DALAI – Disjunctive Action Landmarks All In

Clemens Büchner[1], Remo Christen[1], Salomé Eriksson[1], Thomas Keller[1,2]

[1]University of Basel, Switzerland
[2]University of Zürich, Switzerland
{clemens.buechner, remo.christen, salome.eriksson, tho.keller}@unibas.ch

This is the planner abstract for DALAI, a reincarnation of LAMA (Richter and Westphal 2010). DALAI uses related concepts, in particular landmarks, but adds recent findings to the mix and implements most landmark-specific parts from scratch. Moreover, it goes all in betting on landmarks and rejects complementary ideas. Our implementation is based on version 22.12 of Fast Downward (Helmert 2006).

Mythology

The first written evidence of the term *landmark* in the planning context can be found in an ancient manuscript that dates back to the very early days of plannerkind (Porteous, Sebastia, and Hoffmann 2001). These early plan-hunters and gatherers observe the existence of *(fact) landmarks* (also called *atoms*) that are encountered in any successful plan-chase, and they use *orderings* among the landmarks to decompose the cumbersome hunt into subproblems that are easier to tackle and have the landmarks as subgoals. However, as this hunting technique is incomplete, they often returned without success despite the presence of plans in their surroundings.

Another scroll (Zhu and Givan 2003), which was written hundreds of days later, indicates that the next step of plannerkind was to raise the concept from its initial, fact-based representation to a methodology that describes which *activities* (or *actions*) are necessary in any successful plan-gathering. The same scroll also describes that those early planners no longer used landmarks for decomposition purposes, but to come up with an estimate of the effort that is required until the plan can be reaped successfully. Further artifacts of these ancient technologies provide evidence that the ideas were expanded to disjunctive sets of atoms (Richter, Helmert, and Westphal 2008), conjunctive sets of atoms (Keyder, Richter, and Helmert 2010), arbitrary propositional formulas over atoms (Karpas and Domshlak 2009), or disjunctive sets of actions (Bonet and Helmert 2010).

The rise of landmarks on the planning horizon climaxed with the creation of the LAMA planner (Richter and Westphal 2010), which emerged victorious from the international planning competition (IPC) 2008 and certified its dominance in the year 2011 with another victory. Nowadays, over a decade after the reign of LAMA, its spirit lives on while its physical remains (i.e., the source code) could not keep up with the advancement of time. During these years, many other, much younger spirits started to show their powers, leaving LAMA, as legendary as the stories are, a mere memory of today's planning society.

Nevertheless, researchers from all over the world continued to study the topic of landmarks; recent findings suggest that LAMA did—despite its success—not live up to its full potential. Here, we describe our aim at taking a step on this unsatisfactory and painful path to true enlightenment, which manifests in the LAMA reincarnation DALAI.

Ideology

> *The goal is not to be better than the other ~~man~~ [planner], but your previous self.*
>
> —Dalai Lama

With DALAI, we revive the spirit of LAMA in a new shape. Our planner is inspired by the many concepts responsible for LAMA's success, first and foremost the use of landmarks. However, we revise crucial parts of LAMA's implementation such as the representation of landmarks; in LAMA, a landmark is a set of atoms such that for every plan it holds that one atom from every landmark is true in some state along this plan's trajectory. In contrast, DALAI considers sets of actions, such that every plan contains at least one action from every landmark. The usage of these so-called *disjunctive action landmarks* provides the first three letters of the name of our planner.

The last two letters of the planner name stem from our commitment on creating a *pure landmark-based planner* rather than a portfolio of successful techniques: DALAI goes *all-in* and renounces all components of LAMA that are not related to landmarks. Specifically, LAMA's best-first search alternates between two open lists whose order is based on two different heuristics, namely the landmark count heuristic and the delete-relaxation based FF heuristic (Hoffmann and Nebel 2001). While techniques that are based no the idea of a delete-relaxed task are among the most important methods to generate landmarks (Zhu and Givan 2003; Richter, Helmert, and Westphal 2008), the FF heuristic itself is not related to landmarks. Therefore, all versions of DALAI are guided by a single heuristic that is based on landmarks.

