

Resolving the Horizon Problem in Planning As Modelling

By Paul Almond, 30 March 2007

Website: www.paul-almond.com
Email: info@paul-almond.com

© Copyright Paul Almond, 2007. All Rights Reserved.

Resolving the Horizon Problem in Planning As Modelling

By Paul Almond, 30 March 2007.

Introduction

Previous articles suggested the *planning as modelling* approach to planning in artificial intelligence (AI). This provides planning in an AI system by using the AI's modelling system to produce probabilistic *predictions* of future behaviour equivalent to *planning* of future behaviour. Planning as modelling was discussed in *Programming of Planning as Modelling in AI* [1], *Planning as Modelling in AI* [2], *How AI Would Work* [3], *Occam's Razor Part 9: Representation and Planning of Actions in Artificial Intelligence* [4] and *AI as a Boundary System* [5].

Planning as modelling as described previously is compromised by a "horizon" problem. Planning as modelling is supposed to limit the searching required for optimum behaviour according to the likelihood of different possible futures. A course of action could be found desirable after being considered part of an unlikely future, the previous consideration that the behaviour is unlikely meaning its selection was based on a shallow, and therefore possibly unreliable, search.

This article will modify the planning as modelling approach to deal with this problem.

Previous Modification of Planning As Modelling

In earlier articles about planning as modelling [3,4], the search performed by the output vectoring system was described as a search of the set of possible future sequences of outputs: inputs were omitted. In *Planning as Modelling in AI* [2] I corrected this, describing the output vectoring system as searching the set of possible future sequences of inputs and outputs. This correction continues to apply.

How Planning As Modelling Works

Planning as modelling uses an AI system's modelling system to perform both modelling and planning functions without a separate planning system.

The modelling system observes past inputs and outputs of the AI system and makes probabilistic predictions of future inputs *and outputs*. *The AI system is predicting what will happen in reality and its own behaviour*: planning as modelling makes no distinction between them as the system's observations of its own outputs are treated as just more inputs about which predictions can be made.

When an output is required, a separate system, the *output vectoring system*, searches the set of possible future sequences of inputs and outputs that the AI system can make and receive to determine the optimum output. This search can be a tree search – as with chess algorithms [6], in which case each branch corresponds to a particular input or output being received or made at some instant, the search tree exploring possible future input and output events in chronological order.

The search does not prioritize all possible sequences of future inputs and outputs equally. Instead, *the search is constrained by the modelling system's predictions*. Likelier futures are explored in greater detail than less likely ones. The likelihood of a particular future is obtained by multiplying together the probabilities of the individual inputs and outputs occurring as they do in that future and the modelling system provides these probability values.

Planning as modelling also uses *prioritization control outputs* – special pseudo-outputs generated by the AI system, observed by it and predicted by it in the same way as conventional outputs. Instead of being sent to the outside world as conventional outputs they act on the modelling system to control its prioritization of computing resources to deal with the *carpet texture problem*. Prioritization control outputs are not important in this article.

Use of a Tree Search in Planning as Modelling

If a tree search is used then the constraint from the model can be applied using a *cut-off probability*. The tree search computes a *running probability* for each node. At the start of the search tree this has a value of 1, and each time the search tree goes down a new branch for a particular input or output having a particular value then the running probability is multiplied by the probability of the input or output occurring with that value (provided by the modelling system) to get the running probability for the new node. The running probability therefore represents the probability that a sequence of inputs and outputs will occur to produce the situation corresponding to a particular node of the search tree. Paths through the search tree are prioritized according to the running probability: a path is never examined if one with a lower running probability is discarded. New branches of the search tree are only extended provided that the running probability for this node does not exceed the cut-off probability. Terminal nodes of the search tree are evaluated by a situational evaluation function which scores the desirability of the situation described by the model. This score is “backed-up” the search tree.

There could be different ways of implementing planning as modelling, and I have tried to generalize the solution to the horizon problem as much as possible, but much of the process is inevitably described in an implementation specific way. This article will focus on dealing with the horizon problem when planning as modelling is implemented as a tree search and, more specifically, when a tree search uses a cut-off probability and running probability.

