Comparing Optimal Search Plans of Landmark Count and Landmark Cut Heuristic using A* Search

George Bradley , Christodoulos Loukaides, Nikhil Suresh , Mohak Narang , Beenitaben Shah
(george.bradley@kcl.ac.uk), (christodoulos.loukaides@kcl.ac.uk),(nikhil.suresh@kcl.ac.uk),
(mohak.narang@kcl.ac.uk), (beenitaben.shah@kcl.ac.uk)
MSc Artificial Intelligence

Part I

Introduction

The heuristics Landmark Count (LM Count) and Landmark Cut (LM Cut) both use landmark identification but are not directly comparable due to their differences in admissibility - the latter is admissible, the former is not. Although admissible heuristics are less informative, their admissibility guarantees an optimal plan when used with an optimal search strategy whereas inadmissible heuristics do not. Since the two heuristics are incomparable, A* search was use to compare search time variations for optimal solutions. The analysis considered findings from domains and problems where both heuristics returned an optimal solution, ensuring valid comparisons between search times. This report analyses the impact of admissibility on heuristic performance in terms of search times and plan cost.

Background

Landmarks can be used as a basis for guided search as an alternative for domain independent heuristics, providing information about important sub-goals that must occur in any valid plan. It is critical to decide the correct ordering of extracted landmarks as these orderings construct a partial plan that does not need to be backtracked (Porteous, Sebastia, and Hoffmann 2001).

There are three different types of landmarks (Helmert and Domshlak 2010):

- Fact Landmarks: a variable is assigned a certain value in at least one state.
- Action Landmarks: an action must exist in the solution.
- Disjunctive Action Landmarks: a set of actions from which at least one must occur in the final plan.

LM Count and LM Cut depend on landmarks for their operations. LM Cut utilises disjunctive action landmarks exclusively (Helmert and Domshlak 2010).

LM-Count The LM Count heuristic propagates landmark information, assuming that each landmark is achieved by a distinct action and that all landmarks must appear in any valid plan (Richter, Helmert, and Westphal 2008). Therefore, the number of actions remaining is approximately equivalent to the number of landmarks that still need to be achieved (Zhu and Givan 2003). The heuristic calculates ”Accepted” and ”Unachieved” landmarks. The ”Unachieved” landmarks are defined as the sum of currently ”Unaccepted” and ”Required again” landmarks. The heuristic “H”-value is then calculated as ”Accepted” - ”Unachieved”; an estimation of distance to the goal (Richter, Helmert, and Westphal 2008).

- A landmark i, is accepted if it is true in the current state and all landmarks before i are accepted in the predecessor state from which the current state was generated.
- A required-again landmark is a landmark which is false in the current state and is a greedy-necessary predecessor of some landmark which is not accepted.

The method of ”H”-value calculation is based on the aforementioned distinct action assumption and so sometimes results in an overestimation, why LM Count is inadmissible and consequently inconsistent. The following two causes could also lead to overestimation (Zhao, Liu, and Yang 2011):

- Approximation of the total number of landmarks is done through incomplete practical methods. Therefore, the number of landmarks is not accurate.
- Accepted landmarks are computed using inaccurate methods which only consider greedy-necessary orderings.

LM Count is a path-dependent heuristic, meaning the ”H”-value is dependent on the path taken to get to a state. Accepted landmarks are not a function of the state but of the path taken, hence one can not deduce anything about ”Accepted” landmarks by simply looking at the current state (Richter and Westphal 2008).

LM-Cut LM Cut combines landmark identification and cost partitioning approaches on relaxed planning graphs (RPGs) to make optimal heuristic estimates(Helmert and Domshlak 2010). It takes an iterative approach to find disjunctive action landmarks by identifying simple cuts in the justification graph, created using a precondition choice function (PCF) (Lauer and Fickert 2020). It computes PCFs and makes cuts in the graph in a ’goal-directed’ way.

