We thank the reviewers for their thoughtful comments and positive feedback.

Reviewer 1. **Q1**: Thank you very much for raising this valid technical point. To satisfy assumption 1, we can introduce box constraints on the entries of $\mathbf{H}$ and $\mathbf{W}$. Our algorithm will still be applicable in the presence of the box constraints. In addition, our experiments show that if the box constraints are chosen large enough, they will not become active over the iterates of the algorithms and hence will not change the trajectory of the algorithm. We will add this discussion to the paper and share our code in the presence of constraints.

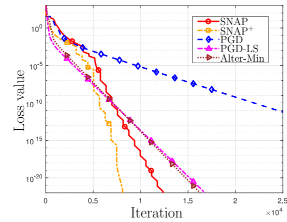

Figure 1: See Sec. E2.1 for the details of the this experiment ($c = 10^{-5}$).

Q2: As mentioned in [1] and [2], solving the subproblems is NP-hard in the number of constraints for linear constrained problems. Thus, even for a simple NMF problem, solving subproblems in [1, 2] requires exponential computational complexities in the number of constraints (which is the same as the number of variables in NMF).

Q3: There are different algorithms leveraging the NMF problem structure such as alternating minimization (Alter-Min). Here, we further compare Alter-Min with SNAP (shown in Figure 1). This plot shows that Alter-Min behaves similar to PGD-LS (but it costs less computational time numerically than PGD-LS due to its simplicity of the subproblems). We plan to add more numerical experiments and include large-scale cases in the revised version.

Reviewer 2. Regarding the computational cost of checking SOSP1, first notice that the number of times we check SOSP1 condition is reduced in our algorithm by introducing flag_α. When $\mathsf{flag}_\alpha = \emptyset$, we do not check SOSP1 condition for at least r_{th} iterations. Second, to reduce the computational cost of checking SOSP1, we proposed SGDN (a first-order method) for SNAP$^+$ with only $\mathcal{O}(d)$ per-iteration computational complexity (instead of $\mathcal{O}(d^2)$ required for other methods such as Lanczos). SGDN approximates the *smallest* eigenvalue in the interested subspace and avoids the evaluation of the restricted Hessian matrix. Our numerical experiments demonstrate that the gain obtained by SGDN procedure outweighs its computational cost. Is particular, in most cases, SNAP$^+$ outperforms classic projected gradient descent in terms of computational time.

Reviewer 3. **Q1**: Yes, we will add the comparison. **Q2**: The line search is a simple step-size selection method with only logarithmic time complexity. Also notice that our algorithm SNAP$^+$ does not require Hessian estimation and it only works with gradient. We will include more high level discussions to simplify the understanding of our algorithm. Thank you for pointing this out.

Q3: The SC condition is only used to establish the connection between SOSP1 and SOSP2. In other words, if the SC condition holds, then the obtained SOSP1 by our algorithms is also a SOSP2. However, our algorithms compute SOSP1 without needing SC condition to hold.

Q4: Showing the global optimality of SOSPs (in some practical problems) is a very interesting future research direction. However, even when they are not globally optimal, there is still value in computing SOSPs. This is because SOSPs satisfy stronger optimality conditions and are in general a subset of FOSPs. Hence they have higher chance of being globally optimal. Moreover, one can always run his/her own favorite algorithm for finding FOSPs and use our algorithm for escaping saddle points if needed. In addition, our *SOSP-finding* algorithms are "hessian-aware" and hence they can escape saddle points much faster as demonstrated in our numerical experiments. These benefits of algorithms developed for finding SOSPs are the main motivation behind many research works in the optimization society for finding SOSPs. Classical algorithms such as Newton, trust-region, or cubic regularization methods, were all (at least partially) motivated by these facts (long before researchers showing the global optimality of SOSPs for certain problem instances).

Q5: Thanks for pointing out this issue. We will revise accordingly. **Q6**: If $q_\pi(\mathbf{x}^{(r)})^T\mathbf{v}(\mathbf{x}^{(r)}) > 0$, then we only need to add a minus sign to $\mathbf{v}(\mathbf{x}^{(r)})$, otherwise, just keep it. We will revise to clarify this point further. **Q7**: Yes. The projection step is used in line 18 algorithm 1. **Q8**: Good suggestion! We will include a discussion on the LICQ and/or MFCQ in the revised version.

Q9: [Prop. 2] The definitions of the exact SOSP1 and SOSP2 have been used in the literature [35], but the relation between these two notations is unknown. Here, we provide rigorous proof to show their equivalence under the SC condition. [Prop. 3] This proposition shows that the introduced notion is continuous and hence proposed algorithms for computing this stationary notion can indeed escape (strict) saddle points asymptotically. Notice that as explained in [31, Remark 2.4] the continuity of the measure is necessary for escaping saddle points (where some previous works could get stuck in a strict saddle point)

Q10: i) It refers to Algorithm 1. We will clarify it in our revision. In the case the inactive set becomes empty and the first-order stationary condition has been satisfied, which implies that Algorithm 1 has already achieved the SOSP1 and therefore will stop. ii) No. Even when a constraint turns active, algorithm 1 is still possible to use PGD to update the iterates (when the conditions shown in line 3 of algorithm 1 are not satisfied), then the active constraint might be deactivated after the PGD step.