which is self-absorbing (as all single-element beliefs are self-absorbing). Therefore, the algorithm takes \(O(\min(H,K)K^3)\) running time.

6 Multiple Alternate Action IDPs

We next consider IDPs with multiple alternate agent actions, namely \(N > 1\). We now need to maintain a belief state over the joint of the alternate agent actions crossed with the incentives. As stated in the definition of the IDP, we assume we are provided with an initial probability distribution over the true incentives \(t_n\) for each alternate action \(a_n, P_0\). We know from the prior section that for an \(N = 1\) IDP are of the form 
\[B_{i,j}^1 = (0, \cdots, 0, p_{i,S}^j, \cdots, p_{j,S}^i, 0, \cdots, 0)\]
where \(S_1 = (i,j).\) We now extend the model to handle multiple actions. Let
\[B_S = (B_S^n)_{n=1}^{N},\]
where \(B_S^n\) denotes the probability vector \((0, \cdots, 0, p_{i,S}^n, \cdots, p_{j,S}^n, 0, \cdots, 0)\) where 
\[S_n = (i,j)\]
and \(p_{i,S}^n\) is the probability that \(t_n = \delta_i\) given the initial joint distribution \(P_0\) and the possible incentive ranges \(S_n\) for each alternate agent action. \(S_n = (s_n, e_n),\) for all \(n \in \{1, \cdots, N\}.\)

**Theorem 6.1.** The number of states of IDP-MDP for \(N > 1\) is \(O(K^{2N})\) states.

**Proof.** Assume that the current state is \(B_S,\) where \(S_n = (s_n, e_n)\) for \(1 \leq n \leq N.\) Consider the case where incentive \(\delta_i\) is offered for action \(a_n.\) The agent will accept this offer with the probability that its true incentive \(t_n\) for \(a_n\) is \(\leq \delta_i,\) which is equal to \(\sum_{j \leq i} p_{j,S}^n.\) If the offer is accepted, then \(s_n \leq t_n \leq i,\) which implies \(S_n = (s_n,i).\) In addition, all alternate agent actions \(m > n\) with higher agent reward than alternate agent action \(a_n\) will also all accept \(t_n,\) so the incentive range of all such actions will also be updated \(S_n = (s_n,i)\) for all \(m > n.\) This defines a new \(S,\) which together with \(P_0\) completely defines the new distribution over the probability of each incentive for each alternate agent action. If the agent rejects the offered incentive, then \(s_n \geq i + 1,\) and so \(S_n = (i+1,e_n).\) All alternate agent actions \(m < n\) with a lower agent reward than action \(a_n\) will also reject the reward, which results in \(S_n = (i+1,e_m)\) for all \(m < n.\) There are \(N\) alternate agent actions, and we know from Theorem 5.1 that there are at \(O(K^2)\) states per individual action, therefore there are at most \(O(K^{2N})\) possible states. Due to the relationship among the agent costs and rewards, this number will often be much lower. \(\square\)

An IDP with \(N > 1\) alternate agent actions has many of the same properties as an IDP with \(N = 1\) actions, and therefore we can use algorithms similar to those described in the prior section (see the text and Algorithm 1) to solve the IDP-MDP with \(N > 1.\) Excluding the cost for the belief updates, this requires \(O(NK^{2N+1})\) and \(O(\min(H,NK)K^{2N+1})\) computation time for the infinite and finite horizon case, respectively. While this is computationally tractable for small \(N,\) this scales poorly as \(N\) increases.

To address this, we will now provide efficient approximate algorithms for computing a decision policy in IDPs with large \(N.\) Note that computing the probability of the next possible states, given an offered incentive \(\delta_k\) for alternate agent action \(a_n\) involves computing the marginal probabilities \(p_{k,S}^n\) that \(t_n = \delta_k\) given the current \(S.\) This is an \(O(K^N)\) operation due to summing over all other action-incentive probabilities. Hence, any approximation algorithm will take at least \(O(K^N)\) time due to the bottleneck of belief updating. This cost can be reduced by assuming structure in the joint probability distribution, such as using graphical models with bounded treewidth. We leave further exploration of this issue for future work.

We now define a new special form of a Markov decision process. Let a SEQ-MDP be an IDP-MDP with \(N > 1\) with the following additional restrictions on the allowable principal’s actions at different states:

1. For the initial state, the principal can offer any incentive for only alternate agent action \(a_1\) (assuming that initially \(S_1 = (s_1,e_1)\) and \(s_1 \neq e_1).\)

2. For any other state, the principal can offer any incentive for alternate agent action \(a_n,\) only when \(S_i = (s_i,e_i)\) and \(e_i = s_i\) for all \(i < n.\) In other words, \(t_i\) must be known for all alternate agent actions \(i < n\) before the principal can offer an incentive for \(a_n.\)

A SEQ-MDP restricts the possible decision policy set \(\Pi\) to policies that only provide incentives for alternate agent actions with higher cost to the principal only after finding the true incentive for the lower cost actions. Note that the SEQ-MDP policy may not identify the true incentive for all actions, but can decide to stick with a previously identified \(a_n, \delta_k\) pair.

We now analyze the number of states in a SEQ-MDP. Consider that at the present state the principal is offering incentives for alternate agent action \(a_n.\) We know that \(S_i = (s_i,e_i)\) for \(i > n\) and \(s_i = e_i\) for \(i < n.\) Define \(c_i + \delta_j\) as the minimum action cost + true incentive across all \(i < n.\) Since we would never take any other \(i' < n\) with a higher cost, \(c_i + \delta_j\) is sufficient to summarize all useful information from prior offered actions. Therefore we can represent a state in the SEQ-MDP as the tuple \((a_i, \delta_j, B_{S_i}).\) For each offered alternate agent action, there can be \(O(K^2)\) states, as in the IDP-MDP with \(N = 1.\) We also must track the current minimal cost alternate agent action and offered incentives, and there are \(O(NK)\) such combinations. Finally we also have to monitor

---

423