Supplementary Material (Goal Recognition with Timing Information)

Chenyuan Zhang,[1] **Charles Kemp,** [1] **Nir Lipovetzky** [1]

[1] The University of Melbourne
chenyuanz@student.unimelb.edu.au, cskemp@gmail.com, nirlipo@gmail.com

Goal Recognition Algorithm with Timing

Here we present how to integrate timing information in the goal recognition algorithms. In the paper we use the posterior $P(g|D, Prior, O)$ as Ramırez and Geffner (2011) to define the output of a goal recognition algorithm. In the following section, we will use $P(O|g)$ to represent the probability value $P(O|D, g)$ and $P(g|O)$ to represent $P(g|D, prior, O)$ as we have the same planning domain D, and assume uniform prior for a GRT instance.

To compute $P(g_i|O) = P(g_i)P(O|g_i)$ for each potential goal $g_i \in G$, then the challenge of the problem is how to evaluate the value $P(O|g)$ with the assumption of uniform prior.

We assume that action and planning time only depend on the current state and the real goal (markovian), and planning time and action are conditionally independent on states and goals. So we can decompose the likelihood $P(O|g)$ as:

$$\begin{aligned}P(O|g) =&P(\langle a_0, t_0\rangle, ..., \langle a_m, t_m\rangle|g)\\ =&\prod_{j=0}^{m} P(t_j|g, \langle a_0, t_0\rangle, ..., \langle a_{j-1}, t_{j-1}\rangle)\\ &P(a_j|g, \langle a_0, t_0\rangle, ..., \langle a_{j-1}, t_{j-1}\rangle, t_j)\end{aligned}$$

Given the transition function in a deterministic environment and known initial state s_0,

$$\begin{aligned}&\prod_{j=0}^{m} P(a_j|g, \langle a_0, t_0\rangle, ..., \langle a_{j-1}, t_{j-1}\rangle, t_j)\\ =&\prod_{j=0}^{m} P(a_j|g, s_j).\end{aligned}$$

Similarly,

$$\prod_{j=0}^{m} P(t_j|g, \langle a_0, t_0\rangle, ..., \langle a_{j-1}, t_{j-1}\rangle) = \prod_{j=0}^{m} P(t_j|g, s_j).$$

Conversion between real planning time and observed planning time

We want to fill in the gap between the real planning time and observed planning time as we mentioned in the Timing Component subsection. We can use discounted accumulated sum to calculate the actual planning time for each state s_i from observation sequence $t_i, t_{i-1}, ..., t_1$ as $t(s_i) = t_i + \lambda t_{i-1} + \lambda^2 t_{i-2} + ... + \lambda^{i-1} t_1$, where λ is a constant factor representing the discount for previous observed planning time. For example, if we observe the agent spend 100 time unit on the first step, 200 time unit on the second step, 50 unit on the third step, then the real planning time for the third step is $t = 50 + 200 * 0.8 + 100 * 0.8^2 = 274$ if we choose the discount rate $\lambda = 0.8$. The intuition under this mechanism comes from both computer science and cognitive science: from the perspective of online planning algorithm, this mechanism can be considered as reusing the previous subtree (Powley, Cowling, and Whitehouse 2014) and many evidence in cognitive science show human use a similar process to solve complex problems (Krusche et al. 2018; Van Opheusden et al. 2017). Under this framework, we can also model the memoryless agent by just setting the discount factor as 0.

To test the validity of this conversion method, we looked at the similarity (i.e. cosine distance) between the timing sequence generated by the agent model without memory mechanism used in our experiment and the agent model with memory mechanism. The planning times generated by the memoryless agent is the actual planning times while the agent model with memory mechanism generates the observed planning times that need to be converted. We considered instances where both agents have the same sequence of actions, where only BLOCKSWORLD and EASYIPCGRID fall under this category. In BLOCKSWORLD, this conversion (λ is set to 0.8 empirically) brings up the similarity from 0.75 to 0.96 on 30 instances. In EASYIPCGRID, the similarity increases to 0.98 from 0.59 by the same conversion on 34 instances. These results implies that the conversion method are effective to map observed planning times generated by an agent with memory mechanism to actual planning times.

Demonstration of Agent Model

Here we show how the agent model generate human-like planning time using a value-based example. Here we simply assume the cost of action is 0 and the value denote the estimated expected utility of the node. In this configuration the best successor state would be the maximal value rather than minimal value, thus the definition of state importance becomes to

$$I(s) = \frac{v_{s,a}}{(1+\beta)v_{s,a} - v_{s,a'}}. \tag{1}$$

.

