Variants of A* for Planning

Minh Tang & Amol Dattatraya Mali[1]

Abstract. Many of the recently developed efficient planners use a variation of A* search algorithm called weighted A*. We report on three sound and complete forward state-space STRIPS planners AWA*-PD, AWA*-AC and AWA*-AC-LE. AWA*-AC is adjusted weighted A* with action conflict-based adjustment. AWA*-PD is adjusted weighted A* with deleted preconditions-based adjustment. AWA*-AC-LE is a variant of AWA*-AC which performs lazy evaluation (LE). The novel ideas in these planners are (i) node-dependent weighting for g(n) and h(n) terms in the path cost equation of A*, (ii) conditional two-phase heuristic evaluation, and (iii) lazy heuristic evaluation which does not construct relaxed plans to compute heuristic values for all nodes. We report on an empirical comparison of the planners with planners HSP-2 and FF on new challenging domains containing dead ends. Our evaluation shows that heuristic search planning is significantly benefitted by node-dependent weighting, conditional two-phase heuristic evaluation and lazy evaluation.

Significant advances have occurred in heuristic search planning in the last seven years. These are clear from the success of planners FF [3] and HSP-2 [1]. FF carries out local search or weighted A* (WA*) search, depending on the version chosen. HSP-2 carries out WA* style search. WA* style search has been also used by planner Sapa [2]. HSP-2, Sapa and FF keep the weight in WA* fixed throughout the search. Using different weights for different nodes can provide better search control without loss of completeness and without significant loss of optimality.

Current planners use the same heuristic/heuristics to evaluate all nodes in the fringe. Using the same heuristic to evaluate all nodes in the fringe can be highly disadvantageous. If the heuristic values cannot be computed fast, solving time increases, with more nodes in the fringe. If the heuristic is not very informative, evaluating all nodes in the fringe with the heuristic may not be useful. More informative heuristics are generally computationally demanding. So one can use a computationally cheap heuristic for all nodes in the fringe (in phase 1) and use a more informative heuristic to re-evaluate a small number of the nodes in the fringe (in phase 2), in order to perform a fast and more informative heuristic evaluation. Our conditional two-phase heuristic evaluation is based on this idea.

Using the same computational process to evaluate all nodes can slow down a planner. Plan synthesis may become efficient if some nodes are evaluated with a computationally very cheap process. This can be done even when the heuristic evaluation is uni-phase. Hence this differs from conditional two-phase heuristic evaluation. Specifically, some nodes can be evaluated only with heuristic H_1 and the remaining nodes can be evaluated only with heuristic H_2 which can be used much faster than H_1. One approach is to use the information about already-evaluated siblings of a node n or randomization, to decide whether to evaluate n with H_1. Such a lazy evaluation can speed up planning since it reduces the time spent on nodes' evaluation. AWA*-AC-LE uses this kind of evaluation.

We report on three sound and complete classical planners AWA*-AC, AWA*-PD and AWA*-AC-LE in this paper. The conditions for their completeness are same as the conditions for the completeness of A* (finite branching factor and positive step costs). These planners use variations of the WA* search algorithm. All these planners use node-dependent weights for the $g(n)$ and $h(n)$ terms in the path cost equation. They also use conditional two-phase heuristic evaluation. AWA*-AC and AWA*-PD differ from each other due to different information used in the second phase of conditional two-phase heuristic evaluation. AWA*-AC-LE is obtained by integrating lazy evaluation with AWA*-AC. AWA*-AC and AWA*-PD find relaxed plans for all nodes in the fringe, to compute the heuristic value. AWA*-AC-LE decides whether to compute relaxed plan for a node based on the heuristic values of its siblings and a randomly generated number. AWA*-AC-LE does not find relaxed plans for all nodes in the fringe, thus performing a lazy heuristic evaluation. The variants of A* outperform FF and HSP-2 on several problems from domains TSP-2n and TSP-3. These domains contain dead ends.

Conditional two-phase heuristic evaluation: This differs from the combination of heuristics in the pattern database approach [4]. The nodes to be evaluated in the second phase are selected based on the evaluation from phase 1. Hence the evaluation in second phase is "conditional". The second phase may detect less promising nodes that phase 1 did not. This idea can be extended to conditional multi-phase heuristic evaluation.