To determine the ”H”-value for a state “s”, first the maximum cost of reaching all the other states from it in the RPG is computed. The PCF function used here is defined such

that it maps each action to exactly one precondition; the one with maximum cost(Štolba, Fišer, and Komenda 2015). This creates a directed graph of preceding-action mappings with facts (preconditions) as nodes and actions as edges. This justification graph, illustrates the transformation of action preconditions into the action add effects(Lauer and Fickert 2020). The graph is segmented into three sets (Bonet and Helmert 2010):

- Zero Goal Zone (V*): set which contains states from which the goal state is reachable on the application of a zero-cost action.
- Before Goal Zone (V^0): set which contains states that can be reached from the state 's' without passing through the zero goal zone states.
- Other Vertices (V^b): set which contains all the states that are neither part of the zero goal zone nor the before goal zone.

A cut is made in the justification graph to separate the nodes in V^b zone from the nodes in V^0 zone, done by determining the edges (actions) that connect the nodes of these two zones together.

The actions identified are added to the action-landmark set. Landmarks (disjunctive action landmarks) are a collection of edges that connect nodes in V* to nodes in the V^b zone. The costs of all actions in the action-landmark set are compared, and the minimum action cost C_m is determined. The C_m is added to the heuristic value and is subtracted from all actions belonging to the action landmark set to get the new action set cost (Bonet and Helmert 2010). That is:

$$ac' = ac - C_m, "H" += C_m$$

LM Cut makes use of both h^{max} for selecting action precondition and minimum action cost (C_m) for updating both the landmark set cost and "H"-value.

This process is iterated until state "s" is the only state left in the V^b zone. The heuristic value is updated throughout the iterative process and is returned in the end. The justification graph and cuts are discarded after returning the heuristic value (Pommerening and Helmert 2012). The "H"-value calculation of one state is not reused for "H"-value calculations of adjacent states, thereby increasing the number of computations. LM Cut can still be computed in polynomial time and gives close estimates to the optimal RPG (h^+) which is NP-hard to compute (Lauer and Fickert 2020).

Experiment Design

Following initial research into the two heuristics, these hypotheses were proposed:

- LM Count would find non-optimal solutions to some problems due to its inadmissibility. LM Cut should always come up with an optimal solution as an admissible heuristic.
- LM Count is expected to use more memory as it incrementally builds up its plan, having to re-evaluate landmarks that may be required again in future states, without discarding anything along the way. LM Cut discards its computed landmarks and justification graph before calculating the heuristic value for each node, thus should not require as much memory as LM Count.
- LM Cut may require more processing time in domains where there are few disjunctive action landmarks as its efficient performance depends on utilising these. The lack of which prevents the heuristic's architecture from using previously calculated values to calculate heuristic values of adjacent nodes, leading to repeated calculations which will result in an increase in search time.
- LM Cut depends on the discovery and utilisation of disjunctive action landmarks leading to the hypothesis that it will perform better in domains where a linear execution of actions (or set of actions) is possible (e.g. Trucks domain). Domains where most of the actions (or set of actions) have a similar distance to the goal (e.g. Gripper domain) could be an issue for LM Cut as the policy of discovering disjunctive action landmarks would be of little to no use.

Experiments were designed such that the above hypotheses were tested by comparing the search times when using either heuristic. Search times were the primary comparison feature, however plan length and peak memory usage were considered. Following this reasoning, a single complete and optimal search strategy was used: A* (along with the lm_exhaust landmark factory when using LM Count). This decision was based based on two reasons. Firstly, comparisons consisted of optimal plans only, hence, using an optimal search strategy was essential. Secondly, this single search strategy allowed for sufficient testing and analysis of the aforementioned hypotheses. Search time results were analysed using the Wilcoxon Ranked Sign so as to check the significance of the results. The null hypothesis assumed: there is no significant difference in the differences seen in search times between the A* search run using LM Count vs LM Cut. This test was chosen as it is a non-parametric statistical test. A confidence level of 5% is used, hence a p-value lower than 0.05 meant the null hypothesis was rejected. The domains selected for experimentation were (1) Trucks, (2) Sokoban, (3) Satellite, (4) Gripper, (5) Pipes-Tankage, (6) Pipes-No Tankage, and (7) Airport. Initial results indicate that certain domains favoured one heuristic over the other, a result which was investigated further via the calculated selection of domains. A representative set of domains were selected such that they would highlight key strengths and weaknesses of each heuristic. Search times were compared in order to determine if a heuristic does consistently outperform the other in certain domains.

