LANDMARKS REVISITED

SILVIA RICHTER, MALTE HELMERT, MATTHIAS WESTPHA

Presented by Shane Racey

WHAT IS A LANDMARK

A landmark is a variable assignment that must be achieved at some point in every valid solution plan for a given planning task

Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (2008)

Landmarks Revisited

Silvia Richter
Griffith University, Queensland, Australia
and
NICTA, Queensland, Australia
silvia.richter@nicta.com.au

Malte Helmert and **Matthias Westphal**
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Freiburg, Germany
{helmert,westpham}@informatik.uni-freiburg.de

Abstract

Landmarks for propositional planning tasks are variable assignments that must occur at some point in every solution plan. We propose a novel approach for using landmarks in planning by deriving a pseudo-heuristic and combining it with other heuristics in a search framework. The incorporation of landmark information is shown to improve success rates and solution qualities of a heuristic planner. We furthermore show how additional landmarks and orderings can be found using the information present in multi-valued state variable representations of planning tasks. Compared to previously published approaches, our landmark extraction algorithm provides stronger guarantees of correctness for the generated landmark orderings, and our novel use of landmarks during search solves more planning tasks and delivers considerably better solutions.

Introduction

Landmarks for propositional planning were introduced by Porteous, Sebastia and Hoffmann (2001) and later studied in more depth by the same authors (Hoffmann, Porteous, and Sebastia 2003). According to their definition, *landmarks* are propositions that must be true at some point in every solution plan for a given planning task. For example, consider a Blocksworld task where the goal is to have block A stacked on block B. If some other block C is initially stacked on B, then C must be unstacked from B and B must be clear at some point for the goal to be achieved. Hence, clear(B) is a landmark for this problem instance. From the definition, goals are trivially landmarks, so on(A, B) is another landmark. We can also conclude an *ordering* on these two landmarks, denoted by clear(B) $\rightarrow$ on(A, B), indicating that block B must be clear *before* A can be stacked on it.

Hoffmann, Porteous and Sebastia propose an algorithm, called LM^{RPG} in the following, that extracts landmarks and their orderings from the relaxed planning graph of a planning task. They use landmarks in a local search procedure, called LM^{local} in the following, which searches iteratively for plans to the "nearest" landmarks, rather than searching for a plan to the goal. Their experiments demonstrate a substantial speed-up compared to an otherwise identical planner that does not use landmarks. However, they also note that the higher greediness of their search results in longer plans, and the iterative search for subgoals can fail on solvable tasks even if the underlying base planner is complete.

In this work, we propose a new way of using landmarks within a heuristic search planner which generally leads to shorter plans and an improved success rate compared to both the original LM^{local} algorithm and state-of-the-art heuristics that do not exploit landmarks. Furthermore, we present an alternative algorithm for identifying landmarks and landmark orderings, in particular for a multi-valued state variable representation of planning tasks. Unlike LM^{RPG}, our algorithm only generates sound orderings.

The rest of this paper is organised as follows: first, we give some background on planning, in particular with multi-valued state variables, and landmarks. Next, we describe our algorithms for finding and using landmarks. We then evaluate our approach experimentally, followed by a discussion of the results and a conclusion.

Notation and Background

We consider planning in the SAS^+ planning formalism (Bäckström and Nebel 1995). A concise SAS^+ representation of a planning task can be generated from a typical PDDL representation automatically (Helmert 2008).

Definition 1 ***SAS^+ planning task***
An ***SAS^+ planning task*** *is a tuple* $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ *where:*