Technology

Optimism doesn't mean that you are blind to the reality of the situation. It means that you remain motivated to seek a solution to whatever problems arise.

—Dalai Lama

Büchner et al. (2023) suggest the LM-BFS framework and describe how the landmark component of LAMA is an instantiation of it. DALAI also fits that framework which describes a best-first search based on landmarks. The LM-BFS framework offers several degrees of freedom to achieve different behaviour of the search: (a) landmark generation method, (b) landmark progression function, (c) landmark heuristic, and (d) search algorithm. We continue by briefly discussing these components and related topics, focusing on the context of DALAI. A discussion of the explicit choices we made for these components for the submission of DALAI to the different tracks of IPC 2023 follows in the next section. Note that Büchner et al. (2023) restrict their discussion of LM-BFS to landmarks as propositional formulas over state features (i.e., atoms). Since DALAI operates on disjunctive action landmarks, though, we need to transfer the theory to this view.

Landmark Generation

In order to become prosperous, a ~~person~~ [planner] must initially work very hard, so ~~he or she~~ [it] has to sacrifice a lot of leisure time.

—Dalai Lama

Hoffmann, Porteous, and Sebastia (2004) show that deciding whether a given atom is a landmark is PSPACE-complete. Nevertheless, there are efficient algorithms that allow to come up with some specific landmarks. In DALAI we use the technique proposed by Richter, Helmert, and Westphal (RHW; 2008) which computes landmarks in the delete relaxation and additionally determines *natural* and *greedy-necessary orderings* between these landmarks. In a subsequent step, we add *reasonable orderings* the same way as Hoffmann, Porteous, and Sebastia (2004) do.

Fast Downward implements both of these techniques in a way that generates disjunctive fact landmarks. Is is possible to transform a disjunctive fact landmark ℓ into a disjunctive action landmark by considering the set of *achieving actions* of each atom in ℓ. Indeed, this is what we do in some configurations. Alternatively, we implemented a direct computation of both disjunctive action landmarks and orderings between them.

In both cases, the reasoning for greedy-necessary orderings needs special care. By definition, a greedy-necessary ordering between landmarks l_1 and l_2 denotes that l_2 can only be achieved from states where l_1 *holds*. Since it is not properly defined for a disjunctive action landmark what it means for l_1 to hold, we need to work around this somehow. Our solution is to store for all such orderings the set of atoms l_1 achieves.

Our variant computing disjunctive action landmarks directly adapts several design aspects to better fit the action landmark world, as well as to increase the amount of landmarks and orderings found in the hope of providing better guidance during the search.

Allow overlapping landmarks. RHW enforce that $l_i \cap l_j = \emptyset$ for all considered fact landmarks l_i and l_j. We lift this restriction for two reasons: first, the translation to disjunctive action landmarks already leads to overlap (for example if we have two fact landmarks $\{v_1\}$ and $\{v_2\}$ and an action a achieving both, then the action landmarks derived from $\{v_1\}$ and $\{v_2\}$ both contain a); and second, we consider different ways of computing a heuristic value from landmarks, some of which do not rely on mere *counting* of unachieved landmarks (for which we assume that overlap-free landmarks are particularly well-suited).

Allow larger shared preconditions. RHW limit the size of shared preconditions to 4, while we turned the limit into an adjustable parameter.

Detect more cases of interfering landmarks. The procedure by Hoffmann, Porteous, and Sebastia (2004) deduces reasonable orders when landmarks *interfere* with each other. The implementation in Fast Downward requires one of the disjunctive fact landmarks to consist of only one fact, but we allow arbitrary disjunctive action landmarks.

Allow cycles in the landmark graph. LAMA enforces that no cyclic dependencies between landmarks occur, while DALAI does not require this restriction.

Progression

Reason well from the beginning and then there will never be any need to look back with confusion and doubt.