Effect of the Running Probability on Search Depth

When a cut-off probability is used the running probability at a node in the search tree affects the depth to which the search will be extended from that node. If the running probability is low then its closeness to the cut-off probability means that, when the search is extended past this node, the effects of later inputs and outputs will tend to reduce it to the cut-off probability before it gets much further. A low running probability at a node therefore indicates that the search will not go much further and a high running probability indicates that the search can go much deeper.

This idea, though used in this article, is a simplification. When the running probability is low there will be some sequences of future inputs and outputs with high probability that do not reduce the running probability much, prolonging the search, though it must be narrow: for any splitting into

many branches most of them will lack high probability and use up the remaining running probability, which is important to limiting the searching. When the running probability is high there will still be sequences of future inputs and outputs with low probabilities that reduce it to the cut-off probability quickly. Saying that the running probability indicates how much deeper the search can go beyond a particular node is therefore about the overall tendency of the search, rather than every possible path through the search tree.

How the Horizon Problem Occurs

In planning as modelling, the output vectoring system searches deeper into likelier futures. The likelihood of futures (at least with regard to selecting different outputs) is related to their desirability. If the system predicts that it is unlikely to behave in a certain way so as to cause a particular future then that is equivalent to saying that, based on past analysis of the system's behaviour, the system will find that future undesirable. We expect less likely futures to have lower scores and when we find that this is so at a particular node we need not waste processing resources on looking further.

A problem possibly occurs, however, if the shallow search for an unlikely future actually returns a higher score than the deep search for a likely future.

For example:

Let us say that for a particular output being made in some hypothetical future situation, the modelling system gives a 99% probability of an output of “1” and a 1% probability of an output of “0”.

The prediction of a 99% chance of an output of “1” being made is equivalent to a prediction of a 99% chance that an output of “1” will be found the more desirable output, based on the system's previous behaviour. There is, however, a 1% chance that an output of “0” will be found preferable.

Because of the different probabilities, the running probability maintained by the output vectoring system has different values as the output vectoring system's tree search goes down each of these paths. The running probability will change further in various ways as branching occurs, but – all else being equal – the search down the “1” path will be deeper because it has “used up” less of the running probability.

What if the score from the shallow search down the “0” output path, backed-up to this node in the output vectoring system's search tree, is *better* than the score backed-up from the search down the “1” output path? This would suggest that, although there was a 99% chance against it, the output vectoring system has determined that the better output to make at this node is “0”. The problem with this is that the score for the “0” output was returned by a shallower search and could therefore be less reliable than the score from the search down the “1” path.

This could compromise the behaviour of the system. It could select a behaviour that its modelling system considers less likely, and therefore less likely to be desirable, because the short-term results of that behaviour seem better than the long-term results for the more likely behaviour, causing the

behaviour to be favoured by the shallower search process for the less likely behaviour. The system is therefore experiencing a horizon problem.

Horizon problems are well known in tree searches, for example in chess algorithms [7]. An obvious solution would be to make *all* the search deep, but this is untenable. It would involve discarding planning as modelling completely because the whole idea of it is to use the modelling system to perform planning by having it constrain a search for optimum behaviour in this way. This article is about resolving this problem.

Why the Horizon Problem Needs Solving

From a mathematical perspective this situation is not too bad. There is nothing intrinsic to the system which must cause the horizon problem to dominate its behaviour. The horizon problem occurs when the short-term consequences of an action which is not explored deeply are significantly better than the long-term consequences, but this need not usually be the case. Although it is desirable to consider things in the long-term, on average we should expect short-term consequences to be similar to long-term consequences and any differences to balance out. Practically, however, the situation needs resolving. Some reasons for this are as follows:

- It causes the system's behaviour to be occasionally sub-optimal. Even if the system still works with reasonably efficiently, its efficiency should be maximized.
- The ability of the system to work properly in many situations may give humans high expectations of it, yet the horizon problem could cause errors in situations where the correct behaviour would be natural to humans who would expect the system to work flawlessly. This could cause overconfidence in the system. It could also make it fit badly into a world designed for humans and human decision-making.
- The weakness caused by the horizon problem could be exploited by an intelligent opponent in some game or confrontational situation such as chess.