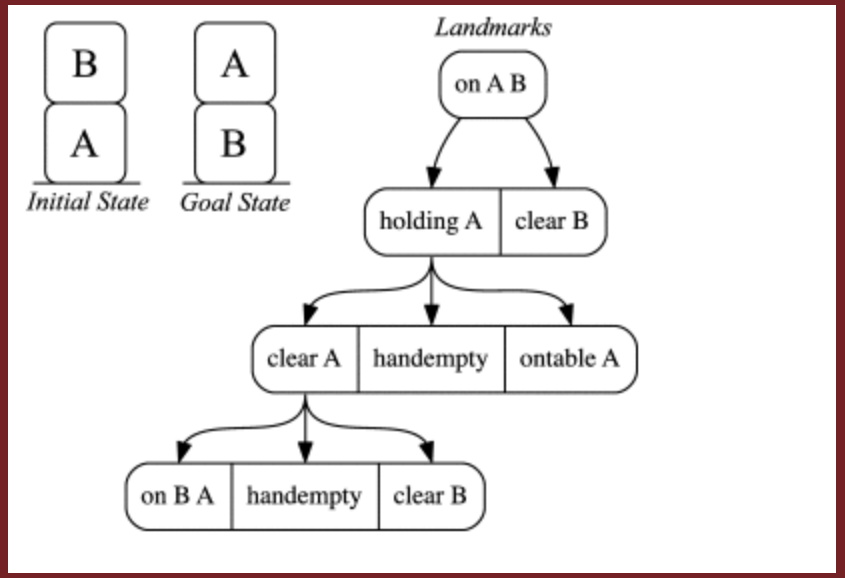
- $\mathcal{V}$ *is a finite set of* ***state variables****, each with a finite domain* $\mathcal{D}_v$*. A* ***fact*** *is a pair* $\langle v, d \rangle$ *(also written* $v \mapsto d$*), where* $v \in \mathcal{V}$ *and* $d \in \mathcal{D}_v$*. A* ***partial variable assignment*** s *is a set of facts, each with a different variable. (We use set notation such as* $\langle v, d \rangle \in s$ *and function notation such as* $s(v) = d$ *interchangeably.) A* ***state*** *is a partial variable assignment defined on all variables* $\mathcal{V}$*.*
- $\mathcal{O}$ *is a set of* ***operators****, where an operator is a pair* $\langle pre, eff \rangle$ *of partial variable assignments.*
- s_0 *is a state called the* ***initial state****.*
- $s_\star$ *is a partial variable assignment called the* ***goal****.*

An operator $o = \langle pre, eff \rangle \in \mathcal{O}$ *is* ***applicable*** *in state* s *iff* $pre \subseteq s$*. In that case, it can be* ***applied*** *to* s*, which produces the state* s' *with* $s'(v) = eff(v)$ *where* $eff(v)$ *is defined and* $s'(v) = s(v)$ *otherwise. We write* $s[o]$ *for* s'*. For operator sequences* $\pi = \langle o_1, \ldots, o_n \rangle$*, we write* $s[\pi]$ *for* $s[o_1] \ldots [o_n]$

EXAMPLE

- Blocks world task
- 2 Blocks (A, B)
- Initial State: Block B on top of Block A
- Goal: Block A on top of Block B

- Clear(A) is a landmark since it must be cleared for the goal to be achieved
- on(A, B) is also considered a landmark
- Ordering: clear(B) --> on(A, B)



PREVIOUS LANDMARK EXTRACTION

- Previously, Hoffmann proposes two landmark algorithms LM^{RPG} and LM^{local}

LM RPG (Relaxed Planning Graph)

- Combines landmarks with a relaxed planning graph to extract more relevant and stronger landmarks

LM Local

- Improves heuristic guidance by considering local dependencies between landmarks

LM RPG IN SAS+ PLANNING

Step 1: Construct the Relaxed Planning Graph

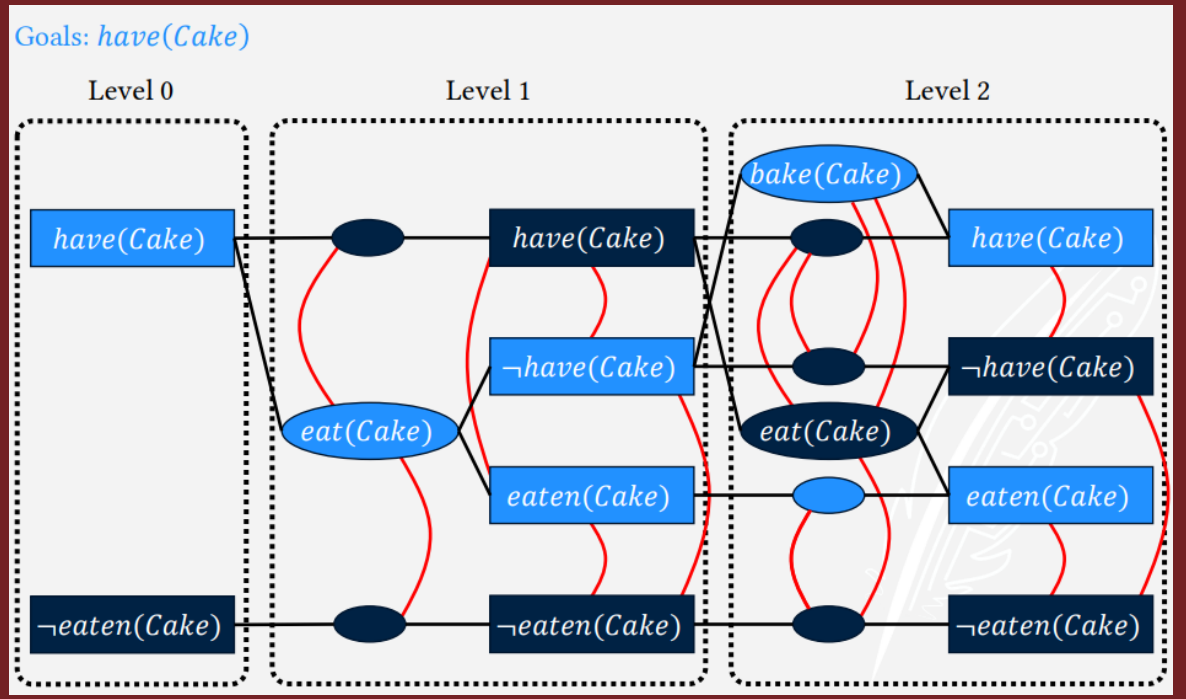
- Si: Over-approximation of facts reachable in i steps
- Oi: Precisely the operators that become applicable at i but not before