—Dalai Lama

Computing good landmarks and orderings is prohibitively expensive. Therefore, the LM-BFS framework by Büchner et al. (2023) advocates computing landmarks once for the initial state and *progressing* them within the search whenever expanding new transitions. This idea was first suggested in the context of the path-dependent landmark heuristic (Richter, Helmert, and Westphal 2008) and later strengthened by combining information from multiple paths (Karpas and Domshlak 2009).

Since Büchner et al. (2023) use landmarks based on propositional formulas over state features, their landmark progressions reason about whether a certain landmark holds in a given state or not. We consider disjunctive action landmarks, where things are not as expensive. Instead, we just need to know which landmarks a given action satisfies. This is easy to test, as these are simply the landmarks that contain that action. In the terminology of Büchner et al., we mark a landmark *past* when applying one of its operators. Landmark orderings of a certain type mark landmarks *future* according to the same rules used by Büchner et al..

Heuristics

Although you may not always be able to avoid difficult situations, you can modify the extent to which you can suffer by how you choose to respond to the situation.

—Dalai Lama

Zhu and Givan (2003) suggest to count the landmarks that need to be achieved in a given state as heuristic for the goal distance. This heuristic became known as the *landmark*

count heuristic and is also used in LAMA. Its implementation in Fast Downward was recently renamed to *landmark sum* (h^{sum}), as it was changed to take action costs of the achievers into account and is therefore not purely counting landmarks anymore. For this heuristic, it does not matter what kinds of landmarks are considered, so its idea carries over easily to the disjunctive action landmark view.

However, the more the (action) landmarks overlap, the less accurate h^{sum} becomes since it does not account for achieving several landmarks with the same action. We thus implemented the *greedy hitting set heuristic* (h^{ghs}), which greedily computes a hitting set, i.e., a set of actions A such that for all landmarks l we have $l \cap A \neq \emptyset$ (the landmark is "hit" by A). The heuristic is greedy in that it iteratively adds an action to A that maximizes the amount of landmarks that newly "are hit". Since the hitting set is not *minimal*, this still leads to inadmissible estimates and is thus only used in satisficing settings.

Computing *minimum* hitting sets would denote an admissible heuristic. However, Büchner, Keller, and Helmert (2021) show that we can do even better using the *cyclic landmark heuristic* (h^{cycle}). Besides the landmarks, h^{cycle} also considers the orderings between them, in particular the cyclic dependencies induced by the orderings. Büchner, Keller, and Helmert prove that one of the landmarks in every cycle needs to be achieved at least twice, and that their heuristic dominates the minimum hitting set approach.

Preferred Operators

In order to carry out a positive action we must develop a positive vision.

—Dalai Lama

Analogous to LAMA, we use alternating open lists where one is populated by preferred operators. In our disjunctive action landmark framework, the definition of preferred operators is straightforward: an operator is preferred if it is both applicable in the current state, and is contained in a landmark that must still be satisfied.

To choose between the open lists, we use the boost mechanism of LAMA. Instead of using the default boost value of 1000, we individually adapt the value for the two non-optimal tracks based on empirical results.

Justified Action Applications

The important thing is that ~~men~~ [actions] should have a purpose in life. It should be something useful, something good.

—Dalai Lama

The notion of *unjustified* action applications stems from Simon and Röger (2015), who in turn based the idea on intended effects introduced by Karpas and Domshlak (2012).

Intuitively, an action application is only justified if at least one of its add effects satisfies the following two conditions: (i) it is part of the goal or enables a future action application, and (ii) it is neither deleted nor made true by a future action application until the goal is reached or the enabled action application is performed. Any non-zero-cost action application that is not justified according to these criteria can never be part of an optimal plan, thus we can safely disregard paths that contain them.

We integrate a relaxed version of this idea into DALAI by casting Condition (i) into a disjunctive action landmark. For every action a whose effects do not contain a goal atom, we compute the set of actions S_a such that every action in S_a has at least one of a's add effects as a precondition. Each set S_a forms a disjunctive action landmark that is active on every path that contains action a and thus ensures that the application of a satisfies Condition (i). In other words, as soon as an action is applied we enforce that its application must later be justified, according to our relaxed notion, by a future applied action.

