Thank you for the thoughtful and careful reviews. We hope the AC nominates some of you for reviewer awards. We start with common concerns and then respond to individual reviewer comments as space permits:

- **Common:** *There should be a baseline using MCTS and assuming access to simulator / common random numbers.* There appears to be some imprecision in reviews about what this means. DirPG is not a planning algorithm, because it does not use search (or a simulator) at test time, so standard MCTS / UCT at test time isn't applicable (L318-319).

  Nevertheless, we designed an algorithm that uses MCTS-style tree search and a simulator to learn a policy similarly to DirPG: A policy is initialized randomly. The algorithm explores k trajectories using UCT tree search with common environment random numbers. After the k trajectories, one best trajectory is extracted by running without the exploration bonus, and that trajectory is "distilled" into the policy by performing a gradient update to increase its probability. Then environment stochasticity is re-sampled and the algorithm repeats.

  This uses the same interface as DirPG, can be made to use the same number of environment interactions and gradient updates via choosing k, and it makes the same assumptions about simulator and common random number availability. We can make a key theoretical distinction, which is that this algorithm will behave asymptotically like DirPG as the ϵ hyperparameter goes to ∞, and thus it will be very risk-seeking. To see this, note that given sufficiently large k, UCT will converge to the trajectory with maximal return under the fixed realization of noise shared across the trajectories, which is the same trajectory that $\boldsymbol{a}_{dir}$ will converge to in DirPG given sufficiently large search budget and $\epsilon \to \infty$. Thus, DirPG offers a degree of control via ϵ that isn't available to this MCTS-style counterpart.
- **R2, R4:** *Surprised REINFORCE works well.* Ah, we should state more clearly that REINFORCE used the same number of environment interactions-per-seed as DirPG. It uses the same common random numbers access as DirPG.
- **R1, R2:** *"If a simulator is available, one typically wouldn't use a model-free algorithm like REINFORCE. ... If a heuristic and simulator are available (both typically assumed to be unavailable in RL), why wouldn't one use heuristic search methods?"* There are state-of-the-art RL approaches to real problems (e.g., [1]) where a simulator is available and researchers use REINFORCE. Heuristic search is not a silver bullet. The above work on solving combinatorial optimization problems using RL is based on the premise that there is room for improvement over traditional solvers. A potential future application of DirPG is to gain the benefits of learning in these domains without having to throw out the benefits of heuristic search. We expect this to become increasingly relevant as RL is applied to more non-traditional domains like combinatorial optimization, program synthesis, and sim-to-real.
- **R3** *"how much improvement can it bring compared with a perfect heuristic?"* With an uninformative heuristic, the search for $\boldsymbol{a}_{dir}$ reduces to sampling without replacement via Gumbel Top-K, which may bring some benefit over independent sampling when policies are peaked [2], but generally the heuristic is important.
- **R1** *"First, if it is only applicable to tabular cases"* It's not.
- **R1,R2** *"Does the method extend to function approximation?"* Yes. See architecture details in Appendix D.3.
- **R1** *"to solve combinatorial bandit problems, there are other algorithms such as CUCB"* We agree there are lots of algorithms for RL and combinatorial bandits. We say clearly in the paper that we are not claiming to be state-of-the-art (e.g. L95-96). Please also note that the specific algorithm suggested is very similar to our "full bandit" baseline. Compare Alg 1 in the CUCB paper to Appendix L614-617.
- **R3** *"Does DirPG also require the state to be finite?"* Continuous states are fine. Gumbels are applied to actions, not states. Continuous actions may also be possible in future work (A* sampling extends to Gumbel processes).
- **R3** *"gradient formula in l.66 and eq(5)"* Thank you! This is a typo. The LHS in l.66 should be expected reward of the argmax like in the leftmost term in eq 14.
- **R4** *"how does this compare to simple finite differences?"* We tried implementing [3] using similar common random numbers access as DirPG but couldn't get it to learn sensibly when given 5x the budget of interactions as DirPG. However, we didn't have time to try all the tricks in the paper (e.g., rank transformations).
- **R4** *"this assumption in general seems to be a bit understated in the presentation"* That wasn't our intention. We felt we said it clearly in L121-122 but understand your position and will increase its prominence.
- **R4** *"contrastive divergence"* Yes! Very interesting!

References

[1] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

[2] Wouter Kool, Herke van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. *arXiv preprint arXiv:1903.06059*, 2019.

[3] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.