An Example of the Problem

Suppose that some node in the output vectoring system's search tree corresponds to a future output event with two possible output values: "0" and "1".

The output vectoring system obtains the relevant probabilistic predictions from the modelling system, which gives probabilities of 99% that the output will be "1" and 1% that it will be "0". The modelling system considering a "1" output more likely at this node means that an output of "1" is more likely to be found preferable in this hypothetical situation.

The search will branch off from this node with separate paths for outputs of "0" and "1" and will branch out from these new nodes. Two scores will be backed-up to this node from the "0" and "1" paths. The output vectoring system will decide which score is preferable – in the case of an output node at least. Let us consider two possible situations for the returned scores:

In the first situation a score of 27 is returned for the "1" output and a score of 15 for the "0" output. The search following on for the "0" output was shallower so could be considered less reliable, but it

returned a low score anyway, so it is not worth spending more time on it. The expected result obtained for the “1” output appears better considered deeply in comparison with a shallow consideration of the consequences of the “0” output. We could be wrong: with a deeper search the “0” output might turn out to be better, but it had all the chance we were prepared to give it and it was not worth spending more processing resources on it. This may seem risky, but is no different from any other differential allocation of resources to research. A human manager, for example, will devote more resources to considering suggestions that seem better.

Let us now consider a second situation in which the search for the “1” output returns a score of 15 and the search for the “0” output returns a score of 27. This is a different situation. We have invested more computation in checking out the future consequences of making the “1” output than in checking out the future consequences of making the “0” output. We thought it was 99% likely that the “1” output would turn out to be better, but the “0” output has turned out to be better. We have expended less computational resources on checking out the “0” output: we used a shallower search due to the initially low value of the running probability. We are therefore going to choose to make the “0” output because it looks better, having only shallowly considered it. On deep consideration we may find that the score for the “0” output is less than the score for the “1” output. This is not the same as the previous situation because we expected the “1” score to be better anyway. Some way of dealing with this is needed.

Resolving the Situation at Output Event Nodes

The Method

Dealing with the horizon problem with outputs involves detecting paths through the search tree where the computation relating to certain output events involves unreliable scoring due to the horizon problem and repeating parts of the computation as if the probabilities for some output values are higher.

For a tree search, the probabilities for different output values indicate how deep the search will tend to be extended from the current output event node. For example:

Suppose at an output event node the running probability is currently 0.0012 and there are two outputs: “0” with a probability of 0.01 and “1” with a probability of 0.99. When the search tree goes down the “0” branch, the running probability is multiplied by 0.01, giving $0.0012 \times 0.01 = 0.000012$, and when it goes down the “1” branch it is multiplied by 0.99, giving $0.0012 \times 0.99 = 0.001188$. After going down these branches the search spreads out for different values of future inputs and outputs. Some of these paths will terminate quickly while others go deeper, depending on the probability values and their effect on the running probability and how quickly it reaches the cut-off value, but all else being equal, searches will tend to be shallower for the “0” value because it starts from this node with a much lower running probability.

The probability for an individual output value at a node therefore indicates the reliability of the backed-up score from the extension of the search for that output value.

We do not have to use the probability from the modelling system for determining the contribution of a particular output value to the constraint on a path through the search tree (that is to say, how close it is to being cut off). On detecting that a particular output value is being assumed to be the output that the system would make for a particular output event because the score backed-up from the search associated with it is higher, we can use a different, *higher* probability, for the purpose of determining the contribution of that output value to the constraint. This higher probability is the *effective probability*.

When using a cut-off probability with a search tree and the score selected at an output node is regarded as being unreliable, due to being associated with a low probability output, we can repeat the extension of the search from the current node for that output value, multiplying the running probability instead by a higher probability value – a higher effective probability – deepening the search, to determine if that output value is still the best.

The Need for Availability of Secondary Searching At All Nodes

Such a secondary search needs to be available at each node of the output vectoring system's search tree – not just at the first node when selecting the very next output to be made.

At each search tree node the results from searches down different paths from that node are combined to give a single score, implying an assumption that the scores are equally reliable.