As the resulting landmarks can contain thousands of actions and thus become both large and uninformative, we implement an ad hoc optimization that discards all landmarks that contain more than 1000 actions.

Methodology

Action is more important than meditation.

—Dalai Lama

We participate with DALAI in all three tracks of IPC 2023. However, we use different versions tailored to the different objectives for each track. The following objectives are used in the competition:

Optimal Find optimal plans for as many problems as possible.

Satisficing Find plans for as many problems as possible, aiming for cheap plans.

Agile Find plans for as many problems as possible as fast as possible.

We experimentally tested different combinations of concrete implementations and parameter choices on the benchmarks from previous iterations of the IPC. The following paragraphs describe the configurations as submitted to the competition and some words of wisdom on why we think they might be suitable for the corresponding objectives.

Optimal

We use A*search (Hart, Nilsson, and Raphael 1968) with h^{cycle} as an admissible heuristic based on linear programming (LP; Büchner, Keller, and Helmert 2021). Landmarks and orderings are generated using the LM-cut procedure[1] (Helmert and Domshlak 2009; Bonet and Helmert 2010) as well as the RHW landmark generation method (Richter, Helmert, and Westphal 2008). For the latter, we use the original implementation based on fact landmarks and translate the result to disjunctive action landmarks. We use Johnson's cycle detection algorithm (Johnson 1975). Note that this configuration does not support conditional effects and aborts without finding a solution when parsing a problem file with conditional effects.

[1] only landmarks, no orderings

Satisficing

We follow LAMA which uses an iterative *anytime search* procedure (Richter and Westphal 2010). It sequentially starts one search after the other on the same input, using the solution cost found by the first search as an upper bound on the solution cost for the second one and so on. The procedure stops when either time or memory runs out, an iteration cannot find an improving solution compared to the previous one, or the last iteration found a plan. In any case, the last plan found (if any) is reported as the final solution.

As a heuristic, this configuration uses h^{ghs} with preferred operators and a boost value of 1. Conditional effects are supported. Landmarks are generated using our disjunctive action landmark variant of RHW. Furthermore, some iterations of the search consider justified action applications. We start with a lazy greedy best-first search without justified action application to find *some* plan quickly. In tasks with non-uniform action costs, we continue with a lazy greedy best-first search with action costs considered, but shifted by $+1$ to prefer short plans over long ones with 0-cost actions. Next, it follows a sequence of weighted A*searches with declining weights (5, 3, 2, 1). Action costs are again transformed in the same manner in case of non-uniform action costs. In case of uniform action costs, the search stops at this point. Otherwise, we append a last iteration of A*with the original action costs.

Agile

Since this track is all about speed, we use h^{sum} as the heuristic which is much faster to evaluate than h^{ghs} or h^{cycle}. Conditional effects are supported. Furthermore, we transform costs to unit-costs to find short paths rather than cheap ones. The search is lazy greedy best-first search and considers preferred operators but without boosting them upon progress in the heuristic value.

Roentgenology

Even when we have ~~physical~~ [computational] hardships, we can be very happy.

—Dalai Lama

In this section we diagnose the performance of DALAI in the IPC 2023. We again split according to the different tracks of the competition. Note that the competition used multiple variants of the same problem files for some domains to compile away certain PDDL features not supported by some of the planners, and then used the best results among the different variants for the scoring. In this analysis, we just use the original version of the problems. This means our results might differ from those of the competition even though we use a similar setup in most aspects.[2]

Optimal

DALAI failed to surprise us in its optimal configuration. Since our heuristic solves an LP in every evaluated state, we expected a slow planner with low coverage compared to the competition. DALAI ended up close to the end of the ranking, on place 21 of 22 competition entries.

[2]For DALAI, normalization only seems to make a minor difference in the `slitherlink` domain anyway.

