Introduction to LaTeX

Author's Name

August 13, 2018

1 Planning Task

$$\pi = (V, I, O, \gamma) \tag{1}$$

- V: finite set of state variables.
- I: initial state varaible evaluation.
- O: set of operators of the form (χ, e)
- γ: goal variable valuations.

2 Moral Planning Task

$$\pi = (V, I, O, \gamma, u) \tag{2}$$

- u: A moral utility function (returning a double). u can provide a moral label to any combination of actions, states, and variable assignments.

3 Combined Moral Planning Task

Assuming sequential turns:

$$\Pi = (A, \Xi, T) \tag{3}$$

- $\Xi = ((A_1, \pi_1), ..., (A_n, \pi_n))$: applies indexable labels to each agent in the task.
- T: Run-time or compile-time turn selection function that assigns the *active agent* at time t:
  - Compile-Time T: parameters are (A, Ξ), returns vector of agent turn probabilities for times $1, ..., T-1$. For independent turns, final result is $n \times T$ matrix. Problem with branching probabilities when actions have dependence on previous actions (as hard as reinforcement learning [would have to return transition matrix for each probability]).

– Run-Time T: returns vector of transition probabilities given some combination of: current state (s), Ξ, A, and previous (turn agents, action, state sets). (Probably easier than compile-time for cases where transition probabilities are state-dependant).

4 Evaluation Procedure Run-time

Given $\Pi = (A, \Xi, T)$:

Gaol-invariant

When the goal does not affect agent decisions:

1. Determine current active agent X. $X = A[argmaxT(\Xi)]$.
2. Determine applicable action set $O_{app} \in O_X$.
3. Determine ethical and applicable action set $O_e \in O_{app}$.
4. Randomly select and apply some $o \in O_{app}$.
5. Repeat.

5 Multi-Agent Ethical Principles

NB: for this simplified ethical approach we assume that the agent is able to call a *predict function* $P(A_i, s, \Xi)$ which returns the next action a' another agent X' will take using relevant information about that agent's s and π_X. Not doing this leads to highly complex recursive equations which I hope to tackle later.

5.1 Deontological Approach

For the deontological approach it is sufficient for active agent X to select $o \in O_X$ such that $u_X(o) \geq 0$.

5.2 Next state before active agent acts again

Remember, we assume $P(A_i, s, \Xi)$ is known (or feasible to compute) and artificially bound the maximum number of time steps T.

Introduce function Q which determines the final state before the current active agent is able to act again:

currentAgent = A[argmax(T(currentState,Ξ))];
while *currentAgent!=activeAgent* **do**
    currentState=app(currentState, P(currentAgent, currentState, Ξ));
    currentAgent= A[argmax(T(currentState,Ξ))];
end
return currentState;

Algorithm 1: Final state before active agent can act again

Looks daunting, but has complexity linear in the number of agents (when taking turns). $O(n)$. Space complexity is constant (or at least linear in the number of variables) if current state denotes a single state, rather than a set of states based on probabilities.

5.3 Utilitarianism

Maximize own resultant states

When bounded, the ethically permissible action for agent X is to find an applicable action which maximizes:

$$max \sum_{t=0,t+n}^{T} u_X(s_t, o_t)$$

$$o^* = argmax_{o \in O_X} \left[max \sum_{t=0,t+n}^{T} u_X(Q(...), o_t) \right]$$

Maximize all agent resultant states

In this formulation the utility of all agent state action pairs is considered. (Note that we still only consider utility as defined by the active agent.)

$$o^* = argmax_{o \in O_X} \left[max \sum_{t=0,t+1}^{T} u_X(s_T, o \in O_{T[...]}) \right]$$

For this case we can modify the Q algorithm to create a new function R which returns the total utility of the non-active agents, under the same assumptions as in the Q algorithm.

```
totalMoralUtility=0;
currentAgent = A[argmax(T(currentState,Ξ))];
while currentAgent!=activeAgent do
    totalMoralUtility=totalMoralUtility+u(currentState,
      P(currentAgent, currentState, Ξ));
    currentState=app(currentState, P(currentAgent, currentState, Ξ));
    currentAgent= A[argmax(T(currentState,Ξ))];
end
return totalMoralUtility;
```

Algorithm 2: Total moral utility before active agent can act again

$$o^* = argmax_{o \in O_X} \left[max \sum_{t=0,t+n}^{T} u_X(Q(...), o_t) + max \sum_{t=0,t+n}^{T} R(...) \right]$$