Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)

Exploiting Cyclic Dependencies in Landmark Heuristics

Clemens Büchner, Thomas Keller, Malte Helmert
University of Basel, Switzerland
{clemens.buechner, tho.keller, malte.helmert}@unibas.ch

Abstract

Landmarks of a planning task denote properties that must be satisfied by all plans. Existing landmark heuristics exploit that each landmark must be achieved at least once. However, if the orderings between the landmarks induce cyclic dependencies, one of the landmarks in each cycle must be achieved an additional time. We propose two novel heuristics for cost-optimal planning that consider cyclic dependencies between landmarks in addition to the cost for achieving all landmarks once.

We show that our heuristics dominate the minimum hitting set solution over any set of landmarks as well as h^+ if all delete-relaxation landmarks are considered. An experimental evaluation on benchmarks from the International Planning Competition shows that exploiting cyclic dependencies can lead to improved heuristics.

Introduction

Properties that are shared by every solution of a classical planning task are called *landmarks*. Early work computed landmarks along with an *ordering* between the landmarks and used landmarks and landmark orderings to decompose the planning task into several smaller subtasks with the landmarks as subgoals (Porteous, Sebastia, and Hoffmann 2001; Hoffmann, Porteous, and Sebastia 2004).

A crucial weakness of these approaches is that they are incomplete. The LAMA planner (Richter and Westphal 2010), winner of the satisficing track of the International Planning Competition (IPC) 2008, was the first planning system that guides a (path-dependent) search algorithm with a *heuristic* that is obtained from landmarks. The landmark-count heuristic computes a set of ordered landmarks for the initial state and keeps track of the number of landmarks that remain to be achieved on every path. Since multiple landmarks can be achieved by a single action, this heuristic is inadmissible and hence not suited for optimal planning.

Karpas and Domshlak (2009) applied the idea of cost partitioning (Katz and Domshlak 2008; Yang et al. 2008) to landmarks to obtain an admissible version of the landmark-count heuristic, and Bonet and Helmert (2010) showed that the heuristic value of an optimal cost partitioning over a given set of landmarks is equal to the linear program (LP) relaxation of the integer program (IP) that computes the minimum hitting set (MHS) over the landmarks.

While the MHS heuristic is the best possible admissible heuristic that only takes the landmarks into account, it can be improved by considering the orderings between the landmarks. One source of information that can be derived from orderings are cyclic dependencies between landmarks. While existing landmark heuristics exploit that each landmark must be achieved at least once, each cyclic dependency of landmarks implies that one of the landmarks in the cycle must be achieved at least twice. Paul and Helmert (2016) exploit this observation for the game of FREECELL and obtain heuristic values close to the perfect heuristic. Later, Paul et al. (2017) found that the same idea can also be used to solve all instances of the LOGISTICS domain of IPC 1998 and 2000.

In this paper, we generalize the idea to domain-independent planning. After introducing the required background, we describe how the MHS IP can be strengthened by the addition of *cyclic landmark constraints* and show that the resulting heuristic dominates the MHS heuristic as well as h^+ if the set of landmarks consists of all delete-relaxation landmarks. We further strengthen this result by presenting another heuristic that additionally takes the *type* of ordering relation into account. As the number of cycles can be exponential in the number of landmarks, it is not tractable to precompute all cycles. We therefore propose an algorithm that is inspired by approaches for implicit hitting set problems (Chandrasekaran et al. 2011): we solve an IP or LP to cover a small (initially empty) set of cycles, generate a cycle not covered by the solution, add the corresponding cyclic landmark constraint, and iterate until no more uncovered cycles exist. We show the potential of considering cyclic dependencies in an experimental evaluation on the IPC 1998–2018 benchmark suite.

Background

Classical Planning We consider classical planning in the SAS$^+$ formalism (Bäckström and Nebel 1995), where a planning task is given as a 4-tuple $\mathcal{T} = \langle \mathcal{V}, s_0, G, \mathcal{A} \rangle$. $\mathcal{V}$ is a finite set of *finite-domain state variables* v with associated domain $\mathrm{dom}(v)$. An *atom* $v \mapsto d$ is a value assignment of value $d \in \mathrm{dom}(v)$ to $v \in \mathcal{V}$. A *partial variable assignment* is a set of atoms, each for a different variable. A *state*

is a variable assignment defined on all variables $v \in \mathcal{V}$. We use set notation such as $v \mapsto d \in s$ and function notation such as $s(v) = d$ interchangeably. s_0 is the *initial state*, and the *goal* G is a partial variable assignment. $\mathcal{A}$ is a finite set of *actions* $a = \langle pre, eff, cost \rangle$ (also called *operators*), where *precondition* $pre(a)$ and *effect* $eff(a)$ are partial variable assignments and $cost(a) \in \mathbb{R}_0^+$ is the cost of a.

An action $a \in \mathcal{A}$ is *applicable* in state s if $s(v) = d$ for all $v \mapsto d \in pre(a)$. Applying an applicable a in s results in the state $s' = s[\![a]\!]$ where $s'(v) = d$ for all $v \mapsto d \in eff(a)$ and $s'(v) = s(v)$ otherwise. An *action sequence* $\pi = \langle a_1, \ldots, a_n \rangle$ is applicable in $s = s_1$ if $s_{i+1} = s_i[\![a_i]\!]$ for all $i = 1, \ldots, n$ and each action a_i is applicable in s_i. The state that results from applying an applicable π in s is written as $s[\![\pi]\!]$. An *s-plan* is an action sequence π such that $G \subseteq s[\![\pi]\!]$. The *cost* of an s-plan $\pi = \langle a_1, \ldots, a_n \rangle$ is the sum over the action costs of the sequence: $cost(\pi) = \sum_{i=1}^{n} cost(a_i)$. An s-plan is *optimal* if it has minimal cost among all s-plans.

Landmarks and Orderings A *disjunctive action landmark* (*landmark* for short) for a state s is a set of actions $L \subseteq \mathcal{A}$ such that every s-plan contains an action $a \in L$. The *first occurrence* of a landmark L in an s-plan $\pi = \langle a_1, \ldots, a_n \rangle$ is i iff $a_i \in L$ and $a_j \notin L$ for all $1 \leq j < i$. The *last occurrence* of L in π is i iff $a_i \in L$ and $a_j \notin L$ for all $i < j \leq n$. We write $first_L(\pi) = i$ if the first occurrence of L in π is i, and $last_L(\pi) = i$ if i is the last occurrence. Furthermore, the *number of occurrences* of L in π is $occ_L(\pi) = \sum_{i=1}^{n} [a_i \in L]$ ($[\cdot]$ denote Iverson brackets).

A *landmark ordering* denotes a dependency between two landmarks. Given landmarks L and L' for a state s, there is a *strong ordering* $L \to_\mathrm{s} L'$ between L and L' iff $first_L(\pi) < first_{L'}(\pi)$ for all s-plans π, and there is a *weak ordering* $L \to_\mathrm{w} L'$ between L and L' iff $first_L(\pi) < last_{L'}(\pi)$ for all s-plans π. Observe that every strong ordering also implies a weak ordering since $first_{L'}(\pi) \leq last_{L'}(\pi)$. We say that an action sequence $\pi = \langle a_1, \ldots, a_n \rangle$ *satisfies* such a strong (weak) ordering iff $first_L(\pi) < first_{L'}(\pi)$ ($first_L(\pi) < last_{L'}(\pi)$), and that it *violates* it otherwise.