Methodology The methodology for domain evaluation was as follows:

1. Run multiple problems for each domain, covering a broad spectrum of difficulty. Problems increased in complexity and difficulty until the planner hit a memory or time limit error.
2. Observe heuristic performance in each domain, noting features such as memory usage and solution optimality.

A memory limit of 4GB and a time limit of 30 minutes were set for the experiments. Constrain decisions were made based on a trade-off between time taken to reach a solution and problem complexity.

As testing of both heuristics, on a per domain basis, was carried out on a single machine, it allowed for valid comparisons as comparisons were intra-domain.

Part II

Results and Analysis

Heuristics were compared in terms of search time with respect to the optimal solutions. Results were only considered where LM Count and LM Cut both returned the same, hence, optimal plan. Comments on cases where both heuristics do not return the same plan are included near the end of the section.

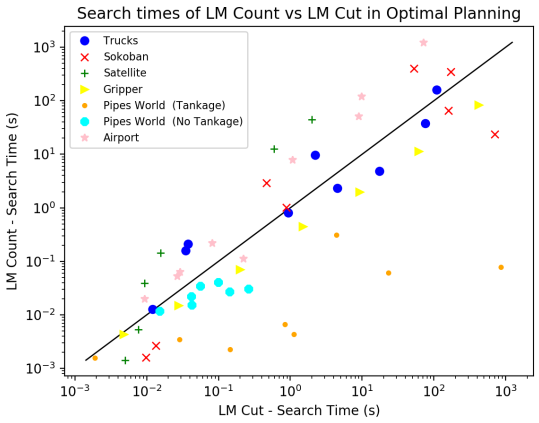

Figure 1: Graph showing comparisons for search time between searches using LM Count vs LM Cut heuristics for optimal plans only.

Domain	Prob No	% of P	Avg Decrease	P Val
Trucks	9	56	11.89s	0.04
Sokoban	8	50	113.30s	0.07
Satellite	6	67	13.57s	0.07
Airport	9	89	165s	0.01

Table 1: This table shows the percentage of problems, in each domain, where LM Cut is faster and by how much (on average) in seconds. The final column shows the p-values resulting from a Wilcoxon Ranked Sign test.

Figure 1 shows how problem search times compare across domains for both heuristics. LM Cut reaches optimal plans faster in the following domains: Trucks, Satellite and Airport as visible in the Table 1. In these domains, LM Cut achieves a shorter search time in more than 50% of the problems. The p-values in Table 1 convey that only the results from the Trucks and Airport domains are significant, under a 5% confidence threshold, when evaluated using a Wilcoxon Ranked Signed test, indicated that LM Cut significantly performs better in these domains.Sokoban and Satellite do not favour either heuristic.

An explanation of LM Count's slower search times in these domains is due to LM Count tackling each landmark in isolation. Where landmarks impede another, sub-goals may be satisfied, however overall planning is computationally expensive as it may lead to landmarks needing to be backtracked and achieved again when tackled in the wrong order. Domains where LM Cut is faster can be described as having sub-goals that interact with each other. From the hypothesis, LM Cut's discovery and utilisation of disjunctive action landmarks lead to faster search times in domains where actions can be executed linearly. Considering the Trucks domain, if the truck loads a package at a location where there are multiple packages, LM Count finds the next best action which is driving to where the package needs to be dropped as it has a lower "H"-value because it achieves another landmark. However, picking up another package may prove to be better for an optimal plan as it means the truck does not have to return to the location to pick up the other packages. The act of loading a package before driving away leads to an optimal plan.

