We thank all the reviewers for their careful reading and valuable feedback. Below, we provide our responses to individual comments.

Reviewer 1:

• *The experiments do not use the algorithm with theoretical guarantees, but a variant of it.*

We apologize for not being precise enough in our explanation. Namely, the implemented algorithm also has theoretical guarantees, but the amortized update time is $\tilde{O}(k)$ as opposed to $O(\text{poly}\log n)$ achieved by our main approach. Both the main approach and the implementation achieve a $(1/2 - \epsilon)$-approximation. In Appendix E, we explain why this modified algorithm achieves these guarantees, and also why we believe that this simplified approach has good empirical performance. Notice that the analysis of our implementation is a simple version of the analysis of our main algorithm. We will make this statement formal in the camera ready version and add the following theorem:

Theorem 1 *The implemented algorithm maintains a $(1/2-\epsilon)$-approximate solution after each operation. The amortized expected number of oracle queries per update of this algorithm is $\tilde{O}(k)$.*

• *This is problematic because the baselines are other algorithms with theoretical guarantees that might also have variants with better empirical performance.*

We did take the utmost care to optimize the implementations of the baselines as well. For example, the Sieve-Streaming implementation we have used only recomputes its sub-sieves lazily as needed, which gives it a large boost in performance.

Reviewer 2:

Thanks for raising this point, we will clarify those aspects in the final version. In Appendix C we explain the subroutine (algorithm Peeling) of [FMZ19] used in our paper. We also state that the subroutine is proposed in [FMZ19]. All the other necessary ingredients, e.g. bucketing or the way we perform lazy evaluation, are novel ideas developed in this work. The work [FMZ19] addresses the different setting of adaptivity complexity, which has very little in common with the dynamic setting. We will add a detailed comparison between [FMZ19] and our submission.

• *How does this relate to online algorithms?*

The two settings are related but they study different objectives. In designing online algorithms one focuses on building a stable solution on the fly. Here instead we design an algorithm to *efficiently* compute a good solution at any point in time.

• *n and OPT are used in Section 3 though they are not defined.*

We will make additional passes on Sections 3 and 4. OPT is defined in Preliminaries, but we will recall its definition in Section 3.

• *Is this work related to the approach used by [1]?*

We think that the main contribution of the paper is to carefully handle a fully dynamic stream with addition and deletions. In particular, to handle deletions we have to introduce new ideas so we think that the work is only vaguely related to [1].

• *It looks like there is no assumptions about a distribution on the insertions/deletions stream, but it could be interesting to look at that.*

Indeed, our guarantees hold for an arbitrary distribution of insertions and deletions. It is a very interesting question whether one can obtain stronger guarantees when operations follow certain distributions, e.g. arrive in a random order. Thank you for pointing this out!

Reviewer 3 and Reviewer 4:

We will make additional passes over Sections 3 and 4 and improve their clarity. In particular, we will adopt R4's suggestion :"*I think, presenting the high level idea behind each of the sets A,B, and S at the beginning of section 3 can make be helpful.*"