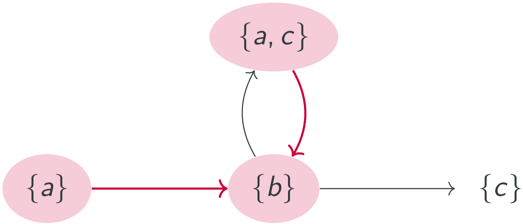
- Minimum landmark hitting set: 3, minimum cycle hitting set: $1 \Rightarrow h = 4$

Decomposing Cycle-Covering from Landmark Hitting Set



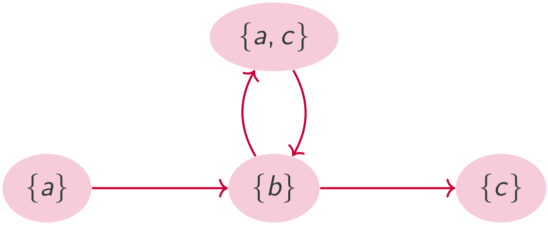
- Minimum landmark hitting set: 3, minimum cycle hitting set: 1 $\Rightarrow h = 4$
- Optimal plan: $\langle a, b, c \rangle \Rightarrow h^* = 3$

Decomposing Cycle-Covering from Landmark Hitting Set



- Minimum landmark hitting set: 3, minimum cycle hitting set: 1 $\Rightarrow h = 4$
- Optimal plan: $\langle a, b, c \rangle \Rightarrow h^* = 3$

Decomposing Cycle-Covering from Landmark Hitting Set



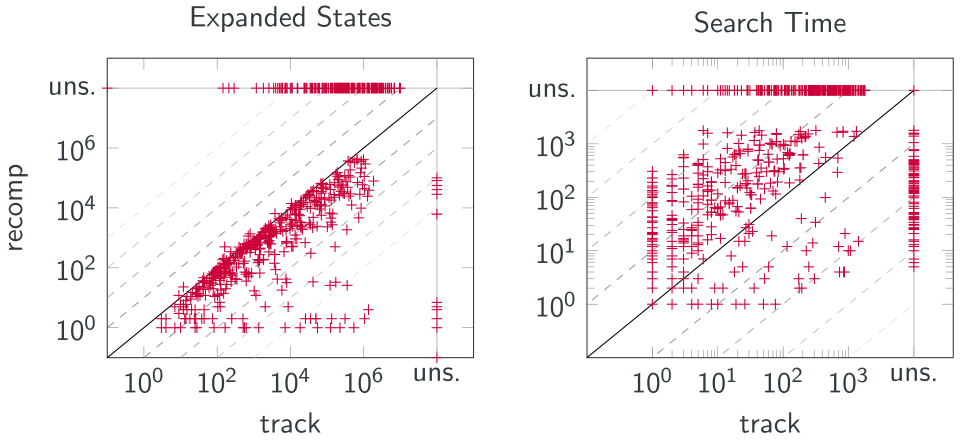
- Minimum landmark hitting set: 3, minimum cycle hitting set: 1 $\Rightarrow h = 4$
- Optimal plan: $\langle a, b, c \rangle \Rightarrow h^* = 3$

Recomputing vs. Tracking Landmarks

Update landmarks in every state:

- always recompute
  - \+ potentially find new landmarks
  - − might be time-consuming
- or track along paths
  - \+ compute landmarks only once in the beginning
  - − applying actions can only decrease heuristic estimates
  - ± heuristic is path-dependent

Recomputing vs. Tracking Landmarks



Coverage Results

- 870 planning tasks from domains with cyclical initial states

	$\mathrm{LM}^{\mathrm{RHW}}$		LM^{h^m}	
	recomp	track	recomp	track
h^{LM}	342	308	222	298
h^{cycle}	336	305	228	305
h^{ord}	340	306	231	308

Coverage Results

- 870 planning tasks from domains with cyclical initial states

	LM^{RHW}		LM^{h^m}	
	recomp	track	recomp	track
h^{LM}	342	308	222	298
h^{cycle}	336	305	228	305
h^{ord}	340	306	231	308

- success of recomputing vs. tracking depends on landmark generator

Coverage Results

- 870 planning tasks from domains with cyclical initial states

	$\mathrm{LM}^{\mathrm{RHW}}$		LM^{h^m}	
	recomp	track	recomp	track
h^{LM}	342	308	222	298
h^{cycle}	336	305	228	305
h^{ord}	340	306	231	308

- success of recomputing vs. tracking depends on landmark generator
- considering cycles is not always beneficial
  - memory is a limitation
  - optimization problems have increased complexity