	DALAI	LM-A* opt.	LM-A* unif.	LM-cut	BJOLP
`folding`	2	2	**6**	3	5
`labyrinth`	**1**	**1**	**1**	**1**	**1**
`quantum-layout`	11	12	**13**	12	**13**
`recharging-robots`	**6**	1	2	1	2
`ricochet-robots`	4	1	**8**	5	6
`rubiks-cube`	0	9	**10**	0	0
`slitherlink`	0	0	0	0	0
Total (140)	24	26	**40**	22	27
Timeout	63	67	43	51	46
Memory out	0	0	10	0	0
Unsupported PDDL features	20	14	14	34	34
Translate Memory out	32	33	33	33	32

Table 1: Coverage results and reasons for failure for different landmark-based optimal planning approaches.

To see whether solving LPs really is the limitation we face, we compare to other optimal planning approaches with landmarks from the literature. In particular, we consider LM-A*(Karpas and Domshlak 2009) with uniform and optimal cost partitioning, LM-cut (Helmert and Domshlak 2009; Bonet and Helmert 2010), and BJOLP (Domshlak et al. 2011). Out of these, LM-A*with optimal cost partitioning and BJOLP also use LPs to determine the heuristic value of a state. Table 1 shows results for these approaches.

Indeed, all LP-based approaches solve similarly many problems, namely between 24 and 27. With uniform cost partitioning, however, 40 problems are solved. This indicates that LPs are indeed a limiting factor. Surprisingly, LM-cut solves the fewest tasks even though it is generally regarded as a solid heuristic. Part of the reason for this are probably the unsupported PDDL features, which also occur for other planners, though. All systems suffer similarly from the Fast Downward translator running out of memory while grounding 32 to 33 problems.

Among all approaches we consider here, DALAI solves noticeably more problems than the others in `recharging-robots`. This is only due to the fact that this domain has axioms in most problems and DALAI is the only configuration which does not complain about them.

Satisficing

Since our satisficing configuration is similar to LAMA, which was used as a baseline in the competition, we focus our analysis on comparing against it. Unfortunately we could not live up to our master: LAMA came in 4th place while we landed in the middle field (13 out of 23).

A major difference between DALAI and LAMA is that the latter utilizes the h^{FF} heuristic in addition to its landmark

	DALAI	LAMA−FF	DALAI+FF	LAMA
folding	6	**11**	10	**11**
labyrinth	**4**	**4**	3	1
quantum-layout	19	18	19	**20**
recharging-robots	13	13	**14**	**14**
ricochet-robots	12	**17**	10	14
rubiks-cube	5	3	**20**	**20**
slitherlink	0	0	0	0
Total (140)	59	66	76	**80**
Cost	1426	1353	1404	**1320**
Iterations	279	**332**	293	306

Table 2: Satisficing coverage of DALAI and LAMA. Cost shows the sum of all reported plans across commonly solved problems. Iterations sums for each commonly solved problem how many of the searches in the iterated search were started.

heuristic, which leads to the assumption that h^{FF} is pulling a lot of the weight. To verify this claim, we include two more configurations in our analysis: LAMA with only the landmark heuristic (LAMA−FF) and DALAI with the h^{FF} heuristic added (DALAI+FF).[3]

Table 2 compares the four configurations in terms of how many tasks were solved, as well as the sum of all plan costs and started iterations across all commonly solved tasks. The started iterations give us a sense how far in the iterated search each configuration got. As we can see, h^{FF} is indeed boosting coverage significantly and brings us close to the performance of LAMA. This is particularly true for the `rubiks-cube` domain, where DALAI and LAMA−FF solve only 3 and 5 tasks respectively, but both configurations with h^{FF} solve all 20 tasks.

In terms of cost however, DALAI+FF fares worse than even LAMA−FF. This is due to the fact that both DALAI style configurations have significantly less iterations, meaning they less often reach the weighted A* iterations which bound cost to a factor of the optimal cost. But why is this the case? One might assume that heuristic guidance is worse, but looking at the first plot in Figure 1, which compares expansions between DALAI and LAMA−FF in the first search, we see that the h^{ghs} heuristic used in DALAI leads to significantly less expansions. Unfortunately, this better guidance comes at too steep a price, as can be seen by the much slower expansion rate shown in the second plot in Figure 1. Even though computing a greedy hitting set can be done in polynomial time, it is still significantly slower than just summing up costs. This is especially true when landmarks have a high number of achievers.