Our definitions of ordering types differ from the orderings that have been described by Hoffmann, Porteous, and Sebastia (2004). *Natural*, *necessary* and *greedy necessary* orderings are strong orderings. *Reasonable orderings* between L and L' guarantee $first_L(\pi) \leq last_{L'}(\pi)$, but unlike with weak orderings, the inequality is not necessarily strict.

Landmark Graph Given a set of landmarks $\mathcal{L}$ and a set of landmark orderings $\mathcal{O}$ for state s, the corresponding landmark graph $\mathcal{G} = \langle \mathcal{L}, \mathcal{O} \rangle$ is a directed graph with a vertex for every landmark in $\mathcal{L}$ and an edge for every ordering in $\mathcal{O}$. An edge from node L to L' is labeled with the type $t \in \{\mathrm{s}, \mathrm{w}\}$ of the corresponding ordering $L \to_t L'$ (i.e., strong or weak). A *path* in $\mathcal{G}$ is a chain of edges $\pi = L_1 \to_{t_1} \cdots \to_{t_n} L_{n+1}$ such that $L_i \to_{t_i} L_{i+1} \in \mathcal{O}$ for $1 \leq i \leq n$. A path is a *cycle* if $L_1 = L_{n+1}$, and a cycle is *elementary* if $L_i \neq L_j$ for all pairs $1 \leq i < j \leq n$. Given a cycle $c = L_1 \to_{t_1} \cdots \to_{t_n} L_{n+1}$, the set of landmarks in c is $\mathcal{L}(c) = \{L_i \mid 1 \leq i \leq n\}$, and the set of orderings is $\mathcal{O}(c) = \{L_i \to_{t_i} L_{i+1} \mid 1 \leq i \leq n\}$. Two elementary cycles c_1 and c_2 are *distinct* if one is not a cyclic permutation of the other (i.e., $\mathcal{O}(c_1) \neq \mathcal{O}(c_2)$).

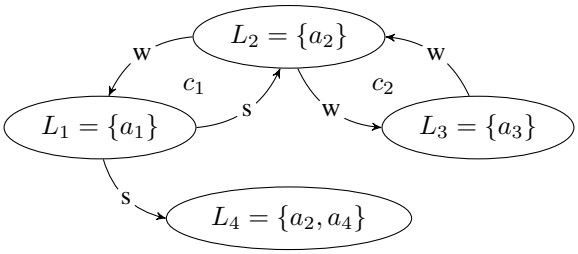

Figure 1: Example landmark graph.

Figure 1 shows a landmark graph with four landmarks $L_1 = \{a_1\}$, $L_2 = \{a_2\}$, $L_3 = \{a_3\}$, and $L_4 = \{a_2, a_4\}$. The orderings in the landmark graph create two elementary cycles $c_1 = L_1 \to_\mathrm{s} L_2 \to_\mathrm{w} L_1$ and $c_2 = L_2 \to_\mathrm{w} L_3 \to_\mathrm{w} L_2$. The cycle $c_1' = L_2 \to_\mathrm{w} L_1 \to_\mathrm{s} L_2$ is not distinct from c_1, and the cycle $c = L_1 \to_\mathrm{s} L_2 \to_\mathrm{w} L_3 \to_\mathrm{w} L_2 \to_\mathrm{w} L_1$ is not elementary, as it is a combination of the elementary cycles c_1 and c_2 that includes L_2 twice.

Landmark Heuristic Most often in planning, landmarks are used to compute heuristic values. The cost of a minimum hitting set is the most accurate admissible heuristic that can be derived from a set of disjunctive action landmarks when no further information is available. Let X be a set, $\mathcal{F} = F_1, \ldots, F_n \subseteq 2^X$ be a family of subsets of X, and $c : X \to \mathbb{R}_0^+$ be a cost function for X. A *hitting set* is a subset $H \subseteq X$ that "hits" all subsets in $\mathcal{F}$, i.e., $H \cap F \neq \emptyset$ for all $F \in \mathcal{F}$. The *cost* of H is $\sum_{x \in H} c(x)$. A *minimum hitting set* (MHS) is a hitting set with minimal cost.

The cost of an MHS over a set of landmarks $\mathcal{L} \subseteq 2^\mathcal{A}$ corresponds to the objective value of the MHS *integer program* which is given as follows:

$$\min \sum_{a \in \mathcal{A}} \mathsf{Y}_a \, cost(a) \quad \text{s.t.} \tag{1}$$

$$\mathsf{Y}_a \geq 0 \quad \text{for all } a \in \mathcal{A} \text{ and} \tag{2}$$

$$\sum_{a \in L} \mathsf{Y}_a \geq 1 \quad \text{for all } L \in \mathcal{L} \tag{3}$$

Note that $\mathsf{Y}_a = 0$ or $\mathsf{Y}_a = 1$ in all optimal solutions of the MHS IP.[1] An MHS can then be constructed from the IP solution as the set of all actions $a \in \mathcal{A}$ with $\mathsf{Y}_a = 1$.

Definition 1. ***Landmark heuristic***
Let $\mathcal{T}$ be a planning task, s be a state of $\mathcal{T}$, and $\mathcal{L}$ be a set of landmarks for s. The landmark heuristic h^LM *for s and $\mathcal{L}$ is the objective value of the MHS IP over $\mathcal{L}$.*

Consider the example landmark graph for a state s from a unit cost planning task in Figure 1, ignoring all ordering

[1] There are optimal solutions with $\mathsf{Y}_a > 1$ for some a in the presence of 0-cost actions, but setting each such Y_a to 1 instead is still optimal.

relations (i.e., all edges) for now. We have $h^{\mathrm{LM}}(s) = 3$, as $\mathsf{Y}_{a_1} = \mathsf{Y}_{a_2} = \mathsf{Y}_{a_3} = 1$ in every solution of the MHS IP because of the landmark constraints for L_1, L_2, and L_3, and $\mathsf{Y}_{a_4} = 0$ in an optimal solution as the landmark constraint for L_4 is already satisfied with $\mathsf{Y}_{a_2} = 1$.

The landmark heuristic is an example of an *operator-counting* heuristic (Pommerening et al. 2014). In general, operator-counting heuristics are defined as the objective value of an IP with the same variables Y_a (and possibly additional auxiliary variables), the same objective function as in Definition 1, the basic constraints $\mathsf{Y}_a \geq 0$, and usually additional constraints.[2]

These additional constraints are called *operator-counting constraints*, and they must guarantee that for every plan π, setting Y_a to the number of occurrences of a in π satisfies the constraint. As long as this condition is satisfied, all operator-counting heuristics are admissible. In particular, this means that operator-counting heuristics can be improved by adding additional operator-counting constraints. Besides landmarks, operator-counting constraints can be derived from network flow balance equations (Bonet 2013), pattern databases (Pommerening, Röger, and Helmert 2013) and abstract transition systems (Pommerening et al. 2014).

