Towards Rational Deployment of Multiple Heuristics in A* (Extended Abstract)

David Tolpin, Tal Beja
Solomon Eyal Shimony
CS Department
Ben-Gurion University
Israel
{tolpin,bejat,shimony}@cs.bgu.ac.il

Ariel Felner
ISE Department
Ben-Gurion University
Israel
felner@bgu.ac.il

Erez Karpas
IEM Department
Technion
Israel
karpase@gmail.com

Abstract

In this paper we discuss and experiment with *Lazy* A^*, a variant of A^* where heuristics are evaluated lazily and with *rational lazy* A^*, which decides whether to compute the more expensive heuristics at all, based on a myopic value of information estimate. **Full version appears in IJCAI-2013 (Tolpin *et al.* 2013)**

Lazy A*

This paper examines the case where we have several available admissible heuristics. Clearly, we can evaluate all these heuristics, and use their *maximum* as an admissible heuristic, a scheme we call A^*_{MAX}. The problem with naive maximization is that all the heuristics are computed for all the generated nodes. In order to reduce the time spent on heuristic computations, Lazy A^* (or LA^*, for short) evaluates the heuristics one at a time, lazily. When a node n is generated, LA^* only computes one heuristic, $h_1(n)$, and adds n to OPEN. Only when n re-emerges as the top of OPEN is another heuristic, $h_2(n)$, evaluated; if this results in an increased heuristic estimate, n is re-inserted into OPEN. This idea was briefly mentioned by Zhang and Bacchus (2012) in the context of the MAXSAT heuristic for planning domains. LA^* is as informative as A^*_{MAX}, but can significantly reduce search time, as we will not need to compute h_2 for many nodes. In this paper we provide a deeper examination of $LA^*$and describe several technical optmizations for LA^*.

The pseudo-code for LA^* is shown in Algorithm 1. In fact, without lines 7 – 10, LA^* would be identical to A^* using the h_1 heuristic. When a node n is generated we only compute $h_1(n)$ and n is added to OPEN (Lines 11 – 13), without computing $h_2(n)$ yet. When n is first removed from OPEN (Lines 7 – 10), we compute $h_2(n)$ and reinsert it into OPEN, this time with $f_{max}(n)$.

It is easy to see that LA^* is as informative as A^*_{MAX}, as they both generate and expand and the same set of nodes (up to differences caused by tie-breaking). The reason is that a node n is expanded by both A^*_{MAX} and by LA^* when $f_{max}(n)$ is the best f-value in OPEN.

In its general form A^* generates many nodes that it does not expand. These nodes, called *surplus* nodes (Felner *et al.* 2012), are in OPEN when we expand the goal node with $f = C^*$. LA^* avoids h_2 computations for many of these surplus nodes. By contrast, A^*_{MAX} computes both h_1 and h_2 for all generated nodes. Thus, LA* can potentially run faster than A^*_{MAX}in many cases.

Algorithm 1: Lazy A^*

Input: LAZY-A^*
1 Apply all heuristics to Start
2 Insert Start into OPEN
3 **while** OPEN *not empty* **do**
4     $n \leftarrow$ best node from OPEN
5   **if** *Goal(n)* **then**
6   **return** trace(n)
7   **if** h_2 *was not applied to* n **then**
8   Apply h_2 to n
9   insert n into OPEN
10   continue   //next node in OPEN
11   **foreach** *child* c *of* n **do**
12   Apply h_1 to c.
13   insert c into OPEN
14   Insert n into CLOSED
15 **return** FAILURE

Rational lazy A*

LA^* offers us a very strong guarantee, of expanding the same set of nodes as A^*_{MAX}. However, often we would prefer to expand more states, if it means reducing search time. We now present *Rational Lazy A** (RLA^*), an algorithm which attempts to optimally manage this tradeoff.

Using principles of rational meta-reasoning (Russell and Wefald 1991), theoretically every algorithm action (heuristic function evaluation, node expansion, open list operation) should be treated as an action in a sequential decision-making meta-level problem: actions should be chosen so as to achieve the minimal expected search time. However, the appropriate general meta-reasoning problem is extremely hard to define precisely and to solve optimally.

Therefore, we focus on just one decision type, made by LA^*, when n re-emerges from OPEN (Line 7). We have two

Copyright © 2013, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

	Problems Solved						Planning Time (seconds)						GOOD	
Domain	h_{LA}	lmcut	max	selmax	LA^*	RLA^*	h_{LA}	lmcut	max	selmax	LA^*	RLA^*	LA^*	RLA^*
miconic	**141**	140	140	**141**	**141**	**141**	**0.13**	0.55	0.58	0.57	0.16	0.16	0.87	0.88
sokoban-opt08	23	25	25	24	26	**27**	3.94	1.76	2.19	2.96	1.9	**1.32**	0.04	0.4
OVERALL	698	697	722	747	747	**750**	1.18	0.98	0.98	0.89	0.79	**0.77**	0.27	0.34

Table 1: Planning Domains — Number of Problems Solved, Total Planning Time, and Fraction of Good Nodes

options: **(1)** Evaluate the second heuristic $h_2(n)$ and add the node back to OPEN (Lines 7-10) like LA^*, or **(2)** bypass the computation of $h_2(n)$ and expand n right way (Lines 11 - 13), thereby saving time by not computing h_2, at the risk of additional expansions and evaluations of h_1.