Step 2: Extracting Landmark Candidate

- If all first achievers of landmark B share a precondition A, then A → B

One-Step Lookahead

- If first achievers don't share a precondition, check their own preconditions
- If these share a fact A, then A is a landmark at least two steps before B



$$S_i := \begin{cases} s_0 & i = 0 \\ S_{i-1} \cup \bigcup_{\langle pre, eff \rangle \in \mathcal{O}_{i-1}} eff & i > 0 \end{cases}$$

$$O_i := \{\langle pre, eff \rangle \in \mathcal{O} \mid pre \subseteq S_i\} \setminus \bigcup_{j<i} O_j.$$

LM RPG IN SAS+ PLANNING

Disjunctive and Conjunctive Landmarks

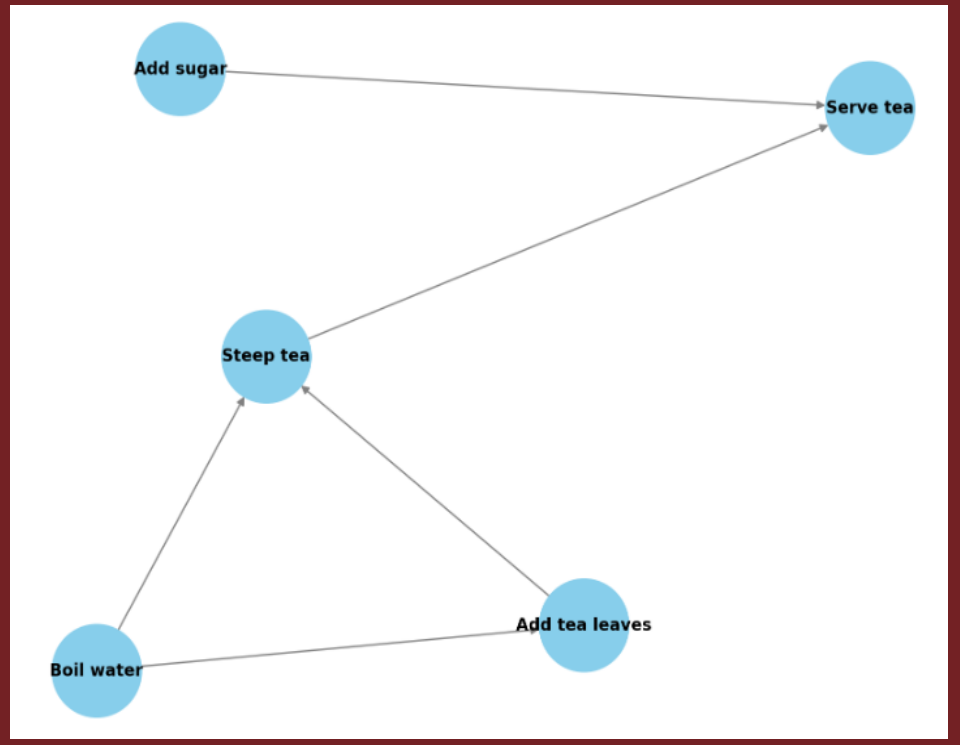
- A fact A is a conjunctive landmark if A must be true in every valid plan before a certain action or goal is achieved
- A set of facts {L1, L2, ..., Ln} is a disjunctive landmark if at least one of these facts must be achieved before the goal can be reached

