BDI-Learning Discussion Paper #4: A Complexity-based Confidence Measure

Dhirendra Singh
dhirendra.singh@rmit.edu.au

December 18, 2009

1 Background and Scope

This document is a corollary of the HYCAS submission [1] and should be read in conjunction with the same. The focus of this document is modifications to the complexity based confidence measure plus fixes for any open issues with the proposal.

2 Recent Changes

2.1 Complexity Calculation

For our purposes, when we talk about *complexity* of the goal-plan tree node we refer to the number of paths below that node in the structure. Where the notion of *coverage* [2] was an accurate representation of these paths, complexity is intended to be an approximation that is calculated based on certain properties of the goal-plan tree. In [1], however, this approximation is not computed but rather user specified for each plan in the example domain (Towers of Hanoi).

The proposed change for complexity calculations is as follows: For non-recursive nodes use $\eta = \rho^{\sigma}$. For recursive nodes use $\eta = log_2(\delta + 2)^{\rho^{\sigma}}$. Here η is the complexity, ρ is the maximum number of possible plan options per goal in factor for plans in the hierarchy, σ is the maximum branching factor for goals in the hierarchy, and δ is the recursive depth.

Dhirendra to self: This should instead be a single formula for 'leaf' and 'non-leaf' (recursive or otherwise) nodes instead.

2.2 Optimised Recording

In an attempt to reduce the number of saved experiences over time and improve decision tree induction times we performed two optimisations. Firstly, decision tree induction is performed only when at least k new experiences have been recorded. In [1] we used $k = 25$ that gives significant improvements in running times with negligible impact on overall performance. Secondly, we reduce the total number of samples used, by compacting all duplicate experiences into a single experience that uses a *weight* proportional to the number of duplicates. So for a given plan then, a maximum of two instances per world are possible, one for success and one for failure, each with different individual weights.

2.3 Confidence Calculation

Saved experiences are used to estimate decision tree classification confidence as follows:

In [1], if a plan has experienced at least one success in a given world then that plan temporarily gets full confidence for that world regardless of the number of failures (the reasoning being that if a plan has succeeded in a world then there is no need to consider previous failures in the decision making). A related limitation however, is that if a successful plan starts failing (due to environmental changes for instance) then this change will go unnoticed because we disregard all failures for a previously successful plan. The fix is to reset failure counts for a given plan in a given world each time a success is experienced, and start counting failures afresh. Confidence for such previously successful worlds is then the ratio $success/attempts$.

For worlds where the plan has never been successful, confidence in the decision tree classification is a gradually increasing value based on the decay function described in [1] (the decay is faster for small sub-trees and slower for more complex ones).

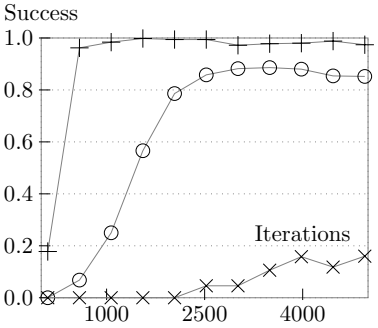
One important issue with the decay-based confidence calculation is that it does not consider the point of failure. So, for instance, a very complex plan structure that fails at the root will still be tried numerous times since the decision tree confidence increases very slowly for such a complex hierarchy. The solution is to adjust the decay based on not only the complexity of the plan sub-tree but also on the complexity of the sub-tree at the point of failure. Previously, the decay was calculated as $\delta = \delta * [1 - (1/\eta_p)]$ where η_p is the complexity of the calculating node. The modification means that the decay is now calculated as $\delta = \delta * [1 - (1/(\eta_p - \eta_o)]$ where η_o is the complexity of the plan at the point of failure. Intuitively, this means that the decay is faster (i.e. confidence grows more quickly) the higher up the complex sub-tree the failure occurs.

2.4 Accuracy Improvements

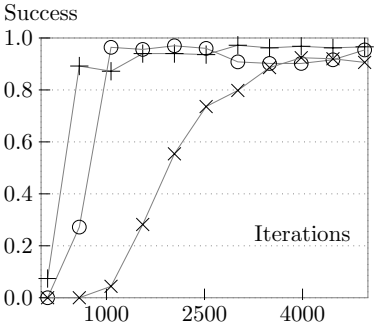
Figure 1 shows the results reproduced from [1]. There are two reasons why the $ACL + \Omega$ performance of Figure 1(c) does not converge to 1.0. The first reason is what was indicated in [1] — that the way a sub-goal is resolved may influence the parent failing or succeeding. This is because a sub-optimal resolution may increase the recursion level beyond what is legally allowed (user specified bounded recursion limit). The second reason is that decision tree classification is not always perfect even for previously successful worlds. To improve this, the following options are now passed to the $J48$ algorithm: $-C\ 0.5 - M\ 1$. See weka $J48$ documentation for details.