The only addition of RLA^* to LA^* is the option to bypass h_2 computations (Lines 7-10). Suppose that we choose to compute h_2 — this results in one of the following outcomes:
1: n is still expanded, either now or eventually.
2: n is re-inserted into OPEN, and the goal is found without ever expanding n.

Computing h_2 is *helpful* only in outcome 2, where potential time savings are due to pruning a search subtree at the expense of $t_2(n)$. Since we do not know this in advance, we calculate and use p_h - the probability that h_2 is *helpful*.

In order to choose rationally, we define a criterion based on value of information (VOI) of evaluating $h_2(n)$ in this context. The following notations are used. $b(n)$ is the branching factor at node n, t_d is the to time compute h_2 and re-insert n into OPEN thus delaying the expansion of n, t_e is the time to remove n from OPENand p_h the probability that h_2 is *helpful*.

As we wish to minimize the expected regret, we should thus evaluate h_2 just when $(1 - b(n)p_h)t_d < p_h t_e$ and bypass this computation otherwise. The complete derivation appears in our full paper (Tolpin *et al.* 2013).

Experimental results

We experimented with LA* and RLA* on a number of domains but focus here on planning domains where we experimented with two state of the art heuristics: the admissible landmarks heuristic h_{LA} (used as h_1) (Karpas and Domshlak 2009), and the landmark cut heuristic h_{LMCUT} (Helmert and Domshlak 2009) (used as h_2). We experimented with all planning domains without conditional effects and derived predicates (which the heuristics we used do not support) from previous IPCs.

Table 1 depicts the experimental results (for two of our domains and the overall over all domains) for LA^* and RLA^* to that of A^* using each of the heuristics individually, as well as to their max-based combination, and their combination using selective max (Selmax) (Domshlak *et al.* 2012). Selmax is an online learning scheme which chooses one heuristic to compute at each state. The leftmost part of the table shows the number of solved problems in each domain. As the table demonstrates, RLA^* solves the most problems, and LA^* solves the same number of problems as selective max. Thus, both LA^* and RLA^* are state-of-the-art in cost-optimal planning.

The middle part of the Table 1 shows the geometric mean of planning time in each domain, over the commonly solved problems (i.e., those that were solved by all 6 methods). RLA^* is the fastest overall, with LA^* second. Of particular interest is the *miconic* domain. Here, h_{LA} is very informative and thus the variant that only computed h_{LA} is the best choice (but a bad choice overall). Observe that both LA^* and RLA^* saved 86% of h_{LMCUT} computations, and were very close to the best algorithm in this extreme case. This demonstrates their robustness.

The rightmost part of Table 1 shows the average fraction of nodes for which LA^* and RLA^* did not evaluate the more expensive heuristic, h_{LMCUT}, over the problems solved by both these methods. This is shown in the *good* columns. We can see that in domains where there is a difference in this number between LA^* and RLA^*, RLA^* usually performs better in terms of time. This indicates that when RLA^* decides to skip the computation of the expensive heuristic, it is usually the right decision.

Finally, Table 2 shows the total number of expanded and generated states over all commonly solved problems. LA^* is indeed as informative as A^*_{MAX} (the small difference is caused by tie-breaking), while RLA^* is a little less informed and expands slightly more nodes. However, RLA^* is much more informative than its "intelligent" competitor - selective max, as these are the only two algorithms in our set which selectively omit some heuristic computations. RLA^* generated almost half of the nodes compared to selective max, suggesting that its decisions are better.

	Expanded	Generated
h_{LA}	183,320,267	1,184,443,684
lmcut	23,797,219	114,315,382
A^*_{MAX}	22,774,804	108,132,460
selmax	54,557,689	193,980,693
LA^*	22,790,804	108,201,244
RLA^*	25,742,262	110,935,698

Table 2: Total Number of Expanded and Generated States

References

Carmel Domshlak, Erez Karpas, and Shaul Markovitch. Online speedup learning for optimal planning. *JAIR*, 44:709–755, 2012.

A. Felner, M. Goldenberg, G. Sharon, R. Stern, T. Beja, N. R. Sturtevant, J. Schaeffer, and Holte R. Partial-expansion A* with selective node generation. In *AAAI*, pages 471–477, 2012.

Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In *ICAPS*, pages 162–169, 2009.

Erez Karpas and Carmel Domshlak. Cost-optimal planning with landmarks. In *IJCAI*, pages 1728–1733, 2009.

Stuart Russell and Eric Wefald. Principles of metereasoning. *Artificial Intelligence*, 49:361–395, 1991.

D. Tolpin, Tal Beja, S. E. Shimony, A. Felner, and E. Karpas. Towards rational deployment of multiple heuristics in A*. In *IJCAI*, 2013.

Lei Zhang and Fahiem Bacchus. Maxsat heuristics for cost optimal planning. In *AAAI*, 2012.