Filtering Out False Landmarks (no gaurantee soundness)

- Test: Remove all actions achieving A and check if a relaxed solution exists
- If no solution, A is a true landmark
- If solution exists, A is rejected

LM LOCAL

- Build Landmark Graph: Create a directed acyclic graph with landmarks as nodes and orderings as arcs
- Disjunctive Goal: Use source landmarks (no incoming arcs) to generate a subgoal for the planner
- Generate Plans: Planner achieves one landmark, removes it, and updates the graph
- Repeat: Continue until all landmarks are achieved, then plan to satisfy the original goal



NEW FORMULATION OF LANDMARKS

Slight Changes

- No one step look ahead, opt to admit disjunctive landmarks
- All facts stem from same predicate symbol, and discard sets greater than 4

Domain Transition Graphs

- Nodes represent possible values of a state variable
- Edges represent actions that transition the variable from one value to another
- DTGs highlight variable values that must be reached before achieving a goal
- Locked → Closed → Open (value transition "Closed" is unavoidable→landmark)

PSEUDO-HEURISTIC LANDMARKS

Pseudo-heuristic

- Landmarks cannot be a heuristic on their own
- They are useful to guide the search and combine with other heuristics

Ways to combine them

- Landmarks L that still need to be achieved: L := n – m + k
- Preferred operators: an operator is preferred in a state if applying it achieves an acceptable landmark in the next step
- Use the Fast Downward planner as framework gives equal importance to all heuristics

EXPERIMENTS AND RESULTS

PERCENTAGE OF TASKS SOLVED

- **Base planners:** FF heuristic, Causal Graph heuristic, blind heuristic
- **Methods for using landmarks:** base planner using no landmarks, LM local algorithm, landmark pseudo-heuristic
- Bold results indicate better performance than the other two methods for a given base planner and domain
- In all cases, LM RPG algorithm was used for generating landmarks and orderings

Domain	FF heuristic			CG heuristic			blind heuristic		
	base	local	heur	base	local	heur	base	local	heur
Airport (50)	**72**	32	64	46	32	**48**	34	32	**64**
Assembly (30)	100	97	100	10	**97**	83	0	**97**	7
Blocks (35)	100	100	100	100	100	100	43	100	100
Depot (22)	86	**100**	95	45	100	100	9	100	**95**
Driverlog (20)	100	100	100	100	100	100	25	100	100
Freecell (80)	95	80	**98**	89	80	**98**	16	80	**98**
Grid (5)	100	100	100	80	100	100	20	100	100
Gripper (20)	100	100	100	100	100	100	25	100	100
Logistics-1998 (35)	94	100	100	100	100	100	6	**100**	97
Logistics-2000 (28)	100	100	100	100	100	100	36	100	100
Miconic (150)	100	100	100	100	100	100	27	100	100
Miconic-FullADL (150)	91	91	90	89	**91**	90	41	**91**	42
Miconic-SimpleADL (150)	100	100	100	100	100	100	37	100	100
MPrime (35)	89	80	**97**	100	80	100	37	80	86
Mystery (30)	53	53	**57**	57	53	**60**	37	53	53
Openstacks (30)	100	100	100	70	100	100	23	100	100
OpticalTelegraphs (48)	4	**8**	4	2	**8**	6	2	8	**100**
Pathways (30)	93	**100**	97	23	100	100	13	100	100
Philosophers (48)	96	67	**100**	100	67	100	8	67	**73**
Pipesworld-NoTankage (50)	84	78	**88**	48	78	**84**	22	78	78
Pipesworld-Tankage (50)	78	58	**86**	28	58	**64**	12	58	**66**
PSR-Large (50)	64	**66**	64	64	**66**	62	20	**66**	64
PSR-Middle (50)	100	100	100	100	100	100	50	100	100
PSR-Small (50)	100	100	100	100	100	100	94	100	100
Rovers (40)	100	100	100	80	100	100	10	100	100
Satellite (36)	97	97	97	97	97	97	11	97	97
Schedule (150)	99	61	100	99	61	**100**	7	61	**94**
Storage (30)	63	63	60	**67**	63	63	40	**63**	57
TPP (30)	100	100	100	77	100	100	17	100	100
Trucks (30)	40	3	40	30	3	30	13	3	**23**
Zenotravel (20)	100	100	100	100	100	100	35	100	100
Averaged over domains	87	82	**88**	74	82	**87**	25	82	**84**