One example is the domain `ricochet-robots`, which is an interesting case study because it is responsible for most

[3]Note the *minus* in LAMA−FF and the *plus* in DALAI+FF.

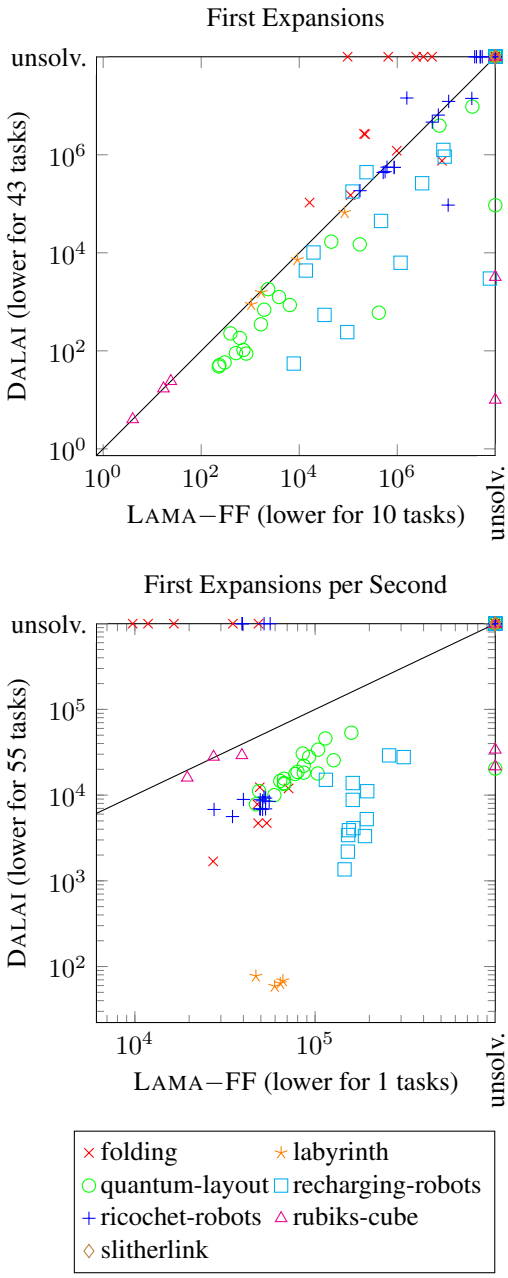

Figure 1: Comparison of expansions and expansions per second in the first iteration of DALAI and LAMA−FF respectively.

of the coverage loss between LAMA and DALAI+FF. Furthermore, h^{FF} has a negative influence, dropping 2 tasks for the DALAI configurations and 3 tasks for the LAMA configurations. In this domain, robots need to reach their goal on a 2D grid, but once they move they are forced to continue moving in the same direction until they hit an obstacle. The domain is encoded such that an action only moves a robot one cell and uses guard variables to ensure that the robot can only stop at obstacles. One such variable is `nothing-is-moving`, which is also a goal variable that ensures that we actually stop at the goal location. It is thus also a landmark, and has 4400 achievers. Furthermore, since one move is done over several states it is not true in most of the states we encounter, and we thus need to consider it most of the time when computing the greedy hitting set.

This performance problem is exacerbated by the rather poor heuristic guidance. Landmarks are found by backtracking from the goal, which in a 2D grid problem tells us that the robot must be in one of the neighboring cells before reaching the goal. Due to the increased maximal disjunctive landmark size in DALAI we can backtrack a bit further than LAMA, but these landmarks are still very close to the goal zone and thus do not offer much guidance, while also having more achievers since the disjunction is bigger the further we backtrack. Furthermore, the domain is encoded in such a way that most states have only one successor (namely when a robot is in the middle of moving), meaning there are only few true decision points. We thus conjecture that heuristic guidance is not worth the cost of computing a heuristic, explaining why LAMA−FF with its poor guidance but very fast computation (much faster than even h^{FF}) performs best.