In this example we assume $\beta = 0$ and $\gamma = 10$ for simplicity. We start with the current state s0 (fig 1a). In the first iteration we expand the start node by generating all the applicable actions a_1, a_2, a_3 and the initial values by the heuristic function are $5, 3, 1$ respectively (fig 1b). Now the state importance $I(s_0) = \frac{5}{5-3} = 2.5$, stopping probability $Prob_{stop} = \frac{1}{1+2.5*10*exp(-1/2.5)} \approx 6\%$. Assume we continue to next iteration, in this iteration we choose expanding s1 by generating two successor states s_4, s_5 and the value of $s1$ is updated to reflect the average gain of all visits $(5 + 4 + 9)/3 = 6$, noting that it also includes the value obtained (i.e. 5) in the initial visit (fig 1c). Now the state importance $I(s_0) = \frac{6}{6-3} = 2$, and the stopping probability increase to $Prob_{stop} = \frac{2}{2+2*exp(-2/2)} \approx 21\%$. Assume we still continue, and after next iteration (fig 1d) the stopping probability becomes to $Prob_{stop} = \frac{3}{3+1.5*10*exp(-3/1.5)} \approx 60\%$. And in next iteration when we expand s_3 we will use a_3 as the second best action rather than a_2 (fig 1e), thus the state importance increase to $6/(6-3) = 2$ and the stopping probability becomes to $Prob_{stop} = \frac{4}{4+2*10*exp(-4/2)} \approx 60\%$. If we still need to continue (very unlucky), we will use UCB rule to choose s_1 to visit then s_5 (fig 1f) and this time the state importance remains 2 and the stopping probability becomes to around 75%. Then assume the stop signal is triggered we can stop and the planning time for this state is 5.

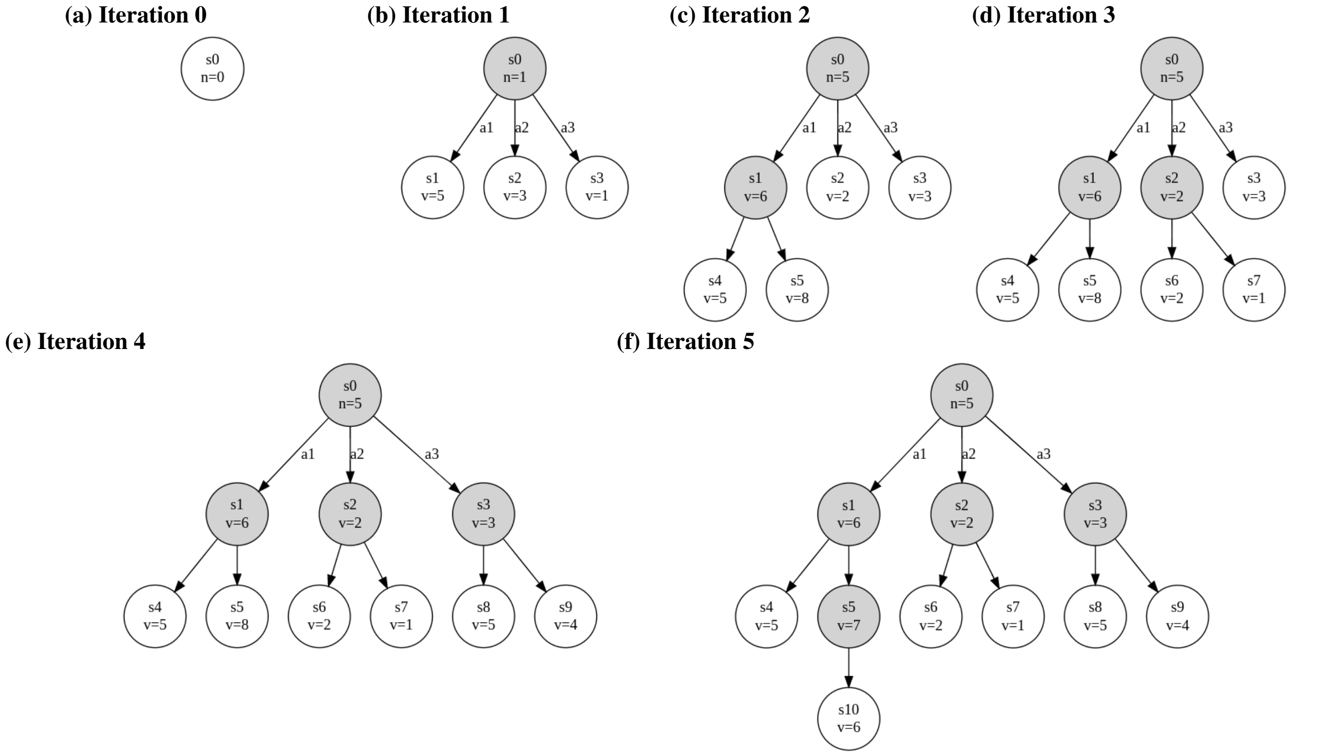
References

Krusche, M. J.; Schulz, E.; Guez, A.; and Speekenbrink, M. 2018. Adaptive planning in human search. *BioRxiv*, 268938.

Powley, E. J.; Cowling, P. I.; and Whitehouse, D. 2014. Information capture and reuse strategies in Monte Carlo Tree Search, with applications to games of hidden information. *Artificial Intelligence*, 217: 92–116.

Ramırez, M.; and Geffner, H. 2011. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *IJCAI*, 2009–2014. Citeseer.

Van Opheusden, B.; Galbiati, G.; Bnaya, Z.; Li, Y.; and Ma, W. J. 2017. A computational model for decision tree search. In *CogSci*.


(a) Iteration 0

(b) Iteration 1

(c) Iteration 2

(d) Iteration 3

(e) Iteration 4

(f) Iteration 5

Figure 1: The agent model run on a value-based example. The dark circle represent the node expanded and the light circle represent the node generated but not expanded yet. The value v of the node is initialized as some heuristic function of the corresponding state and then updated to reflect the average value of all visits passing through the node.