Considering output nodes, for example:

For a future output event node, backed-up scores for each possible value of that output are sent up to that node. The best scoring output value is selected as the output which would actually be made at that node, meaning that the score corresponding to that output value is regarded as the score which should be backed-up from that node.

This means that at every output event node the system is effectively modelling its future choice of the best output to make in that situation by comparing the backed-up scores from the paths through the search tree following on from that node. If, however, the reliability of different searches varies depending on their depths then the scores backed-up to an output node for some output values could be less reliable than the scores backed-up for other output values and the selection of some output values based on them having the highest scores could be less justified.

Details of the Method

Step 1:

Let $\text{Probability}_{\text{Max}}$ be the probability value given by the modelling system for the most likely output value.

$\text{Probability}_{\text{Max}}$ does not change in subsequent steps.

Step 2:

Extend the search from this node for each possible output value as described previously for planning as modelling [1,2]. Make a table showing, for each possible output value, the backed-up score and “Deep Search” – a flag (yes/no) value for each possible output value indicating whether the score for it was obtained using a deep search. A deep search is defined as one in which the search was extended from the current node, for that output value, with an effective probability of $Probability_{Max}$. The effective probability is the probability value assumed for that output value, for the purpose of controlling the depth of the search, when extending the search from this node for that output value. (If a cut-off probability is used then, as the search goes down the branch for each possible output value, the running probability becomes running probability x effective probability.)

Step 3:

If the highest score in the table corresponds to an output value for which the “Deep Search” flag is set (that is to say, it was extended from the current node with an effective probability of $Probability_{Max}$) then assume that the resulting score at this node is this score and that the output selected at this node is the output value corresponding to it.

If the highest score in the table corresponds to an output value for which the “Deep Search” flag is *not* set (that is to say, the search was extended from the current node with an effective probability *less* than $Probability_{Max}$) then extend the search from this node again for this output value, but with an effective probability of $Probability_{Max}$, setting the “Deep Search” flag for this output value to indicate that the search has been extended from the current node with an effective probability of $Probability_{Max}$, and update the score backed-up for this output in the table. (If a cut-off probability is used then, as the search goes down the branch for this output value again, the running probability becomes running probability x the new effective probability.) Repeat Step 3.

Example 1:

At an output event node we have these possible output values with the probabilities from the modelling system:

Output Value	0	1	2	3
Probability	0.15	0.16	0.05	0.2

$Probability_{Max}$ is 0.6.

We will store the “Deep Search” flag for each possible output value, indicating whether or not the score that we have for it was obtained using a deep search. A deep search is one in which the search was extended from the current node, for that output value, with an effective probability of $Probability_{Max}$. We extend the search as in the previously described planning as modelling process. The probability given by the modelling system is used as the effective probability in each case, and the only output value for which the effective value is $Probability_{Max}$ is “1”, the most likely output. If a cut-off probability is used then in each case the running probability is multiplied by the

effective probability to give the running probability at the next node. For example, if the running probability is 0.07 then, with a cut-off probability in use, the running probabilities are as follows:

Output Value	0	1	2	3
Probability	0.15	0.16	0.05	0.2
Running Probability	0.07×0.15 =0.0105	0.07×0.6 =0.042	0.07×0.05 =0.0035	0.07×0.2 =0.014

Suppose the scores backed-up to this node from the lower branches of the search tree give the following situation:

Output Value	0	1	2	3
Probability	0.15	0.16	0.05	0.2
Running Probability	0.07×0.15 =0.0105	0.07×0.6 =0.042	0.07×0.05 =0.0035	0.07×0.2 =0.014
Deep Search	N	Y	N	N
Score	15	27	13	20

The highest scoring output value is “1”, with a score of 27, for which the “Deep Search” flag is set: the search for it was extended from this node with an effective probability of $\text{Probability}_{\text{Max}}$, which is 0.6. Therefore, this score should be accepted. In this situation it is assumed that an output of “1” would be selected at this node and a score of 27 is assigned to this node.