Considering information ignored in the computation of relaxed plans: Information ignored in the construction of relaxed plans can be used in the future phases of multi-phase heuristic evaluation. AWA*-AC, AWA*-PD and AWA*-AC-LE use some information ignored in phase 1 evaluation in phase 2. These three planners ignore delete effects of actions in the synthesis of relaxed plans in phase 1 and consider the delete effects in phase 2. This can help in efficient plan synthesis, as our results show.

The variants of A* use the following path cost equation: $f(n) = \phi(n) * g(n) + \psi(n) * h(n)$, where $\phi(n)$ and $\psi(n)$ are functions of the current state (world state in node n). In the implementations of AWA*-AC, AWA*-PD, and AWA*-AC-LE, we chose the functions $\phi(n)$ and $\psi(n)$ as follows: $\phi(n) = \mu_g$, and $\psi(n) = \mu_h - \frac{s(n)}{\delta_h * |G|}$, where $\mu_g, \mu_h,$ and δ_h are positive user-specified constants, $s(n)$ is the number of subgoals from G true in state in node n, and $\mid G \mid$ is the number of subgoals (propositions or predicates) in the goal of the planning problem. We require $\mu_h > \frac{1}{\delta_h}$ so that $\psi(n)$ is always positive.

Algorithm of AWA*-AC: c is a user-specified positive constant. PAD-mutex actions are those whose mutual exclusivity can be inferred simply by comparing preconditions, add effects and delete effects.

[1] EE & CS, University of Wisconsin, Milwaukee, WI 53211, USA, minhtang@uwm.edu, mali@miller.cs.uwm.edu

1. Evaluate all nodes in the fringe whose h value was not found, by computing $h(n)$, where $h(n)$ = number of actions in the relaxed plan at n.
2. For all nodes in the fringe { If $h(n) < c$, and n was not evaluated in second phase before, then $h(n)$ = $h(n)$ + (Total number of pairs of concurrent PAD-mutex actions in the relaxed plan at n) }.
3. For all nodes in the fringe whose f value has not been found, compute
$f(n) = \mu_g * g(n) + (\mu_h - \frac{s(n)}{\delta_h * |G|}) * h(n)$
4. Expand node in the fringe with lowest $f()$ value.
5. Go to 1 if no plan is found.
6. Return plan.

Algorithm of AWA*-PD: This variant of AWA*-AC was created to evaluate another adjustment strategy in phase 2 of conditional two-phase heuristic evaluation. AWA*-PD differs from AWA*-AC only in second step. The adjustment in second phase is the total number of preconditions of actions in the relaxed plan at n that are deleted by some preceding action in the relaxed plan.

Algorithm of AWA*-AC-LE: AWA*-AC-LE decides whether to compute relaxed plan for a node based on the number of its interesting and less interesting siblings and a randomly generated number. AWA*-AC-LE is motivated by the intuition that constructing relaxed plans for fewer nodes may speed up plan synthesis significantly.

$parent(n)$ is parent of node n. θ_1 is the limit for number of interesting siblings. θ_2 is the limit for the number of less interesting siblings. n_i is i th child of node n. A node n is interesting if $h(n) < h(parent(n))$. A node n is less interesting if $h(n) > h(parent(n))$. α denotes the user-specified positive constant used to compute the h value of a node fast. $IS(n)$ is the number of interesting siblings of node n. $LIS(n)$ is the number of less interesting siblings of node n. p is a user-specified positive integer used in deciding whether to compute relaxed plan for a node. θ_1 and θ_2 are user-specified.

