We thank the reviewers for the detailed feedback. R1 and R2 ask for a comparison to a prior method. R3's main comments are about the theoretical results. We've addressed all of this below.

(R1 & R2) Comparison to Kearns et al. in Appendix D.1: We provide a comparison to the prior method by Kearns et al. (2018) [3] in Appendix D.1, and will move it to the main text. We'll also expand the discussion we have about this method in the related work section. Kearns et al. consider an intersectional fairness problem similar to ours, but assume that the subgroups are defined by a class of functions with finite capacity. In contrast, we make no assumption on the given groups, but instead limit the flexibility of the Lagrange multiplier model we use to enforce constraints on them. We are thus able to handle a more general set of constraints, including those for ranking fairness.

More importantly, Kearns et al. assume access to an oracle to find the maximally-violated constraint at each step of their optimization. This assumption can be unrealistic, e.g. with the ranking fairness problems we consider, where we have one constraint for each incoming query, and the oracle would need to solve a maximization problem over *all* queries seen so far, or worse, over all realizable queries (this may be impractical without additional strong assumptions). Our optimization strategy is more practical and computes a simple gradient update on the Lagrangian model for each sampled query.

Table 4 in the appendix contains an empirical comparison to Kearns et al. on the intersectional fairness task (Sec 6.1). With a 3-layer multiplier model, we were able to achieve a similar test error and 95-th percentile constraint violation as their method, which we implemented with a brute-force search to find the max-violated constraint (Proposed: 0.22 [0.03], Kearns et al: 0.22 [0.04]). We aren't aware of a straight-forward way to implement Kearns et al. for the ranking fairness experiments.

(R1 & R3) Guarantees are on aggregated constraints: When there are a small number of constraints, it only makes sense that we should require that they all hold simultaneously. However, in the heavily-contrained setting that we consider, in which there can even be a potentially infinite number of constraints, expecting a model to satisfy all the constraints may not be reasonable, and therefore an alternate approach must be taken. This is the role of the Φ function: it tells us how we should measure the magnitudes of the constraint violations. The reviewers' main concern, we believe, is that there is a close correspondence between the function class chosen for the Lagrange multiplier model, and the (potentially unknown) Φ. It might be helpful to think of it this way: for every Lagrange multiplier model function class, there exist many possible choices of Φ (needed only by the theory, not the algorithm), each of which describes the sort of infeasibility and generalization guarantees that we can achieve. For some simple function classes, a convenient Φ is known (Table 1), while for more complicated function classes, it is not, but can still be reasoned about formally.

We also emphasize that we *do not* make a strong assumption about the constraints. While this allows us to handle general constraint sets, it restricts our ability to provide stronger theoretical guarantees.

(R3) Stochastic classifiers: In theory, stochastic classifiers are difficult to avoid for *non-convex* Lagrangian optimization, since the minimax theorem doesn't hold, so one must seek a *mixed* equilibrium (see e.g. ref. [11, 12] for prior use of stochastic classifiers for constrained problems). In practice, however, deterministically taking the last iterate usually (but not always) works, and if it doesn't, one can convert the stochastic classifier to a deterministic one (Cotter et al., 2019).[1] We'll add a brief discussion and citation of Cotter et al. to the paper.

(R2) Run-times: All experiments were run with TensorFlow. For the intersectional fairness experiment (Sec 6.1), our method took 31.7 mins with a linear multiplier model, and 33 mins with a 3-layer multiplier model. For the ranking fairness experiment on the larger MSLR-WEB10K data (Sec 6.3), our method took on average 6.8 hrs to train, the baseline constrained method took 6.2 hrs, and the method with no constraints took 4.3 hrs. We'll report these numbers, and also include the run-times for the experiments in Sec 6.2, along with details about the CPU configurations used.

(R3) Complexity of multiplier model: We expect that on a real-world problem, as one adds constraints, they will tend to become increasingly redundant. Our intuition is that the Lagrange multiplier model can *learn* these redundancies, and therefore cope with much larger constraint sets than one might expect based on its complexity alone. For example, in our robust fairness experiment in Sec 6.2, the number of constraints is exponential in the data size (one for each grouping of the data), but the multiplier model we used needed far fewer parameters.

(R3) Title: Thank you for the suggestion—we'll definitely consider other titles.

[1] A. Cotter, H. Narasimhan, and M. Gupta. On making stochastic classifiers deterministic. In *NeurIPS*, 2019.