Domain	FF heuristic	
	base	heur
Airport (50)	**6**	2
Depot (22)	0	**2**
Freecell (80)	1	**3**
Logistics-1998 (35)	0	**2**
Miconic-FullADL (150)	**2**	0
MPrime (35)	0	**3**
Mystery (30)	0	**1**
Pathways (30)	1	**2**
Philosophers (48)	0	**2**
Pipesworld-NoTankage (50)	0	**2**
Pipesworld-Tankage (50)	1	**5**
Schedule (150)	0	**1**
Storage (30)	**1**	0
Total	12	**25**

COMPARING THE NUMBER OF TASKS SOLVED

- Exclusively he FF-heuristic base planner and the landmark pseudo-heuristic
- An entry of n for a given approach and domain means that the approach solved n tasks in this domain which the other approach did not solve

NUMBERS OF LANDMARKS AND ORDERINGS

New generation method: (disjunctive landmarks, landmarks found by the domain transition graph criterion)

Bold results indicate largest number of landmarks/orderings found in a given domain across the three approaches

Domain	Hoffmann et al.		Zhu & Givan		New generation method	
	LMs	Orderings	LMs	Orderings	LMs (Disj./DTG)	Orderings
Airport (50)	**42614**	294965	37156	73850	38203 (1014/7287)	**1459285**
Depot (22)	1420	4937	1240	2629	**1440** (159/179)	**6961**
Freecell (80)	**8448**	38809	7855	13700	7716 (0/2834)	**95330**
Gripper (20)	960	1400	960	1380	**1420** (460/460)	**2780**
Logistics-1998 (35)	2374	5261	2177	1965	**2909** (732/1230)	**8167**
Miconic-SimpleADL (150)	6583	8676	**10045**	**11469**	6583 (0/80)	10762
MPrime (35)	**199**	159	132	92	164 (44/51)	**198**
Rovers (40)	**2827**	1946	1565	785	2338 (379/2)	**2095**
Schedule (150)	8572	6508	7555	5491	**11530** (0/2958)	**9466**
Total	121056	433977	**153370**	345515	140630 (4977/19052)	**2104220**

Domain	base vs. local-HPS	base vs. heur-HPS	base vs. heur-ZG	base vs. heur-RHW
Airport (50)	**4**/0 (+6%)	0/**14** (−1%)	2/**14** (−2%)	0/**14** (−1%)
Assembly (30)	16/**23** (±0%)	13/**16** (±0%)	30/**43** (−1%)	0/0 (±0%)
Blocks (35)	**57**/37 (+22%)	34/**51** (−7%)	20/**48** (−3%)	20/**65** (−13%)
Depot (22)	40/**45** (−11%)	27/**45** (−5%)	22/**45** (−16%)	22/**54** (−17%)
Driverlog (20)	45/45 (−1%)	35/**60** (−4%)	30/**60** (−4%)	30/**60** (−5%)
Freecell (80)	**57**/6 (+12%)	**61**/23 (+5%)	**81**/7 (+20%)	**72**/10 (+12%)
Grid (5)	**40**/20 (+5%)	40/40 (+3%)	**60**/20 (+4%)	20/**60** (−7%)
Gripper (20)	**100**/0 (+7%)	**100**/0 (+4%)	0/0 (±0%)	0/**100** (−23%)
Logistics-1998 (35)	**71**/14 (+5%)	31/**37** (−1%)	**57**/20 (+2%)	14/**60** (−3%)
Logistics-2000 (28)	**96**/0 (+13%)	**46**/7 (+2%)	**60**/14 (+3%)	**32**/14 (+1%)
Miconic (150)	6/**80** (−8%)	0/**96** (−17%)	23/**56** (−2%)	0/**96** (−18%)
Miconic-FullADL (150)	23/**25** (±0%)	8/**30** (−1%)	**62**/20 (+6%)	**50**/28 (+4%)
Miconic-SimpleADL (150)	**96**/0 (+28%)	**58**/28 (+4%)	6/**80** (−9%)	**56**/29 (+3%)
MPrime (35)	14/**20** (+2%)	5/**22** (−6%)	5/**22** (−6%)	8/**28** (−6%)
Mystery (30)	3/**23** (−5%)	0/**26** (−9%)	0/**26** (−9%)	0/**26** (−12%)
Openstacks (30)	**86**/0 (+3%)	**86**/0 (+3%)	**76**/0 (+2%)	**76**/0 (+2%)
OpticalTelegraphs (48)	0/0 (±0%)	0/0 (±0%)	0/0 (±0%)	0/0 (±0%)
Pathways (30)	33/33 (+1%)	23/**40** (±0%)	26/**36** (±0%)	30/**33** (±0%)
Philosophers (48)	**60**/2 (+25%)	**25**/4 (+3%)	**22**/4 (+2%)	**25**/4 (+3%)
Pipesworld-NoTankage (50)	**36**/24 (+9%)	18/**40** (−3%)	22/**42** (−3%)	30/**42** (±0%)
Pipesworld-Tankage (50)	**24**/22 (+8%)	22/**40** (+4%)	**38**/32 (+20%)	22/**42** (±0%)
PSR-Large (50)	10/**24** (−5%)	8/**14** (−2%)	**14**/12 (−1%)	12/12 (+1%)
PSR-Middle (50)	2/**32** (−5%)	6/**18** (−2%)	12/**18** (±0%)	12/**14** (+3%)
PSR-Small (50)	**14**/0 (+4%)	**4**/0 (+1%)	0/0 (±0%)	**2**/0 (±0%)
Rovers (40)	22/**62** (−2%)	22/**50** (−2%)	22/**52** (−2%)	22/**55** (−2%)
Satellite (36)	25/**55** (−6%)	2/**63** (−10%)	11/**61** (−7%)	22/**50** (−6%)
Schedule (150)	16/**33** (−2%)	20/**66** (−8%)	32/**58** (−3%)	20/**66** (−8%)
Storage (30)	**46**/0 (+44%)	**16**/0 (+22%)	6/6 (+1%)	**26**/6 (+25%)
TPP (30)	**86**/0 (+32%)	13/**50** (−3%)	13/**50** (−3%)	16/**46** (−2%)
Trucks (30)	**3**/0 (+10%)	6/6 (±0%)	**13**/3 (+1%)	**13**/10 (+1%)
Zenotravel (20)	**60**/10 (+14%)	**40**/10 (+4%)	**40**/20 (+3%)	25/**45** (+0%)
Averaged over domains	**38**/20 (+6%)	25/**29** (−1%)	26/**28** (−1%)	22/**34** (−3%)

