Response to Reviewer 1 Thanks for all the remarks and the useful suggestions. We'll address all of them in the final version of the paper.

- **Re "The number of constraints is bounded by $1 + |I_1||\Sigma_2| + |\Sigma_1||I_2|$?"** Yes, exactly. We will add a remark about this after we define Ξ in Definition 3.
- **Re "Scalability of the Very Recent Subgradient Technique"** Good point. We originally chose to not give much space to the technique of Farina et al., since it does not guarantee feasible iterates and therefore the comparison with our algorithm and the linear programming approach is not apples-to-apples. You can see the performance of the subgradient technique for the Small instance in Figure 1; the plot does not account for the computation of the sparse factorization needed by the algorithm, which took ≈500ms. The algorithm was run on the same machine and under the same conditions described in the experimental section of our paper. The stepsize used was 0.001. We will add a few sentences to the final version and include this plot in the appendix.
- **Re "Is there some explanation for how the max deviation drops so much on the "ours" line, when the text is suggesting it is doing a $t \cdot x^t$ weighted average?"** Good question. We tried to look into it after we ran the experiments, but we don't have any definite answer. Our best guess is that the polytope of EFCE has interior, and that the gap drops suddenly because the average of the iterates entered the polytope of EFCE. The choice of using linear averaging was popularized by the original CFR$^+$ work, and is now standard in the computational game theory literature.

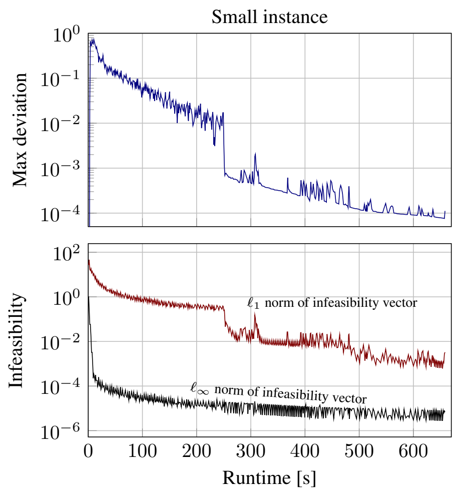

Figure 1: Performance of the subgradient technique of Farina et al. (2019c). The infeasibility vector of an iterate $\boldsymbol{\xi}$ is defined as the vector of absolute differences between the left-hand and right-hand sides of all the constraints that define Ξ (Definition 3).

Response to Reviewer #2 Thanks for the nice and thorough review, and for catching that typo on line 232! We will add a reference to the suggested JAIR article.

- **Question 1** In the specific case of the sequence-form strategy polytope, the top-down (i.e., using scaled extension) and the bottom-up (i.e., using convex hulls and Cartesian products) constructions lead to the *same* regret minimizer. So, for that application, there is no difference between the two approaches. We did not state that in the reviewed version in order to keep the focus on EFCEs. (Based on your comment, we will state the equivalence for the sequence-form case in the camera ready.)
- **Question 2** Broadly speaking, the problem of the bottom-up approach is that it cannot capture the affine structure (Line 302 in the algorithm, lines 235-241 in the example).
- **Question 5** The plots currently don't include the time it took to construct/deconstruct the regret minimizers, as much as they don't include the time it took to construct/deconstruct the Gurobi LP objects. The difference in the plots from including those times would be nearly invisible. We will make sure to point this out in the final version, thanks!

Response to Reviewer #3 Thanks for your review. Because of the tight space constraints, in the paper we were forced to take the saddle-point formulation of the EFCE problem as a given, and only focus on the really novel part, that is show how to construct an efficient regret minimizer for the Ξ polytope. In order to make the paper more self-contained, we will add an appendix in which we restate the saddle-point formulation of EFCE, and give a description of the full algorithm with pseudocode.

- **Re "Why does the algorithm actually converge to EFCE?"** As noted in Section 2.2, the problem of computing an EFCE is a bilinear saddle-point problem (BSPP), that is a problem of the form $\min_{\boldsymbol{x}\in\mathcal{X}} \max_{\boldsymbol{y}\in\mathcal{Y}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y}$ where $\mathcal{X} = \Xi$ is the convex polytope defined in Definition 3, $\mathcal{Y}$ is a convex and compact set (more details in Farina et al. 2019c), and $\boldsymbol{A}$ is a real matrix. It is known that *any* BSPP can be solved using regret minimization (Section 2.4). In particular, given access to two regret minimizers that output decisions on Ξ and $\mathcal{Y}$, respectively, we can compute a saddle point by letting the two regret minimizers face each other. In this paper we construct an efficient regret minimizer that can output decisions on the set Ξ.
- **Re "What loss is optimized in each local regret minimizer?"** The loss vector that is received by the overall regret minimizer for Ξ is as in Farina et al. (2019c); as mentioned above, we will dedicate a new appendix to recalling the details of how that vector is computed. From there, the loss is split into smaller vectors that are input to each local regret minimizer according to the OBSERVELOSS procedure (Algorithm 1 in our paper).
- **Re "How is average strategy computed?"** The average $\boldsymbol{\xi}$ strategy is computed using *linear averaging* (Line 324), a standard technique in computational game theory literature. The linear averaging of n vectors $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_n$ is defined as the weighted average $(\sum_{t=1}^n t \cdot \boldsymbol{\xi}_t)/(\sum_{t=1}^n t) = 2(\sum_{t=1}^n t \cdot \boldsymbol{\xi}_t)/(n(n+1))$. We'll remind the reader of this definition in the final version.