Tackling landmarks in isolation means that it takes longer to realise the optimal plan. LM Count evaluates more states in order to reach the optimal plan as the "H"-value generated by it proves to be misleading, resulting in the "wrong" decision being made at critical points in the plan. This keeps it from finding the optimal plan for some problems. LM Count may recalculate "required again" landmarks for states it has already seen multiple times in its "greedy" exploration of the search space. Comparatively, LM Cut's utilisation of disjunctive landmarks and re-computation of all relevant information leads to a "H"-value which is now worth more than the time taken to re-compute information.

We infer from Table 1 that, unlike LM Count, there is no domain where LM Cut proves to be faster for all the problems throughout the domain. However, in domains where LM Cut's decrease in search time is significant, i.e. Trucks and Airport domain, the difference in search times for problems where LM Count has a faster search time proves to be insignificant. It can be concluded that in these domains, LM Cut strictly outperforms LM Count.

Domain	Prob No	% of P	Avg Decrease	P Val
Trucks	9	44	13.06s	0.07
Sokoban	8	50	192.89s	0.07
Satellite	6	33	0.003s	0.18
Gripper)	7	100	211.0s5	0.01
Pipes(T)	8	100	111.22s	0.01
Airport	9	11	0.11s	0.32
Pipes(No T)	7	100	0.07s	0.02

Table 2: This table shows the percentage of problems, in each domain, for which LM Count is faster and by how much (on average) in seconds. The final column shows the p-values resulting from a Wilcoxon Ranked Sign test.

LM Count performs better than LM Cut on the Gripper, Pipesworld-Tankage, and Pipesworld-No Tankage do-

mains. On these domains, LM Count computes faster across all problems with a 5% confidence threshold level, shown in Table 2. In problems of increasing difficulty, LM Count compiles a plan significantly faster than LM Cut whilst generating, expanding and evaluating more states. LM Count computes landmarks once for the entire relaxed plan and then computes heuristics for states based on this initial computation. It is an estimate of the heuristic value and not necessarily an underestimate of the goal distance. Consequently, LM Count is able to consistently achieve a shorter search time than LM Cut in these domains seen in Figure 1. In these cases, LM Cut's continuous re-computations are time-consuming without providing an advantage over LM Count's landmark counting strategy. LM Cut's informativeness does not provide a sufficient pay-off in comparison to the time taken to calculate a solution. LM Count computes values only once and uses them to expand and evaluate states to search more of the search space faster, allowing for the optimal plan to be found quicker when applied to these problems.

In the Pipesworld-Tankage domain, as the problem size increases, the difference in search times between LM Count and LM Cut increases exponentially. Therefore is can be concluded that as the problem difficulty grows, not only does LM Cut's search time increase, but it increases at a faster rate than LM Count's, supporting the claim that in domains where there are fewer disjunctive action landmarks, re-computations during search prove to be time-consuming without yielding any real advantage.

LM Count's inadmissibility can be observed by the fact that certain plans found with LM Cut are of shorter makespan than the ones found by LM Count but never the other way around, as LM Cut is admissible, as evidenced in Table 3.

Domain	Prob No	LM Count	LM Cut
Pipes(T)	1	8	5
Pipes(T)	6	12	10
Airport	18	113	107
Airport	19	92	90

Table 3: This table shows a non-exhaustive list of plan lengths in selected; namely problems and domains where LM Count produces a longer plan than LM Cut.