Cyclic Landmark Heuristic

We generalize the idea to exploit cyclic dependencies in FREECELL (Paul and Helmert 2016) to domain-independent planning. For each cycle in a given landmark graph, we can deduce that at least one of the landmarks in the cycle must occur twice to satisfy all of its orderings. This gives rise to the following definition of *cyclic landmark constraints*.

Theorem 1. ***Cyclic landmark constraints***
Let $\mathcal{T}$ be a planning task, s be a state of $\mathcal{T}$, π be an s-plan and $c = L_1 \to_{t_1} \cdots \to_{t_{n-1}} L_n \to_{t_n} L_1$ be a cycle in a landmark graph for s.

The accumulated number of occurrences in π of all landmarks in c must be at least $n + 1$, i.e.,

$$\sum_{i=1}^{n} \mathit{occ}_{L_i}(\pi) \geq n + 1.$$

Proof. Every landmark in $\mathcal{L}(c)$ must occur in every s-plan π, so $\mathit{occ}_{L_i}(\pi) \geq 1$ for all $1 \leq i \leq n$, which implies $\sum_{i=1}^{n} \mathit{occ}_{L_i}(\pi) \geq n$. We assume that π is an s-plan with $\sum_{i=1}^{n} \mathit{occ}_{L_i}(\pi) = n$ and derive a contradiction, from which the claim follows.

Because all landmarks must occur at least once in π, by the pigeonhole principle every landmark occurs exactly once in π. Let L' be a landmark that occurs first in π among all landmarks from c, i.e. $\mathit{first}_{L'}(\pi) \leq \mathit{first}_L(\pi)$ for all $L \in \mathcal{L}(c)$. Since $\mathit{occ}_{L'}(\pi) = 1$, we get $\mathit{first}_{L'}(\pi) = \mathit{last}_{L'}(\pi)$. As L' is part of cycle c, there is an $L \in \mathcal{L}(c)$ such that $L \to_{\mathrm{w}} L' \in \mathcal{O}(c)$ (recall that every strong ordering implies a weak ordering). By our choice of L', the first (and only) occurrence of L cannot be earlier than the first occurrence of L', i.e., $\mathit{first}_L(\pi) \geq \mathit{first}_{L'}(\pi) = \mathit{last}_{L'}(\pi)$. It follows that $\mathit{first}_L(\pi) \geq \mathit{last}_{L'}(\pi)$, which violates the ordering $L \to_{\mathrm{w}} L' \in \mathcal{O}(c)$. □

We apply this result by expressing the bound on the number of occurrences of the landmarks of a cycle as an operator-counting constraint. Combining this constraint with the constraints of the landmark heuristic gives rise to the *cyclic landmark heuristic*.

Definition 2. ***Cyclic landmark heuristic***
Let $\mathcal{T}$ be a planning task with actions $\mathcal{A}$, s be a state of $\mathcal{T}$, $\mathcal{G} = \langle \mathcal{L}, \mathcal{O} \rangle$ be a landmark graph for s, and $\mathcal{C}$ be a set of cycles in $\mathcal{G}$.

The cyclic landmark heuristic h^{cycle} *for s, $\mathcal{G}$ and $\mathcal{C}$ is the objective value of the* cyclic landmark IP*:*

$$\min \sum_{a \in \mathcal{A}} \mathsf{Y}_a \mathit{cost}(a) \quad \textit{s.t.} \tag{4}$$

$$\mathsf{Y}_a \geq 0 \quad \textit{for all } a \in \mathcal{A} \tag{5}$$

$$\sum_{a \in L} \mathsf{Y}_a \geq 1 \quad \textit{for all } L \in \mathcal{L} \textit{ and} \tag{6}$$

$$\sum_{L \in \mathcal{L}(c)} \sum_{a \in L} \mathsf{Y}_a \geq |\mathcal{L}(c)| + 1 \quad \textit{for all } c \in \mathcal{C}. \tag{7}$$

Consider again the example landmark graph in Figure 1, this time ignoring only the types of the ordering relations (i.e., the transition labels). It still holds that Y_{a_1}, Y_{a_2} and Y_{a_3} must be set to at least 1 in all solutions to satisfy the landmark constraints for L_1, L_2, and L_3, and $\mathsf{Y}_{a_4} = 0$ in an optimal solution. Every solution must additionally satisfy the cyclic landmark constraints $\mathsf{Y}_{a_1} + \mathsf{Y}_{a_2} \geq 3$ and $\mathsf{Y}_{a_2} + \mathsf{Y}_{a_3} \geq 3$. An optimal solution has $\mathsf{Y}_{a_2} = 2$ because a_2 occurs in both cyclic landmark constraints, which yields a heuristic value of $h^{\mathit{cycle}}(s) = 4$.

Note that a feasible solution for the cyclic landmark IP always exists in solvable planning tasks: setting $\mathsf{Y}_a = 2$ for all $a \in \mathcal{A}$ satisfies all constraints of the IP. Because it is an operator-counting heuristic, h^{cycle} is admissible.

Theorem 2. ***Admissibility of*** h^{cycle}
The cyclic landmark heuristic h^{cycle} is admissible.

Proof. h^{cycle} follows the template of an operator-counting heuristic. Pommerening et al. (2014) show that all operator-counting heuristics are admissible and that Constraints 5 and 6 are valid operator-counting constraints. Constraint 7 is an operator-counting constraint due to Theorem 1, noting that $\sum_{a \in L} \mathsf{Y}_a = \mathit{occ}_L(\pi)$ for all s-plans π when Y_a is set to the number of occurrences of a in π as required by the operator-counting framework. □

Because h^{cycle} includes all operator-counting constraints of h^{LM}, it can never result in a worse heuristic estimate.

Theorem 3. ***Dominance of*** h^{cycle} ***over*** h^{LM}
Let s be a state of a planning task. Consider h^{LM} and h^{cycle} using the same landmark graph for s.

Then $h^{\mathit{cycle}}(s) \geq h^{\mathrm{LM}}(s)$. Furthermore, there are cases where the dominance is strict.

[2] If the IP is infeasible, the heuristic value is ∞.

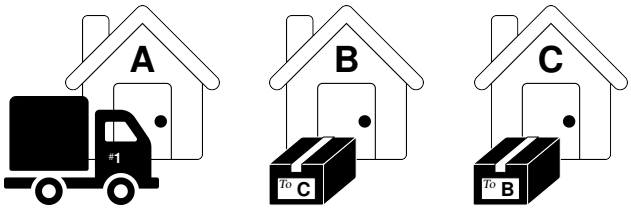

Figure 2: Example logistics task with cyclic landmark dependencies: in order to deliver both packages, the truck must visit location B or C twice.

Proof. Dominance follows directly from Proposition 1 by Pommerening et al. (2014).

As discussed above, Figure 1 shows a case where $h^{cycle}(s) > h^{\mathrm{LM}}(s)$, proving that there are cases where the dominance is strict. □

Bonet and Helmert (2010) have shown that the landmark heuristic is equal to h^+ if the set of landmarks consists of all sets of actions that are landmarks of the delete relaxation (Hoffmann and Nebel 2001) of a planning task. Together with Theorem 3 this leads to the following corollary.

Corollary 1. ***Dominance of*** h^{cycle} ***over*** h^+
Let $\mathcal{T}$ be a planning task, s be a state of $\mathcal{T}$ and $\mathcal{G}$ be a landmark graph for s over all delete-relaxation landmarks of s.