PLAN LENGTH COMPARISON

- Each column compares the base planner to a configuration using landmarks
- An entry like "71/14 (+5%)" indicates that the base planner found a shorter plan than the landmark configuration for 71% of the instances and a longer plan for 14% of the instances
- The number in parentheses indicates that the plans generated by the landmark approach were 5% longer on average

RICHTER'S LANDMARK-BASED PSEUDO-HEURISTIC EXAMPLE

EXAMPLE

Initial State:

- Water is in the kettle
- The kettle is off
- The tea bag is in the cupboard
- The cup is empty

Goal State:

- Tea is ready

Available Actions:

- **Turn on the kettle**: Requires the kettle to have water
- **Boil water**: Requires the kettle to be on
- **Pour hot water into the cup**: Requires boiled water
- **Add a tea bag to the cup**: Requires a cup
- **Steep tea**: Requires hot water and a tea bag in the cup

IDENTIFYING LANDMARKS USING LM RPG

- A Relaxed Planning Graph is built by ignoring negative effects (delete relaxation)

- Use LM RPG to generate landmarks
- Estimate the goal distance of a state s by the number of landmarks l that still need to be achieved
- Combine with preferred operators
- Use alongside of FF heuristic to get better results than FF alone

COUNTING UNACHIEVED LANDMARKS

State	Unachieved Landmarks	Heuristic Value
Initial State	4 (Boil water, Pour, Add tea bag, Steep)	4
After turning on kettle	4 (Boil water, Pour, Add tea bag, Steep)	4
After boiling water	3 (Pour, Add tea bag, Steep)	3
After pouring water	2 (Add tea bag, Steep)	2
After adding tea	1 (Steep)	1
Goal state	0	0

The heuristic value is the count of remaining unachieved landmarks. The planner prioritizes actions that reduce the number of remaining landmarks, guiding the search effectively

HOW THIS HELPS SEARCH

- Richter's pseudo-heuristic counts remaining landmarks instead of estimating cost. Improves search efficiency by reducing wasted exploration of states missing preconditions
- Guides planning by necessity ensuring critical steps are done early
- Combines well with other heuristics like FF to refine estimates.

THANK YOU

https://www.sciencedirect.com/science/article/pii/S0004370219300013