1. Expand root node. Find the h value for root node using relaxed plan heuristic.
2. Let n be the most recently expanded node. Let nc be the number of children of n. If $nc \leq min(\theta_1, \theta_2)$, then evaluate all children using relaxed plan heuristic.
3. If $nc > min(\theta_1, \theta_2)$, then evaluate first $min(\theta_1, \theta_2)$ children using the relaxed plan heuristic. (Note: i th child of n is the i th node generated while expansion of n.)
4. For $i = (min(\theta_1, \theta_2) + 1)$ to nc
{
If $((IS(n_i) < \theta_1) \wedge (LIS(n_i) < \theta_2))$
then $h(n_i)$ = Number of actions in the relaxed plan at n_i
Else
{ Generate a random positive integer x.
If (x modulo p) == 0), then
$h(n_i)$ = Number of actions in the relaxed plan at n_i
Else
$h(n_i) = h(n) + \alpha$
}
}
5. For ($i = 1$ to nc) { If (n_i was evaluated using relaxed plan heuristic) $\wedge$ ($h(n_i) < c$) $\wedge$ ($h(n_i)$ was not adjusted in second phase)), then
$h(n_i)$ = $h(n_i)$ + Total number of pairs of concurrent PAD-mutex actions in the relaxed plan at n. }
6. For all nc children of n, find
$f(n_i) = \mu_g * g(n_i) + (\mu_h - \frac{s(n_i)}{\delta_h * |G|}) * h(n_i)$
7. Expand node in the fringe with lowest $f()$ value.
8. Go to 2 if no plan is found.
9. Return plan.

In the empirical comparison, we used version 2.3 of FF (with default options) since domains TSP-2n and TSP-3 contain non-invertible actions. We used default options of HSP-2. TSP-2n is a traveling salesperson domain where the salesman must visit a set of cities exactly once, except the relaxed cities. TSP-3 is a domain where a salesman can visit a set of cities either by flying or driving. The salesman can drive to a city at the most once. He/She can fly to any city an unlimited number of times, if she/he has enough cash. Cash can be found only at a small number of cities. a and t in table 1 respectively denote the number of actions in the plans found and the solving time in cpu seconds. * denotes that the problem was not solved in 20 cpu minutes.

Prob	AWA*-AC		HSP-2		FF	
	a	t	a	t	a	t
tsp3-33	33	19.9	*	*	*	*
tsp3-34	*	*	*	*	35	16.1
tsp3-35	29	20.9	*	*	*	*
tsp3-36	32	33.8	31	12.7	*	*
tsp3-37	32	179.6	35	33.1	34	4.1
tsp3-40	29	10.2	*	*	*	*
tsp2n-1	27	24.6	29	19.4	28	2.1
tsp2n-2	28	319.9	*	*	*	*
tsp2n-4	27	35.9	*	*	*	*
tsp2n-5	30	149.6	29	333.7	*	*
tsp2n-16	32	184.4	34	500.4	32	2.2
tsp2n-19	32	175.9	32	33.4	*	*

Table 1. AWA*-AC, HSP-2 and FF on TSP-3 and TSP-2n

We compared AWA*-AC-LE and AWA*-AC on 30 problems from TSP-3 domain. AWA*-AC-LE solved 26 problems and AWA*-AC solved 22 problems. AWA*-AC-LE solved 23 problems faster than AWA*-AC. We did run the variants and planners AltAlt [6] and STAN 4 [5] on some problems from TSP domains. The variants were more efficient than these planners on these problems.

ACKNOWLEDGEMENTS
This work is funded by NSF grant IIS-0119630 to the second author.

REFERENCES

[1] B. Bonet and H. Geffner, Planning as Heuristic Search, AI Journal, Vol.129(1-2), June 2001, pp. 5-33.
[2] M. Do and S. Kambhampati, Sapa: A Domain-Independent Heuristic Metric Temporal Planner, ECP proceedings, 2001, pp. 109-120
[3] J. Hoffmann and B. Nebel, The FF Planning System: Fast Plan Generation through heuristic search, Journal of Artificial Intelligence Research, Vol. 14, 2001, pp. 253-302.
[4] Richard E. Korf and Ariel Felner, Disjoint pattern database heuristics, Artificial Intelligence 134, 2002, pp. 9-22
[5] Maria Fox and Derek Long, STAN 4: A hybrid planning strategy based on subproblem abstraction, AI Magazine 22(3), 2001, pp. 81-84
[6] XuanLong Nguyen, Subbarao Kambhampati and Romeo Sanchez Nigenda, Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search, Artificial Intelligence 135(1-2), 2002, pp. 73-123.