Then $h^{cycle}(s) \geq h^+(s)$. Furthermore, there are cases where the dominance is strict.

Proof. Dominance follows directly from Theorem 3 and the result of Bonet and Helmert (2010).

We show that there are cases where $h^{cycle}(s) > h^+(s)$ based on the LOGISTICS example depicted in Figure 2: two packages need to be exchanged between locations B and C by a truck which is initially stationed at a third location A. The set of all delete-relaxation landmarks for the depicted state contains the following (pairwise disjoint) disjunctive action landmarks:

- landmarks L^d_B and L^d_C for driving to B and C,
- landmarks L^l_B and L^l_C to load packages at B and C, and
- landmarks L^u_B and L^u_C to unload packages at B and C.

Each package induces a weak ordering $L^d_o \to_{\mathrm{w}} L^d_t$ because it must be loaded into the truck at its origin o before it can be unloaded at its target location t. With $\mathcal{C} = \{L^d_B \to_{\mathrm{w}} L^d_C \to_{\mathrm{w}} L^d_B\}$, we have $h^{cycle}(s) = 7 > 6 = h^+(s)$. □

Strong Cyclic Landmark Heuristic

In the previous section, we enhanced the landmark heuristic with constraints that take into account the existence of landmark orderings. In this section, we strengthen these cyclic landmark constraints by additionally considering the *type* of ordering (weak or strong).

Theorem 4. ***Strong cyclic landmark constraints***
Let $\mathcal{T}$ be a planning task, s be a state of $\mathcal{T}$, π be an s-plan and $c = L_1 \to_{t_1} \cdots \to_{t_{n-1}} L_n \to_{t_n} L_1$ be a cycle in a landmark graph for s. Let $\mathcal{L}^{\mathrm{w}}(c) = \{L' \mid L \to_{\mathrm{w}} L' \in \mathcal{O}(c)\}$ be the set of landmarks on the right-hand side of a weak ordering in c.

The accumulated number of occurrences in π of all landmarks in $\mathcal{L}^{\mathrm{w}}(c)$ must be at least $|\mathcal{L}^{\mathrm{w}}(c)| + 1$, i.e.,

$$\sum_{L \in \mathcal{L}^{\mathrm{w}}(c)} occ_L(\pi) \geq |\mathcal{L}^{\mathrm{w}}(c)| + 1.$$

Proof. If $\mathcal{L}^{\mathrm{w}}(c) = \emptyset$, the inequality simplifies to the impossible $0 \geq 1$. In this case c is a cycle consisting only of strong orderings. Because strong orderings are transitive (which is immediate from their definition), the landmarks on the cycle are strongly ordered before themselves, which is impossible to satisfy. Hence, no s-plan exists and $0 \geq 1$ vacuously holds for all s-plans π.

Otherwise, c contains at least one weak ordering. If all orderings in c are weak, the result follows from Theorem 1. (Note that for cycles that only consist of weak orderings, both theorems make identical claims.) Otherwise, c must contain a strong ordering cyclically followed by a weak ordering: $L \to_{\mathrm{s}} L'$ and $L' \to_{\mathrm{w}} L''$. Hence, for every s-plan π we have $\mathit{first}_L(\pi) < \mathit{first}_{L'}(\pi)$ and $\mathit{first}_{L'}(\pi) < \mathit{last}_{L''}(\pi)$, which implies $\mathit{first}_L(\pi) < \mathit{last}_{L''}(\pi)$ and therefore $L \to_{\mathrm{w}} L''$. We can thus contract the cycle by replacing $L \to_{\mathrm{s}} L'$ and $L' \to_{\mathrm{w}} L''$ with $L \to_{\mathrm{w}} L''$, which reduces the number of strong orderings by one and does not change the set $\mathcal{L}^{\mathrm{w}}(c)$. By iterating the argument, we eventually arrive at the case where all orderings are weak. □

The *strong cyclic landmark heuristic* corresponds to the landmark heuristic enhanced with strong cyclic landmark constraints.

Definition 3. ***Strong cyclic landmark heuristic***
Let $\mathcal{T}$ be a planning task with actions $\mathcal{A}$, s be a state of $\mathcal{T}$, $\mathcal{G} = \langle \mathcal{L}, \mathcal{O} \rangle$ be a landmark graph for s, and $\mathcal{C}$ be a set of cycles in $\mathcal{G}$.

The strong cyclic landmark heuristic h^{strong} *for s, $\mathcal{G}$ and $\mathcal{C}$ is the objective value of the* strong cyclic landmark IP*:*

$$\min \sum_{a \in \mathcal{A}} \mathsf{Y}_a \mathit{cost}(a) \quad \textit{s.t.} \tag{8}$$

$$\mathsf{Y}_a \geq 0 \quad \textit{for all } a \in \mathcal{A} \tag{9}$$

$$\sum_{a \in L} \mathsf{Y}_a \geq 1 \quad \textit{for all } L \in \mathcal{L} \textit{ and} \tag{10}$$

$$\sum_{L \in \mathcal{L}^{\mathrm{w}}(c)} \sum_{a \in L} \mathsf{Y}_a \geq |\mathcal{L}^{\mathrm{w}}(c)| + 1 \quad \textit{for all } c \in \mathcal{C}. \tag{11}$$

Consider once more the example from Figure 1, this time also taking into account the ordering types. To satisfy the landmark constraints, an optimal solution must set Y_{a_1}, Y_{a_2} and Y_{a_3} to at least 1. The strong cyclic landmark constraint for cycle $c_1 = L_1 \to_{\mathrm{s}} L_2 \to_{\mathrm{w}} L_1$ cannot be satisfied by setting $Y_{a_2} = 2$ as it was the case in the cyclic landmark heuristic as L_2 is reached via a strong ordering. The constraint can hence only be resolved by setting $\mathsf{Y}_{a_1} = 2$, and it is therefore not possible to satisfy both strong cyclic landmark constraints by setting the same operator count to 2. As a result, we have $h^{strong}(s) = 5$.

Theorem 5. ***Admissibility of*** h^{strong}
The strong cyclic landmark heuristic is admissible.

Proof. The same argument as in Theorem 2 applies, but this time we have to show that Constraint 11 is an operator-counting constraint. This follows from Theorem 4 and the observation $\sum_{a \in L} \mathsf{Y}_a = occ_L(\pi)$ for all s-plans π when Y_a is set to the number of occurrences of a in π as required by the operator-counting framework. □

The heuristic values of the strong cyclic landmark heuristic h^{strong} can never be lower than the heuristic values of the cyclic landmark heuristic h^{cycle} presented in the previous section.

Theorem 6. ***Dominance of*** h^{strong} ***over*** h^{cycle}
Let $\mathcal{T}$ be a planning task, s be a state of $\mathcal{T}$ and $\mathcal{G}$ be a landmark graph for s.

Then $h^{strong}(s) \geq h^{cycle}(s)$. Furthermore, there are cases where the dominance is strict.

Proof sketch. Constraints 10 and 11 imply Constraint 7, so we can add Constraint 7 to the strong cyclic landmark IP without changing its objective value. With this, dominance follows from Proposition 1 by Pommerening et al. (2014).

