First we would like to thank all the reviewers for their time and comments.

Reviewer 1: **(a)** We conjecture DAC will outperform AHP when it is important to coordinate among different options. This could be the case when the task is complicated. In our experiments, Cheetah and Walker (1&2), where the advantage of DAC+PPO is clear, are usually considered more complicated than CartPole and Reacher. **(b)** PPOC tries to combine OC with PPO and is compared in our paper, although that combination is not theoretically sound. Combining IOPG and PPO is even harder. As discussed by Smith et al., this combination requires products of importance sampling ratios to correct trajectory distributions, which could yield high variance. **(c)** Thanks for pointing out other minor issues. This is indeed very constructive and we shall fix typos and make statements clearer accordingly in the final version.

Reviewer 2: **(Related Work)** We would like to clarify that our work builds on the option framework and we compared to all related works (e.g., OC, IOPG, PPOC). Particularly, we also compared with AHP, which is 8 years old and was never evaluated in deep RL setting. As other HRL frameworks usually do not have option termination functions, their ability to express a hierarchy is different from the option framework. They thus are not directly comparable. **(Performance)** In the single task setting and the first phase of the transfer learning setting, DAC is similar to other baselines. This phenomenon is expected. Usually we do not expect an option-based method to outperform option-free counterparts in a single task, as learning options adds overhead unless useful options can be learned easily. This performance similarity is also observed in previous option-based works (e.g., OC, IOPG, PPOC). In the second phase of the transfer setting, DAC outperforms all baselines in 50% tasks and is no worse than any other baseline in the other 50% tasks. We consider this to be a solid achievement but we'll tone down our wording and make it less ambiguous in the final version. **(Visualization)** We visualize option structures following the convention of OC and IOPG. All those visualizations are descriptive and aim to understand the behavior of the agent. **(AHP)** The reviewer pointed out that the data efficiency of DAC over AHP is not directly supported by experimental results. We will clarify that this data efficiency is the major difference between DAC and AHP and comes from the formalization directly (intra-option vs. SMDP). If the performance of DAC and AHP are different, this should be the major cause. We shall examine other possible minor implementation differences but they are unlikely to contribute to the performance difference.

Reviewer 3: **(Master Policy)** We agree that our π^H is similar to Bacon's mixture distribution (Sec 3.5, Bacon 2018), which is also used in IOPG. Bacon discussed two mechanisms for sampling from the mixture distribution: a two-step sampling method and a one-step sampling method. The latter can be viewed as an expected version of the former. The two-step one is implemented by the call-and-return model and is explicitly modelled in AHP (Levy and Shimkin, 2011) via introducing an extra variable. This new variable is not used in either Bacon's thesis or our work. Bacon's thesis mentions that the one-step modelling can lead to reduced variance compared to the two-step one. However, there is another significant difference: the one-step modelling is more data efficient than the two-step one. The two-step one (e.g., AHP) yields SMDP learning, where the policy gradient of the master policy is non-zero only when an option terminates. (This is a side effect of the extra variable in AHP.) The one-step one (e.g., our approach) yields intra-option learning, where the policy gradient is non-zero every step. This difference is not recognized in Bacon's thesis and we are the first to establish it, both conceptually and experimentally. **(Policy Gradients)** We agree that the gradient of the master policy appears in Bacon's thesis, which, however, is mixed with all other gradients. Unless we work on the augmented MDP directly, we cannot easily drop in other policy optimization techniques, which is our second contribution and is not done in Bacon's thesis. Furthermore, that policy gradient is never used in Bacon's thesis. All the experiments use Q-Learning for the master policy. We are happy to rephrase our claim about the policy gradient and more explicitly acknowledge Bacon's thesis. Doing so does not reduce our main contribution, a framework where 1) we can drop in all advanced policy optimization techniques 2) we can maintain intra-option learning for both the master policy and options. Bacon's thesis achieves 2) not 1); AHP achieves 1) not 2). We have achieved both. Although our underlying chain is the same as Bacon's thesis after algebraic manipulation, our new formulation brings in new insights for combining the option framework and other advanced policy optimization techniques in a data efficient way.

Reviewer 4: **(1)** We will reword our claim to make it less ambiguous, as in our rebuttal Reviewer 2(Performance). **(2)** It is not used. We shall remove it. **(3)** We agree. We shall change it accordingly. **(4)** They are defined in L52-56, e.g., S_1 is the state at step 1. **(5)** AHP tries to model the call-and-return sampling strategy explicitly in one single augmented MDP, which is the reason that the triple (O, B, A) is used as a new action. **(6)** At the starting state S_0, we do not have a previous option (the initial option O_0 is the current option). However, π^H depends on both the current state and the previous option (if it exists). To make a consistent expression for π^H, we introduced this placeholder # and O_{-1}. **(7)** As indicated in the caption of Table 1, we say an algorithm is compatible if it is possible to drop in other policy optimization techniques (instead of vanilla policy gradient) directly. **(8)** If an MDP is non-stationary, old transitions may belong to a different transition kernel. Off-policy methods, however, usually require a fixed transition kernel. **(9)** There is a tendency for an agent to use a single option within one task as learning other options may involve overhead. This could be the reason why termination goes closely to one at the end of the first task. When the task changes, the previously used single option is likely to be not so useful. So the agent has the motivation to build different new options and use them. It can achieve this by selecting the same option every step.