2.5 Testing

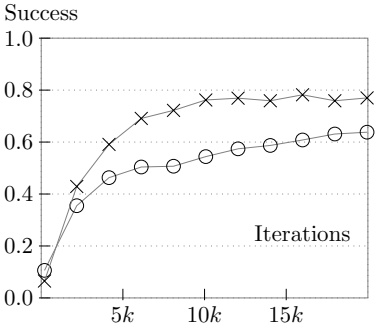
The individual recursive level tests reported in Figure 1(a) and Figure 1(b) and the full test reported in Figure 1(c) and Figure 1(d) are reproduced from [1] and use a *subset* of all possible goal/setup configurations for each experiment. This has now been extended to include *all* possible goal/setup configurations per recursive level.



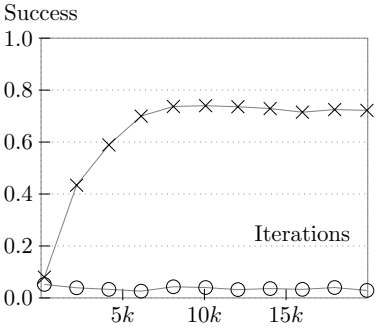
(a) ACL with solutions at recursion levels one (pluses), three (circles) and five (crosses).



(b) $ACL + \Omega$ with solutions at recursion levels one (pluses), three (circles) and five (crosses).



(c) ACL (circles) and $ACL + \Omega$ (crosses) schemes with applicability threshold 0%



(d) ACL (circles) and $ACL + \Omega$ (crosses) schemes with applicability threshold 20%

Figure 1: Agent performance under ACL and $ACL + \Omega$ schemes. Each point represents results from 5 experiment runs using an averaging window of 100 samples. Reproduced from [1].

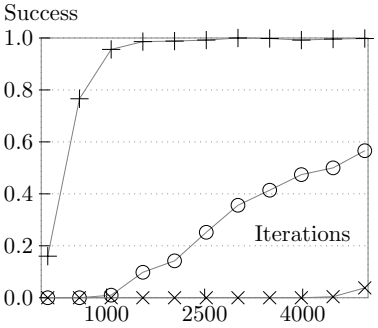
3 Results

The final results after all the listed changes are shown in Figure 2. The repository revision for the final changes is $svn : 323$.

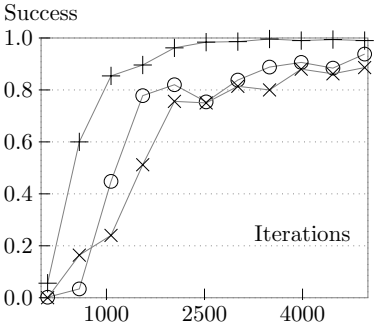
3.1 Limitations

Two known limitations in the treatment of recursive goals are:

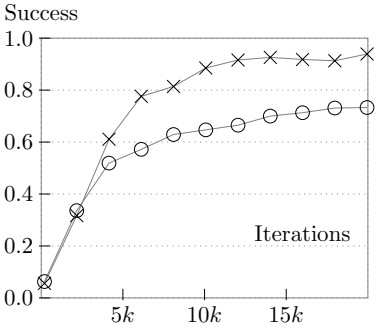
- Only one recursive goal-type G is supported in the code. This is sufficient for the simple Towers of Hanoi problem of [1] but will have to be extended for domains where multiple recursive goal-types are possible.
- Only $G \rightarrow P \rightarrow G$ recursion is supported where a plan posts the same goal that it handles. More complex interactions such as $G_1 \rightarrow P_1 \rightarrow G_2 \rightarrow P_2 \rightarrow G_1$ require global knowledge of goals in the system at the plan level that we do not support currently.



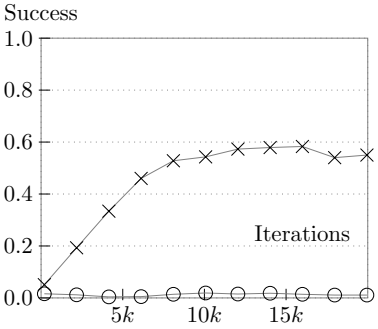
(a) ACL with solutions at recursion levels one (pluses), three (circles) and five (crosses).



(b) $ACL + \Omega$ with solutions at recursion levels one (pluses), three (circles) and five (crosses).



(c) ACL (circles) and $ACL + \Omega$ (crosses) schemes with applicability threshold 0%



(d) ACL (circles) and $ACL + \Omega$ (crosses) schemes with applicability threshold 20%

Figure 2: Agent performance under ACL and $ACL + \Omega$ schemes. Each point represents results from 5 experiment runs using an averaging window of 100 samples. Repository revision 323.

References

[1] D. Singh, S. Sardina, L. Padgham. Extending BDI Plan Selection to Incorporate Learning from Experience. Submission under review for the Special Issue on Hybrid Control of Autonomous Systems (HYCAS), Journal of Robotics and Autonomous Systems, 2010.

[2] D. Singh, S. Sardina, L. Padgham, S. Airiau. Learning Context Conditions for BDI Plan Selection. Submission under review for Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS), 2010.

[3] S. Airiau, L. Padgham, S. Sardina, and S. Sen. Enhancing Adaptation in BDI Agents Using Learning Techniques. In *International Journal of Agent Technologies and Systems*, 2009.