As discussed above, Figure 1 shows a case where $h^{strong}(s) > h^{cycle}(s)$, proving that there are cases where the dominance is strict. □

From Theory to Practice

The heuristics we describe in the preceding sections are hard to compute in practice for several reasons: solving IPs is NP-hard (Karp 1972), and the amount of elementary cycles in a landmark graph can grow faster than exponentially in the number of vertices (Johnson 1975). In this section we discuss how to make computing our heuristics feasible in practice.

LP-relaxation of integer programs It is possible to approximate an IP solution by solving the *linear program* with the same objective function and set of constraints. This is called the LP-relaxation, and it can be computed in polynomial time. LP solutions may be real-valued whereas only integer values are allowed in IP solutions. Pommerening et al. (2014) show that the LP-relaxation of an operator counting (IP) heuristic is also an admissible heuristic. Note that the proofs for Theorems 3 and 6 do not assume integer-valued solutions, and they are therefore also valid for the LP-relaxations of the heuristics.

Generation of Ordered Landmarks Our contribution is to exploit given landmarks and orderings; finding them is an independent problem which we do not study in this paper. Several methods for computing landmark graphs are described in the literature (e.g., Richter, Helmert, and Westphal 2008; Keyder, Richter, and Helmert 2010). These methods are designed to find *fact landmarks* for a state s: every node in a landmark graph is associated with an atom $v \mapsto d$, and every s-plan must reach a state s' where $s'(v) = d$. Obviously, the orderings generated by these methods also order fact landmarks rather than disjunctive action landmarks.

A fact landmark for atom $v \mapsto d$ induces a disjunctive action landmark $L_{v \mapsto d}$ that contains its possible *achievers*, i.e., $L_{v \mapsto d} = \{a \in \mathcal{A} \mid v \mapsto d \in \mathit{eff}(a)\}$ (Karpas and Domshlak 2009). We exploit this to construct the disjunctive action landmark graph $\mathcal{G}_L = \langle \mathcal{L}, \mathcal{O}_L \rangle$ from a set of fact landmarks $\mathcal{F}$ and orderings $\mathcal{O}_F$ over $\mathcal{F}$, where

1. $\mathcal{L}$ contains a landmark $L_{v \mapsto d}$ for every $v \mapsto d \in \mathcal{F}$;
2. $\mathcal{O}_L$ contains a strong ordering $(L_{v \mapsto d}) \to_{\mathrm{s}} (L_{v' \mapsto d'})$ for every natural ordering $(v \mapsto d) \to (v' \mapsto d') \in \mathcal{O}_F$; and
3. $\mathcal{O}_L$ contains a weak ordering $(L_{v \mapsto d}) \to_{\mathrm{w}} (L_{v' \mapsto d'})$ for every reasonable ordering $(v \mapsto d) \to (v' \mapsto d')$ if $L_{v \mapsto d} \cap L_{v' \mapsto d'} = \emptyset$[3] and if there is no natural ordering $(v \mapsto d) \to (v' \mapsto d')$ in $\mathcal{O}_F$.

Note that the mapping of landmarks is surjective as different atoms from $\mathcal{F}$ may have identical sets of possible achievers and may hence be mapped to the same disjunctive action landmark. This may even introduce cycles in $\mathcal{G}_L$ that are not present in $\mathcal{G}_F$.

Finding Cycles in Landmark Graphs A crucial part for the quality of our heuristics are i) the existence and ii) the efficient detection of cycles in the landmark graph. We do not address i) in this paper, even though we believe that landmark generation methods that actively search for cyclic ordering dependencies, e.g., based on the domain-specific landmark generation methods of Paul and Helmert (2016) for FREECELL or Paul et al. (2017) for LOGISTICS tasks, can significantly impact the performance of our heuristics.

We consider two different versions that address ii) here. The JOHNSON cycle detection method uses Johnson's Algorithm (Johnson 1975) in every state s to determine all elementary cycles $\mathcal{C}_J$ for a given landmark graph $\mathcal{G}$ and computes our heuristics for s, $\mathcal{G}$ and $\mathcal{C}_J$. Johnson's Algorithm finds the next cycle in time that is polynomial in the number of landmarks $|\mathcal{L}|$, but the number of elementary cycles can be more than exponential in $|\mathcal{L}|$. Detecting cycles with JOHNSON is hence prohibitively expensive in the worst case.

The fact that a single cycle can be found efficiently is still encouraging, though. The ORACLE cycle detection method does not precompute all elementary cycles. Instead, it embeds the heuristic computation in an iterative process based on the generic algorithm for solving instances of the implicit hitting set problem (Chandrasekaran et al. 2011): starting with the MHS LP (or IP) computed by the landmark heuristic, ORACLE iteratively solves the LP, queries a cycle oracle for a cycle constraint that is violated by the current solution $\mathsf{Y} = \{\mathsf{Y}_a \mid a \in \mathcal{A}\}$ (an *uncovered cycle*) and adds that constraint to the LP. The process terminates when the oracle confirms that all cycle constraints are satisfied by the current LP solution. The generated set of cycles is a (typically

[3]The definition of reasonable orderings permits achieving both facts simultaneously (Hoffmann, Porteous, and Sebastia 2004). This is prohibited in our definition of weak orderings.

small) subset of all cycles which is large enough to represent the entire information gain of cyclic dependencies in the landmark graph.

For this implicit cycle covering approach to be beneficial, querying the cycle oracle has to be computationally cheap. ORACLE first computes the *excess operator count* for each landmark as $\mathsf{Y}_L := (\sum_{a \in L} \mathsf{Y}_a) - 1$. Note that Y must satisfy all landmark constraints, so $\sum_{a \in L} \mathsf{Y}_a \geq 1$ and hence $\mathsf{Y}_L \geq 0$ for all $L \in \mathcal{L}$. ORACLE determines if a cycle c is uncovered only based on Y_L: if the excess operator counts of all $L \in \mathcal{L}(c)$ for the cyclic landmark heuristic h^{cycle} and of all $L \in \mathcal{L}^{\mathrm{w}}(c)$ for the strong cyclic landmark heuristic h^{strong} sum up to less than 1, the (strong) cyclic landmark constraint of c is violated by Y and c is an uncovered cycle.

As generating all cycles can be prohibitively expensive, checking every cycle if it is uncovered will not make ORACLE more efficient than JOHNSON. Instead, we use the excess operator counts of all landmarks to directly generate an uncovered cycle or determine that no uncovered cycle is left. For an IP heuristic, we can do this by creating a subgraph $\mathcal{G}'$ of $\mathcal{G}$ which is such that all cycles in $\mathcal{G}'$ are cycles that are not covered by Y and by applying any cycle detection algorithm to $\mathcal{G}'$. Because of the restriction to integer solutions, $\mathsf{Y}_L \neq 0$ implies $\mathsf{Y}_L \geq 1$; therefore, we know for all uncovered cycles c that $\mathsf{Y}_L = 0$ for all $L \in \mathcal{L}(c)$ (for all $L \in \mathcal{L}^{\mathrm{w}}(c)$) for h^{cycle} (h^{strong}) as a single landmark with $\mathsf{Y}_L \neq 0$ would satisfy the (strong) landmark constraint of c by itself.