Example 2:

Let us imagine a situation like the previous one, except with different scores backed-up to the current node:

Output Value	0	1	2	3
Probability	0.15	0.16	0.05	0.2
Running Probability	0.07×0.15 =0.0105	0.07×0.6 =0.042	0.07×0.05 =0.0035	0.07×0.2 =0.014
Deep Search	N	Y	N	N
Score	24	20	10	27

The highest scoring output value is now “3” with a score of 27. As the “Deep Search” flag indicates, this is no longer the most likely output and was obtained with an effective probability less than $\text{Probability}_{\text{Max}}$ – actually a value of 0.2. We therefore extend the search from this node again, down the path for the “3” output value, with an effective probability of $\text{Probability}_{\text{Max}}$, which is 0.6, setting the “Deep Search” flag. If a cut-off probability is used then the running probability for this output changes from 0.014 to $0.07 \times 0.6 = 0.042$. Suppose a score of 23 is backed-up from this search, so the situation is now like this:

Output Value	0	1	2	3
Probability	0.15	0.16	0.05	0.2
Running Probability	0.07x0.15 =0.0105	0.07x0.6 =0.042	0.07x0.05 =0.0035	0.07x0.6 =0.042
Deep Search	N	Y	N	Y
Score	24	20	10	23

The deeper search for the “3” output value caused the score to decrease and it is no longer the highest score. The highest score is 24, for the “0” output value. The “Deep Search” flag is not set for this score, however: it was only obtained with an effective probability of 0.15 – less than the Probability_{Max} value of 0.6 – so the search needs to be deepened with an effective probability of Probability_{Max} (0.6) and the “Deep Search” flag is then set for this output value. If a cut-off probability is used then the running probability for this output value now becomes 0.07x0.6=0.042. Suppose that when this is done a score of 19 is returned so the situation is now as follows:

Output Value	0	1	2	3
Probability	0.15	0.16	0.05	0.2
Running Probability	0.07x0.6 =0.042	0.07x0.6 =0.042	0.07x0.05 =0.0035	0.07x0.6 =0.042
Deep Search	Y	Y	N	Y
Score	19	20	10	23

The deeper search for the “0” output reduced its score, making it no longer the highest score. The highest score is now 23, for the “3” output, for which the “Deep Search” flag is set – this score was obtained with an effective probability of 0.6, which is Probability_{Max}. We therefore accept it. We assume that in the situation corresponding to this node the system would make an output of “3” and we assign a score of 23 to this node.

This example shows a lot of searching and some readers may wonder what the point of planning as modelling is if we have to do all this anyway. This situation is contrived, however, to show how the horizon issue could be resolved. Situations like this will be infrequent. When they occur, it does not mean an entire tree search must be performed: the decision to deepen the search is only taken at the current node. For any search deepened in this way the decision about whether or not to deepen the search must be taken at every subsequent node and most often it will not be deepened.

Alternative Implementation

The above method involves deepening the search by using the most likely output value’s probability as an effective probability, but a different effective probability value could be used.

Instead of using a flag (“Deep Search”) to indicate whether or not a search has been done with the higher effective probability, the actual value of the highest effective probability used for a given output value could be stored with it in the table. I avoided that here, mainly because of technical issues with rounding of numbers and comparison.

Special Case for Two Output Values

Planning as modelling will often be used in systems with only two possibilities for inputs and outputs – binary systems in which we can designate possible inputs or outputs as “0” or “1”. In a binary system, such a process is simpler:

Step 1:

Extend the search from this node for output values of “0” and “1” as described previously for planning as modelling [1,2], with the effective probability for each output value being the probability from the modelling system. (If using a cut-off probability, multiply the running probability by the probability from the modelling system.) Store the score backed-up to this node for each such path.

Step 2:

If the higher score is that associated with the output value assigned the *higher* probability by the modelling system then assume that this output value would be selected in the situation corresponding to this node and assign this score to this node.

If the higher score is that associated with the output value assigned the *lower* probability by the modelling system then perform a secondary extension of the search from this node, down the branch corresponding to this lower probability output value only. Instead of using the probability given by the modelling system for this lower probability output value to control the depth of this secondary search, use the probability value for the *higher* probability output value as the effective probability. (If using a cut-off probability, multiply the running probability by the probability for the higher probability output value, so the search has the same depth.) Use the new backed-up score to replace the score previously calculated for the lower probability output in the shallower search. Compare the scores for the higher probability and lower probability outputs again. Assume that the higher scoring output value is the one that would be selected in the situation corresponding to this node and assign the corresponding score to this node.