Agile

As in the satisficing track, we compare DALAI against LAMA(-first),[4] including a version of DALAI with h^{FF} and a version of LAMA without h^{FF}. Even though LAMA was used only as a baseline, the old master's wisdom stood undefeated in first place, while DALAI achieved place 10 out of 23. Still, DALAI managed to surpass LAMA in three domains: `labyrinth`, `recharging-robots` and `ricochet-robots`. For the latter two, it is even the best performing planner overall.

Looking at Table 3, we see that both for LAMA and DALAI the configurations without h^{FF} perform better in those three domains. A deeper analysis reveals that in most cases, configurations with h^{FF} have significantly fewer expansions, but the higher computation cost of h^{FF} compared to h^{sum} can negate this advantage again.

Comparing DALAI and LAMA−FF, we see that they behave very similar with a slight edge for LAMA−FF. The similarity is explained by the fact that they have an almost identical configuration, but there are some small differences. First, DALAI represents landmarks as disjunctive action landmarks, which leads to some small changes on how many landmarks exist for a task. Secondly, LAMA only computes preferred operators from the h^{FF} heuristic and thus LAMA−FF does not have them, while we also compute preferred operator for h^{sum} and use them without boosting. Finally, the changed progression function can also influence the search behaviour. These differences lead to DALAI needing fewer expansions but requiring more time for each expansion in general.

[4]For the remainder of this section we use LAMA to denote LAMA-first.

	DALAI	LAMA−FF	DALAI+FF	LAMA
`folding`	7	8	2	**10**
`labyrinth`	**4**	**4**	0	0
`quantum-layout`	18	17	18	**19**
`recharging-robots`	**12**	**12**	8	10
`ricochet-robots`	8	**10**	9	5
`rubiks-cube`	3	3	**20**	**20**
`slitherlink`	0	0	0	0
Total (140)	52	54	57	**64**

Table 3: Agile coverage of DALAI and LAMA.

Apology

Please excuse inaccuracies due to entertainment purposes.

Acknowlodgy

We thank all Fast Downward contributors, and in particular we thank the developers of LAMA for inspiration.

Bibliology

Bonet, B.; and Helmert, M. 2010. Strengthening Landmark Heuristics via Hitting Sets. In Coelho, H.; Studer, R.; and Wooldridge, M., eds., *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, 329–334. IOS Press.

Büchner, C.; Eriksson, S.; Keller, T.; and Helmert, M. 2023. Landmark Progression in Heuristic Search. In Koenig, S.; Stern, R.; and Vallati, M., eds., *Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling (ICAPS 2023)*, 70–79. AAAI Press.

Büchner, C.; Keller, T.; and Helmert, M. 2021. Exploiting Cyclic Dependencies in Landmark Heuristics. In Goldman, R. P.; Biundo, S.; and Katz, M., eds., *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*, 65–73. AAAI Press.

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.

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.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 162–169. AAAI Press.

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

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *Journal of Artificial Intelligence Research*, 22: 215–278.

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

Karpas, E.; and Domshlak, C. 2009. Cost-Optimal Planning with Landmarks. In Boutilier, C., ed., *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 1728–1733. AAAI Press.

Karpas, E.; and Domshlak, C. 2012. Optimal Search with Inadmissible Heuristics. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 92–100. AAAI Press.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and Complete Landmarks for And/Or Graphs. In Coelho, H.; Studer, R.; and Wooldridge, M., eds., *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, 335–340. IOS Press.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the Extraction, Ordering, and Usage of Landmarks in Planning. In Cesta, A.; and Borrajo, D., eds., *Proceedings of the Sixth European Conference on Planning (ECP 2001)*, 174–182. AAAI Press.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks Revisited. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, 975–982. AAAI Press.

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

Simon, S.; and Röger, G. 2015. Finding and Exploiting LTL Trajectory Constraints in Heuristic Search. In Lelis, L.; and Stern, R., eds., *Proceedings of the Eighth Annual Symposium on Combinatorial Search (SoCS 2015)*, 113–121. AAAI Press.

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