For h^{cycle}, $\mathcal{G}'$ is the subgraph of $\mathcal{G}$ that is induced by all landmarks L with $\mathsf{Y}_L = 0$; and for h^{strong}, $\mathcal{G}'$ is a copy of $\mathcal{G}$ except that all orderings $L' \to_{\mathrm{w}} L$ for all L with $\mathsf{Y}_L \neq 0$ are discarded. Any cycle in $\mathcal{G}'$ induces a (strong) cyclic landmark constraint that is violated by the current solution Y, and if the graph is acyclic, no more violated cycles are left and the objective value of the IP is the heuristic value of the (strong) cyclic landmark heuristic for the current state.

If an LP heuristic is computed, we also compute excess operator counts for all landmarks, but we cannot simply remove vertices or edges from the landmark graph to compute an uncovered cycle: excess operator counts with $0 < \mathsf{Y}_L < 1$ make the problem harder, as it is possible that no single landmark satisfies the (strong) cycle constraint but a set of landmarks may. It is nonetheless possible to determine efficiently if there is an uncovered cycle c in $\mathcal{G}$: we use the excess operator counts to generate a weighted graph $\mathcal{G}'$ that is a copy of $\mathcal{G}$ except that it is weighted in the following way: for h^{cycle}, we set the weight of all orderings $L \to L'$ to $w_{L \to L'} := \mathsf{Y}_{L'}$; and for h^{strong}, we set $w_{L \to L'}$ to $\mathsf{Y}_{L'}$ if $L \to L'$ is weak, and we set $w_{L \to L'}$ to 0 otherwise.

ORACLE then applies an all-pairs shortest path algorithm like the (polynomial) Floyd-Warshall algorithm (Floyd 1962; Roy 1959; Warshall 1962) to $\mathcal{G}'$. It tests for each $L \to L' \in \mathcal{O}$, if the sum of its weight $w_{L \to L'}$ and the cost of the shortest path from L' to L is smaller than 1. If this is not the case, $L \to L'$ cannot be part of an uncovered cycle and the next edge is tested until no more edges are left; otherwise, ORACLE found the uncovered cycle whose (strong) cyclic landmark constraint is violated the most among all cycles including $L \to L'$, and it returns that cycle.

Experimental Evaluation

We implemented our heuristics in version 19.06 of the Fast Downward planner (Helmert 2006) with CPLEX 12.9 as LP solver. We consider a benchmark set consisting of all 1827 planning tasks without conditional effects from the optimal sequential tracks of the International Planning Competitions 1998–2018. All experiments are conducted on Intel Xeon Silver 4114 processors running on 2.2 GHz with a time limit of 30 minutes and a memory limit of 3.5 GB. All benchmarks, code, and experiment data are published online (Büchner, Keller, and Helmert 2021).

We present results for the LM$^{\mathrm{RHW}}$ (Richter, Helmert, and Westphal 2008), LM$^{h^m}$ (Keyder, Richter, and Helmert 2010), and LM$^{\mathrm{BJOLP}}$ (Domshlak et al. 2011) landmark generators. For LM$^{h^m}$, we consider $m = 1$ (LM$^{h^1}$) and $m = 2$ (LM$^{h^2}$). LM$^{\mathrm{BJOLP}}$ combines the landmarks from LM$^{\mathrm{RHW}}$ and LM$^{h^1}$. The other landmark generators that are implemented in Fast Downward – LM$^{\mathrm{ZG}}$ (Zhu and Givan 2003) and a landmark generator that checks for each fact if it is a landmark in the delete relaxation – do not generate orderings and are hence omitted from our discussion. We furthermore enrich the LM$^{h^m}$ and LM$^{\mathrm{BJOLP}}$ landmarks with the weak orderings found by the method of Hoffmann, Porteous, and Sebastia (2004). LM$^{\mathrm{RHW}}$ already uses the same method as a subroutine and is not altered.

The MHS landmark heuristic h^{LM} that considers landmarks but ignores orderings serves as a baseline for our experiments. We use the LP relaxation of h^{LM}, h^{cycle} and h^{strong} to guide a search with the A*algorithm (Hart, Nilsson, and Raphael 1968) in all experiments.

In our implementation, the landmark heuristics compute a new landmark graph for every evaluated state and not just once as a preprocessing as the BJOLP planner (Domshlak et al. 2011) or LAMA (Richter and Westphal 2010) do. This is due to an observation of Büchner (2020), who shows that there are some instances where the landmark graph for the initial state is cyclic, but there are many more instances with cyclic landmark graphs for states that are encountered during search. Computing the landmark graph just once and keeping track of open landmarks is likely much better suited to maximize planner performance in terms of coverage. We leave this interesting research question for future work and recompute landmarks in every state as it allows a better analysis of the impact of our cyclic landmark heuristics, which is the focus of this evaluation.

Heuristic Accuracy This decision is supported by looking at two metrics that allow to compare the accuracy of our heuristics: the heuristic values in the initial state and the number of expansions before the last f-layer. For all considered landmark generators, there are less than 30 solved instances where $h^{cycle}(s_0) > h^{\mathrm{LM}}(s_0)$, but more than 100 solved instances where less states are expanded with h^{cycle} than with h^{LM} for LM$^{\mathrm{RHW}}$ and LM$^{\mathrm{BJOLP}}$. The picture is even more pronounced when looking at h^{strong}: for LM$^{\mathrm{RHW}}$ and LM$^{\mathrm{BJOLP}}$, we observe 33 solved instances where $h^{strong}(s_0) > h^{\mathrm{LM}}(s_0)$ and 154 solved instances

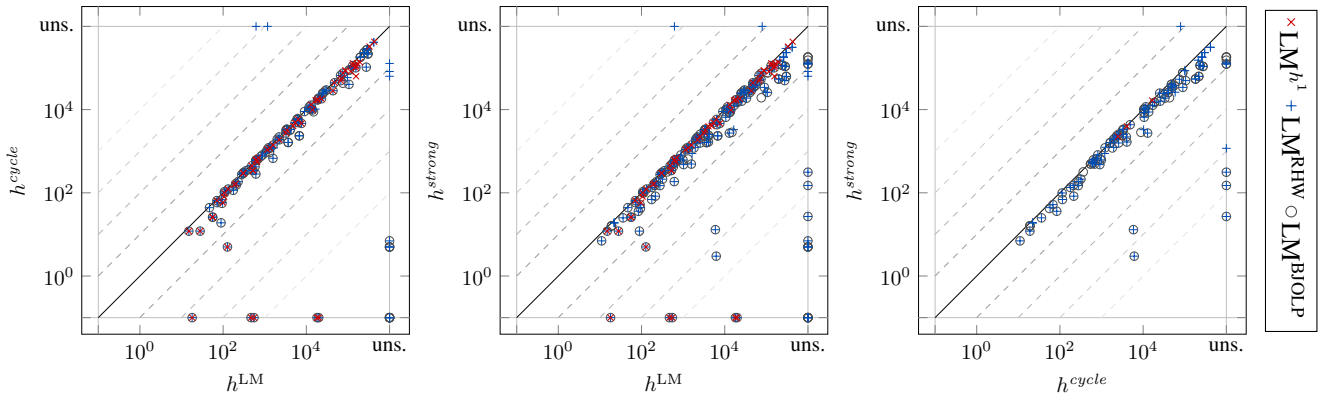

Figure 3: Number of expansions before the last f-layer for h^{LM}, h^{cycle} and h^{strong} in combination with different landmark generators and ORACLE cycle detection. Only instances where the numbers differ are displayed.