Example 1:

At an output event node in the search tree these are the probabilities from the modelling system:

Output Value	0	1
Probability	0.2	0.8

For each output value we extend the search from the current node, using the probability for that output value as the effective probability, meaning the “1” output has a deeper search.

Suppose the scores backed-up from these searches are as follows:

Output Value	0	1
Probability	0.2	0.8
Score	15	41

The higher score is for the “1” output and was obtained using a search with the higher effective probability of 0.8, so this score is accepted. It is assumed that an output of “1” would be made in the situation corresponding to this node and a score of 41 is assigned to this node to be backed-up to other nodes.

Example 2:

We have a system like the one in the previous example, with the same probability values, but when the search is extended down each path the scores backed-up to this node give the following situation:

Output Value	0	1
Probability	0.2	0.8
Score	25	22

The higher score is 25, for the “0” output, but we do not trust it because it was obtained with the lower probability as the effective probability. We therefore extend the search from this node down the “0” path again, this time with an effective probability of 0.8. Suppose that a score of 24 is backed-up to this node from this revised search. This is still greater than the other score of 22, so we would assume that an output of “0” would be made in this situation and assign a score of 24 to this node. Suppose instead, now, that a score of 21 is backed-up to this node from the revised search down the “0” path. This is less than the other score of 22 so we would now reject the “0” output, assuming instead that the output made in this situation would be “1” and assigning a score of 22 to this node.

Practical Considerations

The above method is idealistic. Its processing overhead may not be worthwhile in situations where the search corresponding to the selected output value is already *deep enough*.

For example, suppose we have two possible output values – “0” with a probability of 0.5001 and “1” with a probability of “0.4999 – and “1” has the higher score. The above method would require us to repeat the extension of the search for “1” from the current node with an effective probability of 0.5001, but this is not much improvement on 0.4999 for a significant amount of computation.

If the probability of the selected output is *close enough* to $\text{Probability}_{\text{Max}}$ we may choose *not* to perform a secondary search.

The ideas in this article are subject to pragmatism: they may be ignored in situations where the horizon problem is not serious enough to merit the processing to resolve it.

Resolving the Situation at Input Event Nodes

For input event nodes the approach will depend on how the score to be assigned to an input node is determined from the backed-up scores corresponding to different input values. Things may be more involved and I will not attempt a complete solution here. The problem, however, is likely to be less severe for inputs: it is probably outputs that we need to worry about. This is because the process of determining the score to be placed at the node is likely to involve some averaging which takes account of probabilities and when a score is unreliable by being backed-up from the path for a low probability input value then the averaging process will automatically tend to reduce its influence on the score to be assigned to the node. We may still want to take account of the reliability of scores, however, and how we do this may depend on how the scores associated with different input values are combined.

Input Event Nodes where the Score is the Highest Backed-Up Score

This is a situation in which the backed-up scores are used *optimistically* to determine the score to be placed at an input event node. All the scores backed-up to the input node from the paths corresponding to different input values are compared and the *highest* score assigned to that node. This will probably be too optimistic in most situations.

Scores are handled in such a situation in the same way as scores for output events, so the method of dealing with the horizon problem is the same as that used for outputs.

Input Event Nodes where the Score is the Lowest Backed-Up Score

This is a situation in which the backed-up scores are used *pessimistically* to determine the score to be placed at an input event node. All the scores backed-up to the input node from the paths corresponding to different input values are compared and the *lowest* score assigned to that node. This may be too pessimistic in many situations.

Scores are handled in such a situation in almost the same way as scores for outputs, except that instead of selecting the highest score we select the *lowest* score. The method of dealing with the horizon problem is the same as that used for outputs, except that any reference to the *highest* in a set of values is replaced by a reference to the *lowest*.

Input Event Nodes where the Score is the Mean Backed-Up Score

In this situation the aim is neither pessimism nor optimism.

