We thank the reviewers for their time, insightful comments, and feedback. We provide a point-by-point response below.

Reviewer 1: We agree with Reviewer 1 and will clarify our main message: that Bandit-PAM can achieve the same loss as state-of-the-art but in much less time. We agree that wall-clock time comparisons are important for this message and, with that in mind, we ran new experiments comparing Bandit-PAM with FastPAM and FastPAM1, and will add a figure demonstrating wall-clock time scaling with dataset size to the final paper. For example, in the additional experiments, Bandit-PAM is 2.7x (3.4x) faster than FastPAM (FastPAM1, respectively) on a subset of MNIST of size $N = 65,000$ and 3.2x (4.0x) faster than FastPAM (FastPAM1, respectively) on the full dataset. We would like to clarify that we decided to focus on the number of distance evaluations because wall-clock time depends on implementation details and is not a perfect proxy for algorithmic complexity. We would also like to explain that FastPAM is a faster variant of FastPAM1 and produces a similar loss, but is not guaranteed to return the same output as PAM/FastPAM1/Bandit-PAM. Therefore, we think it is more appropriate to compare Bandit-PAM to FastPAM1 conceptually, but to FastPAM empirically for complexity. We will explain this and also add FastPAM1 results to all experimental results in the paper.

The reason why we stated that Bandit-PAM can be viewed as a batched version of the UCB algorithm is that it uses upper and lower confidence bounds to discard points. In the final paper, we will be more specific about the relationship to UCB and also mention the connection with the successive elimination algorithm by Even-Dar et al. Regarding the question about confidence intervals in the plots, they are the standard error of the empirical average over 10 repetitions, and are indeed small. Regarding Reviewer 1's other comments, we will include more details regarding the reduction from $\sum_i \Delta_i^{-2} \log n$ to $O(n \log n)$, clarify that Theorem 1 is a result of $(k + T)$ applications of Theorem 2, and make the $O(d)$ dependence explicit. We will also define σ_x before its first use and enlarge the legends of the plots.

Reviewer 2: We would like to clarify what we see as the novelty of our work. We emphasize that the general k-medoids problem is NP-hard, while 1-medoid is not, so the technique in Bagaria et al. 2018 cannot be directly applied when moving from 1-medoid to k-medoids. As such, we present a heuristic generalization of the approach in Bagaria et al. 2018 in which we track PAM's optimization path and show that this new problem can be efficiently solved as a sequence of bandit problems. This insight requires different objects to be formulated as bandit "arms" in the BUILD and SWAP steps, which we consider nontrivial. In contrast, the 1-medoid problem presented in Bagaria et al. 2018 is exactly solvable in $O(n^2)$ time, requires no heuristic solution, and the technique presented therein is a single bandit problem. Fundamentally, our work has achieved an $O(n^2)$ to $O(n \log n)$ reduction in a classical clustering problem. Therefore, we would respectfully argue that our paper is beyond a simple adaptation of prior work and our contribution is not small as suggested by Reviewer 2.

In addition to providing an efficient k-medoids algorithm, we provide an optimized C++ implementation alongside our paper. We anticipate this implementation will enable clustering large datasets with hard-to-compute distances using k-medoids, such as in the MOOC example presented in the paper. We think this is a valuable contribution to the applied ML community and can be used in various applications where k-medoids algorithms are currently used such as healthcare, education, operations research, etc.

Reviewer 2 observed that our theoretical result, as stated, would be vacuous if T is large. We would like to offer the following counterpoints. First, we would like to clarify that, as long as T is poly(n), one could easily modify the proof to get the same bound with slightly worse constants, by taking the hyperparameter δ also to be 1/poly(n). Second, we would also like to note that T has been empirically observed to be $O(k)$ (e.g., Figure 3a in Schubert et al. 2019); indeed, $T < 2k$ in all our experiments, including in additional experiments we ran for $k = 30$ and $k = 50$. Third, in additional experiments, we observed that Bandit-PAM is consistent with PAM in 599 out of 600 calls to Algorithm 1 (BUILD or SWAP steps). We will add a discussion about this to the final paper.

Following Reviewer 2's suggestion, we ran experiments on the total number of unique pairwise distances used. When $N = 1,000$, all 10^6 pairwise distances were used and, when $N = 70,000$, 8.7% of distances were used. We will add these statistics in the final paper and discuss intelligently caching distance computations in future work. We agree that [Aghaee 2016] is related to lowering the number of unique distance computations (and designing the cache); we will discuss it and cite it in our final paper.

Reviewer 5: Despite being from 1990, PAM is actually considered state-of-the-art in clustering quality (although not in runtime) according to Schubert et al. 2019. FastPAM produces a similar loss as PAM/FastPAM1/FastPAM2 and is the fastest, which makes it the focal point for our comparisons. Other algorithms, including CLARA, CLARANS, FastCLARA, and FastCLARANS produce empirically worse clustering results than PAM (see Figure 5 from Schubert et al. 2019 and Figure 1a in our work). trimed, from Newling et al. 2017, scales exponentially in the dimension d and hence is omitted from comparison, as Bandit-PAM scales as $O(d)$. We will clarify these in the final paper and cite Newling et al. 2017. In addition, we will also make the dependency on k explicit. Reviewer 5 also astutely observes that Theorems 1 and 2 can be written in terms of general $\delta < n^{-2}$, where $\delta = n^{-3}$ was chosen for convenience. We will include the result for general $\delta < n^{-2}$ in the paper.