		JOHNSON		ORACLE	
	h^{LM}	h^{cycle}	h^{strong}	h^{cycle}	h^{strong}
$\mathrm{LM}^{\mathrm{RHW}}$	644	630	643	657	**665**
LM^{h^1}	652	653	651	**659**	**659**
LM^{h^2}	377	**383**	**383**	**383**	**383**
$\mathrm{LM}^{\mathrm{BJOLP}}$	619	611	622	631	**639**

Table 1: Coverage for different combinations of landmark generator, heuristic and cycle detection method.

where less expansions are required to compute an optimal plan with h^{strong} and $\mathrm{LM}^{\mathrm{RHW}}$.

Figure 3 compares the number of expansions before the last f-layer for all three pair-wise heuristic combinations. Note that only instances where these numbers differ are displayed, i.e., even though many data points appear to be on the diagonal, they are actually slightly below it. We can see that there are a few instances where the number of expansions drops by more than an order of magnitude. Most importantly, though, we see that there are both many instances where considering orderings has the potential to strengthen the heuristic and where considering the type of the cycles yields additional information on top of that.

Heuristic in Search We have observed a gain in heuristic accuracy in a significant number of instances, but the computation of h^{cycle} and h^{strong} is also more expensive than the computation of h^{LM}: in combination with JOHNSON, an LP has to be solved that contains all constraints of the MHS LP plus one additional constraint for every elementary cycle in the landmark graph. And ORACLE starts by computing the same LP as h^{LM}, but it has to do so multiple times with a (slowly) growing number of constraints until all cycles are covered. The question here is if the improved heuristic guidance outweighs the more expensive heuristic computations.

Table 1 shows coverage results for different combinations of landmark generator ($\mathrm{LM}^{\mathrm{RHW}}$, LM^{h^1}, LM^{h^2} and $\mathrm{LM}^{\mathrm{BJOLP}}$), heuristic ($h^{\mathrm{LM}}$, h^{cycle} and h^{strong}), and, in the case of h^{cycle} and h^{strong}, cycle detection method (JOHNSON and ORACLE). Comparing the coverage of the h^{LM} baseline to the coverage of the best performing h^{cycle} configuration shows an increase in coverage of 13 and 12 instances for the $\mathrm{LM}^{\mathrm{RHW}}$ and $\mathrm{LM}^{\mathrm{BJOLP}}$ landmark generators, respectively, and of 6 and 7 instances for the two versions of LM^{h^m}. In the case of $\mathrm{LM}^{\mathrm{RHW}}$ and $\mathrm{LM}^{\mathrm{BJOLP}}$, taking the type of cycles into account pays off even more, with a total coverage increase of 21 and 20 instances compared to h^{LM}. For all landmark generators and both for h^{cycle} and h^{strong} there is at least one configuration that significantly outperforms the h^{LM} baseline, and it is particularly encouraging that the combination of ORACLE and h^{strong} consistently performs best for all considered landmark generators.

Cycle Detection Detecting cycles with the ORACLE method always solves at least as many instances as the otherwise identical configuration with JOHNSON. Even worse, we observe a negative impact in comparison to the h^{LM} baseline in some cases where JOHNSON is used to generate cycles. This decrease in coverage is mostly caused by domains with a large number of cycles (in particular PEGSOL, SPIDER and SOKOBAN). Figure 4 shows that the average number of cycle constraints explicitly added to the LP is consistently much higher for JOHNSON than it is for ORACLE. While this is not surprising – after all, JOHNSON adds all cycles at once whereas ORACLE only adds cycles as long as necessary – the difference is large: over all instances that are solved by both cycle detection methods in combination with h^{cycle} and $\mathrm{LM}^{\mathrm{RHW}}$, JOHNSON adds 10.85 cycles on average and ORACLE only 0.19 cycles – a factor of more than 50 on average, that goes up to more than 5000 in the worst case.

The ORACLE approach trades solving a single large LP with solving several smaller LPs consecutively. Solving n LPs with the implicit hitting set approach does usually not mean that it takes n times as long, though. This is because the solver uses the solution of the previous iteration as a starting point for the current LP, which is often already very close to an optimal solution for the current problem. Over all

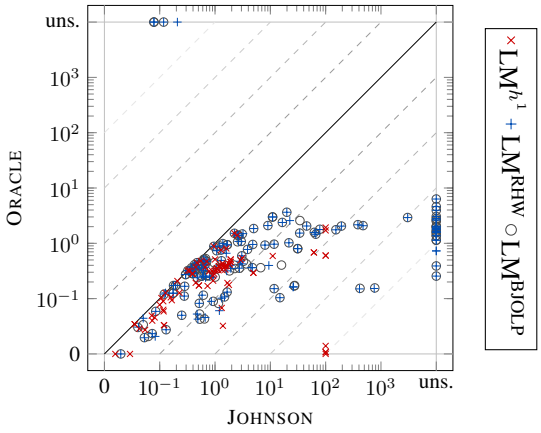

Figure 4: Average number of cycle constraints per evaluated state for JOHNSON and ORACLE.

instances, it never takes more than 7.32 iterations on average until an optimal solution that covers all cycles is computed.

This clearly shows that generating all cycles is not a viable option in many instances, and that our ORACLE approach generates cycle constraints that cover all cycles within only a few iterations.

Landmark Generators We can also observe that the choice of landmark generators affects the results in Table 1 significantly. Looking at the *relative* change compared to the h^{LM} baseline for each landmark generator reveals that LM$^{\text{RHW}}$ and LM$^{\text{BJOLP}}$ on the one hand and the two LM$^{h^m}$ versions on the other behave similarly: for the former, performance decreases with h^{cycle} and is comparable with h^{strong} if JOHNSON is used, and performance improves with h^{cycle} and even more so with h^{strong} in combination with ORACLE. For both LM$^{h^1}$ and LM$^{h^2}$, there is moderate improvement when h^{cycle} is used (for LM$^{h^2}$ with both cycle detection methods, for LM$^{h^1}$ only for ORACLE), and h^{strong} does not outperform h^{cycle}.

As LM$^{\text{BJOLP}}$ combines the LM$^{h^1}$ and LM$^{\text{RHW}}$ landmarks, one would have expected that it also combines properties from both, but it looks like a weaker version of LM$^{\text{RHW}}$ instead. We were surprised to see that the landmarks of LM$^{\text{BJOLP}}$ are almost always very similar (and often identical) to the landmarks of either LM$^{h^1}$ or LM$^{\text{RHW}}$, depending on which is the larger set. This implies that the smaller set of LM$^{h^1}$ and LM$^{\text{RHW}}$ landmarks is often a subset of the larger one. As orderings are derived from the set of landmarks, this also holds for the set of orderings. Since LM$^{\text{RHW}}$ generates more landmarks than LM$^{h^1}$ in the vast majority of the instances, LM$^{\text{BJOLP}}$ performs similar to but slightly worse than LM$^{\text{RHW}}$ as it essentially computes the same thing but requires additional effort (the computation of LM$^{h^1}$ landmarks) to get there.

