We thank all reviewers for their thorough reading of our paper and for their detailed and insightful reviews. We are encouraged to read that they find our approach interesting and novel (**R1**, **R2**, **R3**, **R4**), and of particular interest to the NeurIPS community (**R3**, **R4**), that our value-function-based model is well-motivated and provides clear benefits (**R2**), that it is naturally suited to the problem at hand (**R3**) and based on compelling and principled ideas (**R4**). We are also glad to hear that this work is well-positioned in the literature (**R2**, **R4**), and that the ablation studies help readers understand the effect of key parameters (**R2**, **R4**). We answer specific questions below and will incorporate all feedback.

[**R1**, **R3**, **R4**] **Can you give more details on the method runtime?** At training time, our two main operations are evaluating many sample paths with the current policy, and re-training the value function estimator (at evaluation time, we just evaluate one sample path). With up to 16 hidden ReLUs, neural net training is fast, so the bottleneck is solving MIP (3). Table 2 shows average MIP runtimes given different architectures and solvers. The runtime of a policy iteration is $(\#$ sample paths$) \times (\#$ MIPs per path$) \times ($MIP runtime$)$. For $n = 21$ cities (16 hidden nodes), SCIP solves the average MIP in ~ 3s (Gurobi in ~ 0.4s), and we almost never exceed 10 MIPs per path, so computing 250 sample paths takes about 2h using SCIP (15min using Gurobi) [**R4**: we use Gurobi and SCIP for runtime evaluation but SCIP for all experiments due to licensing.] We can reduce runtime with parallelism: with as many machines as sample paths, the SCIP running time becomes about 30s (plus parallel pipeline overhead). For $n = 51$, SCIP is slower (~ 240s per MIP), and a policy iteration may take up to an hour in parallel. In contrast, Nazari et al.'s runtime bottleneck is neural net training (13h), but evaluation is much faster (seconds for them vs. minutes for us). A properly-tuned OR-Tools is faster than other approaches (solves in minutes). We note that actual runtime measurements are unreliable in our distributed setting due to a large variance in machines and load balancing overhead. We will discuss all this in the paper.

[**R1**, **R3**] **The empirical results are a bit less than exciting.** Our results in Table 1 outperform all RL baselines on small instances, but not for $n = 51$. We address this concern by (1) implementing a simple local search warm start for SCIP, enabling us to run more policy iterations and (2) giving a sharper statistical analysis of our performance with 95% confidence intervals. With these improvements, we obtain a score of 10.68 ± 0.09 on 51 cities, surpassing Nazari et al. (11.15) and nearly matching Kool et al.'s performance (10.62). We now describe (1) and (2) in detail. (1) The new warm start (1-opt over visited cities + TSP solver, will include in appendix) allows us to set a 60s SCIP time limit with little decrease in solution quality. We can now perform 80 policy iterations (as before, with 16 hidden ReLUs), improving our average objective from 11.36 to 10.92. We note the warm start would be unnecessary if using Gurobi. (2) Given iid graphs from the Nazari distribution X_i, we previously used the direct estimator $\frac{1}{50}\sum_{i=1}^{50} \text{RLCA}(X_i)$, which (with our warm start) gives an estimate of 10.92 ± 0.41, making comparisons with Nazari et al. and Kool et al. unclear (as **R4** suggested). By using the alternate estimator $\frac{1}{50}\sum_{i=1}^{50}(\text{RLCA}(X_i) - \text{ORTools}(X_i)) + \frac{1}{1000}\sum_{i=51}^{1050} \text{ORTools}(X_i)$, which is still unbiased but in practice has a lower variance, we obtain a confidence interval of 10.68 ± 0.09 (note the interval center also shifts down—our 50-instance sample leans towards higher-than-average OR-Tools objectives). To avoid this complexity in the final version, we will run over 1000 instances instead of 50.

[**R1**, **R2**, **R3**] **Why not try to generalize to unseen instances without relearning the value function? Can the framework be adjusted for other problems?** We're headed there, but not there yet. We could generalize to multi-instance CVRP or extend to more problems by enhancing the state representation, e.g., augmenting the state vector with the remaining number of vehicles to model a fixed fleet (**R1**). We felt that focusing on a single problem with a simple state space would help readers evaluate the key elements of our framework while making the paper easier to follow.

[**R1**, **R4**] **It would be a tough ask to reproduce your numbers from the information provided.** We recognize that our method requires nontrivial engineering work, especially when alternately training the neural network and optimizing over it. We will release our source code with the final paper and include in the appendix the detailed MIP formulation optimizing over the neural network (**R1**) and an algorithm block for our method (**R2**).

[**R1**, **R2**] **Fig 1: is 0 hidden nodes really better?** The same issue applies on Nazari51 instances, so we answer the question for this dataset instead, with three key points: (1) Without the local search heuristic we added above, the MIPs in the 16-neuron model would often time out with a large gap using SCIP. (2) We verified that by running for more policy iterations, 16 neurons eventually outperform 0 neurons. (3) Even with 0 hidden nodes, our model is nonlinear due to our combinatorial lower bounds. We will update Fig. 1 and Table 1 in the final paper to reflect these points.

[**R1**] **OR-Tools is terrible on the random instances (Table 1).** We're not sure what is meant here. In Table 1, OR-Tools outperforms all methods on every problem. The average OR-Tools objective is within the margin of error from the average optimal values reported by Nazari et al. (4.55 vs. 4.55 for $n = 11$ and 6.13 vs. 6.10 for $n = 21$), hence why we use OR-Tools to compute an "optimality gap" later on (though following **R4**, we will add the true optimum as a baseline where practical). Perhaps **R1** is referring to the bold numbers in Table 1, which intended to highlight the best RL-based results (not best overall). We will remove the bold formatting, as **R4** also found it unhelpful.

[**R4**] **Relation to LNS.** An RL framework in which actions are local improvement operators (e.g. k-opt) is an exciting alternative approach in bridging the gap between RL and CO. We will add a discussion, and thank you for the suggestion!