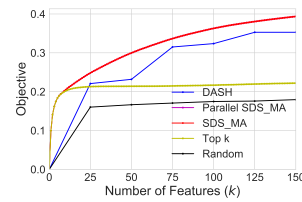
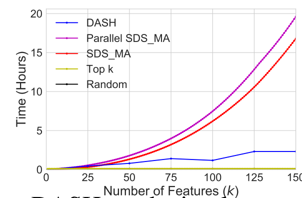
We thank all the reviewers for their comments and suggestions. Reviewer-specific comments to follow.

Reviewer 1. Thank you for your thoughtful review. To address your main concern regarding larger scale experiments, we ran experiments with $k = 150$ and $n = 5000$ which are larger than any of those in published works at NeurIPS and ICML on submodularity in the past two years, with the exception of one paper using $k = 50$ and $n = 10000$. We will discuss this and explain the fundamental differences between this work on parallelization and the MapReduce framework designed for distributed computing. We will appreciate if, in light of this response, you would consider revising your score.

- **Regarding larger scale experiments:** We ran all algorithms for $k = 150$ and $n = 5000$ and found the results consistent with those reported in the paper (see Figures). With the existing experimental setup described in this paper we can easily run DASH for $k > 1000$. Note that the bottleneck is that the benchmarks such as SDS_{MA} are too slow, which is the main advantage of using DASH.





- **Regarding sample complexity of line 5 in Algorithm 1:** Please see lines 534-539 in Appendix G. To obtain the guarantee with probability $1 - \delta$ one needs $m = n(\frac{OPT}{\epsilon})^2 \log(\frac{2n}{\delta})$ samples. As discussed in lines 254-257, this is a worst-case lower bound and, in practice, as few as 5 samples suffice. Similarly, the number of rounds needed in practice is much lower than the theoretical number (lines 259-260).
- **Regarding applicability to MapReduce:** Yes, DASH is applicable in the MapReduce setting. Algorithms in the MapReduce setting split the data across multiple machines and run Greedy on each machine. Every such MapReduce algorithm can run DASH instead of Greedy and enjoy a dramatic speedup. Referring to our discussion above, DASH can be implemented on much larger instances than those that have been used in previous work, including those in the MapReduce setting.
- **Regarding parallelization of SDS_{MA}:** In each round of SDS_{MA}, the algorithm computes the marginal contribution of each element to the solution set, which are parallelized. In lines 274-276, we state "When the calculation of the marginal contribution is computationally cheap, parallelization of SDS_{MA} has a longer running time...due to the cost of merging parallelized results." We will be happy to include more details in the full version.

Reviewer 2. Thank you for your review. The main concern is the lack of discussion about Theorem 6 being related to previous work [GD18] that appears on the arXiv. There is a slight technical difference between proof of Theorem 6 and that of [GD18]. More importantly, since this work is unpublished, we were not aware of this work at the time of writing the paper and we will be happy to cite it. Beyond this analysis, there are many technical and conceptual contributions in this paper that enable the exponential acceleration of statistical subset selection problems. We will appreciate if you would consider re-evaluating your score based on our response.

- **Regarding proof of Theorem 6:** Our proof of Theorem 6 first upper bounds the marginal contribution of a single element a unlike the proof in Lemma 5.4 in GD18, which bounds the marginal contribution of the set of A. The constants in the bounds also differ. Theorem 6 was introduced as an intermediate result to show that statistical subset selection objectives are differentially submodular, which allows for effective parallelization by DASH. The definition of differential submodularity and its application to parallelizable algorithms, which allow for both theoretical guarantees as well as empirical performance, are the core novelties of our paper and not discussed in [GD18].
- **Regarding [HS16]:** Please see line 84 for "relaxations of submodularity and relationship to differential submodularity in Appendix B" and line 439 "Horel et al. [HS16] define ϵ-approximately submodular functions...". Approximate submodularity defined in [HS16] is fundamentally different since the function is approximated pointwise by a submodular function, but not its marginals. Differential submodularity stipulates that the *marginals* of a function are approximated pointwise by submodular functions. This is a crucial difference: maximizing approximate submodular functions leads to intractable optimization problems (for any $\epsilon \in \Omega(1/k)$ maximizing an ϵ-approximate submodular function under a cardinality constraint requires exponentially-many queries to obtain a constant factor approximation).
- **Regarding relationship to relaxation of submodularity by Gupta et al. [GPB18]:** Differential submodularity generalizes the definition of Gupta et al. so that $g(A)$ is not equivalent to $h(A)$. This is necessary in cases where the objective function contains a diversity factor as in lines 180-181, 187-189. Showing that functions can be lower and upper bounded by two different functions is crucial here. We will include this in the discussion as well.

Reviewer 3. Thank you for your comments. We focus on objectives that are fundamental to statistical subset selection. We are working on extending this to dictionary selection and other applications. Regarding prior work, background on adaptive sampling can be found in lines 37-44 and relaxations of submodularity in lines 421-441. Our differential submodularity definition allows g and h to be different functions for added flexibility, which is necessary for objectives with diversity terms (lines 180-181). Regarding experimental design, we will be happy to include results for larger k to examine SDS_{MA} saturation. Regarding speedups, the bottleneck is the slowness of SDS_{MA}, which makes it difficult to compare speedups for large k. However, we would be able to run DASH, but not SDS_{MA}, for $k > 1000$.