A closer look at the orderings reveals one difference between LM$^{\text{RHW}}$ and LM$^{h^1}$ that explains why considering weak orderings is beneficial for LM$^{\text{RHW}}$ but not for LM$^{h^1}$: while the total number of orderings is similar for both landmark generators, LM$^{\text{RHW}}$ finds much more strong orderings and LM$^{h^1}$ finds much more weak ones. It turns out that this trade-off is no coincidence. In fact, most of the weak orderings from LM$^{h^1}$ are actually found to be strong by LM$^{\text{RHW}}$, which means that LM$^{h^1}$ essentially underevaluates the impact of these strong orderings. As a consequence, the strengthening promoted by h^{strong} has little impact when using LM$^{h^m}$, in contrast to LM$^{\text{RHW}}$.

Finally, we would like to emphasize that no landmark generator considered here is tailored to our needs. Our results show that all landmark generators produce similar sets of landmarks; Richter and Westphal (2010) regard cyclic dependencies between landmarks as undesirable and actively steer away from them, and they discard disjunctive landmarks without estimating what is lost; and the technique of Hoffmann, Porteous, and Sebastia (2004) is the only generation technique for the weak orderings that are essential for cycles in the landmark graph. We conjecture that landmark generators that actively pursue the generation of diverse landmarks, weak orderings and cycles will help to close the gap to the results of Paul and Helmert (2016) and Paul et al. (2017) who report close to perfect heuristic values in their domain-specific cyclic landmark heuristics.

Conclusions

We introduce two novel operator-counting heuristics h^{cycle} and h^{strong} which generalize the cycle-covering heuristic (Paul and Helmert 2016; Paul et al. 2017) to domain-independent planning. Theoretically, we show that h^{cycle} dominates the MHS landmark heuristic h^{LM} and that h^{strong} dominates h^{cycle}. The dominance comes from additional operator-counting constraints that are derived from cycles in the landmark graph which imply that some landmark has to be achieved at least twice.

To avoid the computation of all elementary cycles in the landmark graph, we propose ORACLE, an implicit hitting set method that generates a typically small set of cycles until all cycle constraints are provably satisfied. We show experimentally on benchmarks from the International Planning Competitions that cycles occur frequently in practice, that our heuristics provide improved guidance and that coverage increases for both heuristics if combined with ORACLE.

Haslum, Slaney, and Thiébaux (2012) compute a minimal subset of all delete-relaxation landmarks that allows to compute h^+. Their algorithm is, like our cycle generation approach, based on the implicit hitting set method. Our approach is not concerned with the generation of landmarks and hence treats the landmark and cycle generation as two separate entities: one where all landmarks are computed and another where orderings are added iteratively. In future work, we plan to entwine both parts to an algorithm that generates both cycles and landmarks iteratively and stops when the heuristic value can no longer increase. This might not only enable a strong heuristic, but will also help us understand cyclic dependencies in the delete relaxation.

Acknowledgments

We have received funding for this work from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 817639). Moreover, this research was partially supported by TAILOR, a project funded by the EU Horizon 2020 research and innovation programme under grant agreement no. 952215.

References

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS^+ Planning. *Computational Intelligence* 11(4): 625–655.

Bonet, B. 2013. An Admissible Heuristic for SAS^+ Planning Obtained from the State Equation. In *Proc. IJCAI 2013*, 2268–2274.

Bonet, B.; and Helmert, M. 2010. Strengthening Landmark Heuristics via Hitting Sets. In *Proc. ECAI 2010*, 329–334.

Büchner, C. 2020. *Generalization of Cycle-Covering Heuristics*. Master's thesis, University of Basel.

Büchner, C.; Keller, T.; and Helmert, M. 2021. Code, benchmarks and experiment data for the ICAPS 2021 paper "Exploiting Cyclic Dependencies in Landmark Heuristics". https://doi.org/10.5281/zenodo.4604735.

Chandrasekaran, K.; Karp, R.; Moreno-Centeno, E.; and Vempala, S. 2011. Algorithms for Implicit Hitting Set Problems. In *Proc. SODA 2011*, 614–629.

Domshlak, C.; Helmert, M.; Karpas, E.; Keyder, E.; Richter, S.; Röger, G.; Seipp, J.; and Westphal, M. 2011. BJOLP: The Big Joint Optimal Landmarks Planner. In *IPC 2011 planner abstracts*, 91–95.

Floyd, R. 1962. Algorithm 97: Shortest path. *Communications of the ACM* 5(6): 345.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2): 100–107.

Haslum, P.; Slaney, J.; and Thiébaux, S. 2012. Minimal Landmarks for Optimal Delete-Free Planning. In *Proc. ICAPS 2012*, 353–357.

Helmert, M. 2006. The Fast Downward Planning System. *JAIR* 26: 191–246.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *JAIR* 14: 253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *JAIR* 22: 215–278.

Johnson, D. 1975. Finding all the Elementary Circuits of a Directed Graph. *SICOMP* 4(1): 77–84.

Karp, R. M. 1972. Reducibility among combinatorial problems. In Miller, R. E.; and Thatcher, J. W., eds., *Complexity of Computer Computations*, 85–103. Plenum Press.

Karpas, E.; and Domshlak, C. 2009. Cost-Optimal Planning with Landmarks. In *Proc. IJCAI 2009*, 1728–1733.

Katz, M.; and Domshlak, C. 2008. Optimal Additive Composition of Abstraction-based Admissible Heuristics. In *Proc. ICAPS 2008*, 174–181.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and Complete Landmarks for And/Or Graphs. In *Proc. ECAI 2010*, 335–340.

Paul, G.; and Helmert, M. 2016. Optimal Solitaire Game Solutions using A^* Search and Deadlock Analysis. In *Proc. SoCS 2016*, 135–136.

Paul, G.; Röger, G.; Keller, T.; and Helmert, M. 2017. Optimal Solutions to Large Logistics Planning Domain Problems. In *Proc. SoCS 2017*, 73–81.

Pommerening, F.; Röger, G.; and Helmert, M. 2013. Getting the Most Out of Pattern Databases for Classical Planning. In *Proc. IJCAI 2013*, 2357–2364.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. LP-based Heuristics for Cost-optimal Planning. In *Proc. ICAPS 2014*, 226–234.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the Extraction, Ordering, and Usage of Landmarks in Planning. In *Proc. ECP 2001*, 174–182.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks Revisited. In *Proc. AAAI 2008*, 975–982.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *JAIR* 39: 127–177.

Roy, B. 1959. Transitivité et connexité. *Comptes rendus de l'Académie des Sciences* 249: 216–218.

Warshall, S. 1962. A Theorem on Boolean Matrices. *JACM* 9(1): 11–12.

Yang, F.; Culberson, J.; Holte, R.; Zahavi, U.; and Felner, A. 2008. A General Theory of Additive State Space Abstractions. *JAIR* 32: 631–662.

Zhu, L.; and Givan, R. 2003. Landmark Extraction via Planning Graph Propagation. In *ICAPS 2003 Doctoral Consortium*, 156–160.