During analysis, it was observed that there are times where LM Cut reaches a timeout where LM Count does not, namely in the Gripper domain and problems where LM Count achieves lower search times. This can be explained by the LM Cut's iterative calculations of costs for the current state, after which values are discarded and repeated for the next state which is computationally expensive. As mentioned in the hypothesis, this is a drawback when considering adjacent states as the planner has to repeat the entire computation again.

Conversely, LM Count runs out of memory where LM Cut does not as evidenced by due to the greater amount of memory it uses, as evidenced by Table 4. This observation relates back to the initial hypothesis, which predicts this occurrence due to LM Count considering landmarks independently, leading to more states being generated and evaluated, and hence more memory being used before the optimal plan is achieved.

Domain	Avg % of more memory used (LM Count)
Satellite	+96.01
Trucks	+93.91
Pipes (T)	-5.90
Sokoban	+875.60
Airport	+484.30
Pipes (No T)	+0.45

Table 4: This table shows the peak memory used by LM Count across the domains as a percentage of LM Cut's peak memory usage. These percentages are an average of the percentage difference for each problem in the respective domain.

Conclusion

Referring back to the hypotheses stated in the Experiment Design Section, the following conclusions were drawn:

- LM Count would return a non-optimal solution for some of the tested problems.

This holds, as evidenced by results in Table 3 which show that LM Cut always returns a solution that is either shorter than or the same length as LM Count. As mentioned previously, this is due to LM Count's inadmissibility, meaning that the makespan of its plans can be longer as a result of overestimation.

- LM Count generally uses more memory than LM Cut, as LM Cut resets memory after calculating "H"-value for each state.

Based on results presented in Table 4, this hypothesis holds. As problem complexity increases, we see a jump in the amount of memory required by LM Count which can be attributed to more states being generated, evaluated and expanded.

- LM Cut may require more processing time for domains with fewer disjunctive action landmarks and less where a linear execution of actions (or set of actions) is possible.

Both of these hold, as evidenced in Figure 1 and the analysis stated in the previous section. Firstly, as expected, in domains where the existence of disjunctive action landmarks are scarce, running search with LM Cut is seen to perform significantly worse than LM Count. While most actions (or set of actions) have similar distances to the goal (e.g. Gripper domain), discovering disjunctive action landmarks prove to be of little use. Not only can it not utilise its main principles for efficiently finding a solution, but due to not storing previously calculated heuristic values, LM Cut had to repeat a lot of calculations when falling back from unsuccessful path discoveries. On the other hand, where the linear execution of actions is a feature of the optimal plan, LM Cut is able to leverage its consideration of disjunctive landmarks, recognising that actions in a disjunctive action landmark can be done sequentially giving it a significant advantage.

References

Bonet, B., and Helmert, M. 2010. Strengthening landmark heuristics via hitting sets. In *ECAI*, volume 215, 329–334.

Helmert, M., and Domshlak, C. 2010. Landmarks, critical paths and abstractions: what's the difference anyway? In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Lauer, P., and Fickert, M. 2020. Beating lm-cut with lm-cut: Quick cutting and practical tie breaking for the precondition choice function. *HSDIP 2020* 9.

Pommerening, F., and Helmert, M. 2012. Optimal planning for delete-free tasks with incremental lm-cut.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the extraction, ordering, and usage of landmarks in planning. In *6th European Conference on Planning*.

Richter, S., and Westphal, M. 2008. The lama planner using landmark counting in heuristic search. In *Proceedings of IPC*, volume 14. Citeseer.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *AAAI*, volume 8, 975–982.

Štolba, M.; Fišer, D.; and Komenda, A. 2015. Admissible landmark heuristic for multi-agent planning. In *Twenty-Fifth International Conference on Automated Planning and Scheduling*.

Zhao, J.; Liu, D.; and Yang, Y. 2011. Enhancing planning heuristic with landmarks. *JOURNAL OF COMPUTERS* 6(12).

Zhu, L., and Givan, R. 2003. Landmark extraction via planning graph propagation. *ICAPS Doctoral Consortium* 156–160.