In such a situation the horizon problem may not be important because the probability will be taken account of in calculation of the mean and low probability input values (resulting in shallow searching) will influence the average score less than higher probability input values (resulting in deeper searching). Further measures, however, may be considered necessary, or the situation may be complicated by being some compromise between averaging and the optimistic or pessimistic approaches. Such measures are beyond the scope of this article and, given the limited effect of the horizon effect with averaged input scores compared with its effect with output scores, and the way

in which the measures needed may depend on specific implementations of the AI system, I do not think a description of them is needed at this stage.

Possible Limitations

The approach to the horizon problem could have limitations due to tendencies which the situation may sometimes have. We need to consider what happens if the situation has a tendency to improve or deteriorate.

In some situations the short-term effects of most actions, including the best ones, may be generally better than long-term ones. In such situations, higher probability outputs, with the searching associated with them being extended further, have an automatic advantage. A lower probability output may be better in the long-term, but its long-term consequences will not be explored. This could cause some potentially useful behaviour to be missed, but it is not a disaster. The favoured behaviour is what the modelling system considers the most probable behaviour anyway – meaning it is likely to be the best available behaviour. Although technically a limitation, this is only an extreme example of what planning as modelling does anyway. We cannot search all the space of possible futures and we must occasionally miss opportunities.

In other situations the short-term effects of most actions, including the best ones, may be generally worse than long-term ones. In such situations the lower probability outputs, with the searching associated with them cutting off early have an initial advantage over higher probability outputs with deeper searching. This will not cause incorrect selection of outputs, because the approach described in this article, or something similar, will cause secondary, deeper searching for such outputs. If, though, there is a situation in which there is a tendency for short-term searches to give better results then this secondary searching could occur continually, negating the advantage of planning as modelling. Most situations would not be like this, but if we decided that this was a problem we could deal with it. One method could be to return separate shallow and deep search scores for likelier output values – maybe even a number of scores – and only even to consider a lower probability output if its score is better than scores returned by *shallow* searching for likelier output values.

Conclusion

The horizon problem in planning as modelling occurs because search depth depends on which input or output values occur. Multiple scores backed-up to a node can vary in reliability. For scores backed-up to an output event node it may be that the search should be deepened because the “best” score is associated with a low probability output value, the low probability of which makes any searching of futures following on from it unreliable. For scores backed-up to an input event node it may be decided that some scores are disproportionately affecting the result for that node, given their reliability.

A way of dealing with this could be to deepen searches extending from a node selectively if it has been determined that they are a problem.

This article is not intended as the final word on resolving the horizon problem in planning as modelling. Selective deepening of recursive tree searches is an area about which there is already a lot of study, such as in chess [8], and a number of solutions will probably be found.

References

[1] Web Reference: Almond, P. (2006). *Programming of Planning as Modelling in AI*. Retrieved 28 December 2006 from <http://www.paul-almond.com/ProgrammingOfPlanningAsModellingInAI.pdf>.

[2] Web Reference: Almond, P. (2006). *Planning as Modelling in AI*. Retrieved 26 November 2006 from <http://www.paul-almond.com/PlanningAsModellingInAI.pdf>.

[3] Web Reference: Almond, P. (2006). *How AI Would Work*. Retrieved 4 September 2006 from <http://www.paul-almond.com/HowAIWouldWork.pdf>.

[4] Web Reference: Almond, P. (2006). *Occam's Razor Part 9: Representation and Planning of Actions in Artificial Intelligence*. Retrieved 29 July 2006 from <http://www.paul-almond.com/OccamsRazorPart09.pdf>.

[5] Web Reference: Almond, P. (2006). *AI as a Boundary System*. Retrieved 17 September 2006 from <http://www.paul-almond.com/AIAsABoundarySystem.pdf>.

[6] Levy, D.N.L. (1984). *The Chess Computer Handbook*. London: Batsford. Chapter 3, pp38-52.

[7] Levy, D.N.L. (1984). *The Chess Computer Handbook*. London: Batsford. Chapter 6, pp82-84.

[8] Heinz, E, A. (2000). *Scalable Search in Computer Chess: Algorithmic Enhancements and Experiments at High Search Depths*. Vieweg Verlag. Chapter 0, p20.