Approximating the Median under the Ulam Metric[*]

Diptarka Chakraborty[†] Debarati Das[‡] Robert Krauthgamer[§]

Abstract

We study approximation algorithms for variants of the *median string* problem, which asks for a string that minimizes the sum of edit distances from a given set of m strings of length n. Only the straightforward 2-approximation is known for this NP-hard problem. This problem is motivated e.g. by computational biology, and belongs to the class of median problems (over different metric spaces), which are fundamental tasks in data analysis.

Our main result is for the Ulam metric, where all strings are permutations over $[n]$ and each edit operation moves a symbol (deletion plus insertion). We devise for this problem an algorithms that breaks the 2-approximation barrier, i.e., computes a $(2-\delta)$-approximate median permutation for some constant $\delta > 0$ in time $\tilde{O}(nm^2 + n^3)$. We further use these techniques to achieve a $(2-\delta)$ approximation for the median string problem in the special case where the median is restricted to length n and the optimal objective is large $\Omega(mn)$.

We also design an approximation algorithm for the following probabilistic model of the Ulam median: the input consists of m perturbations of an (unknown) permutation x, each generated by moving every symbol to a random position with probability (a parameter) $\epsilon > 0$. Our algorithm computes with high probability a $(1 + o(1/\epsilon))$-approximate median permutation in time $O(mn^2 + n^3)$.

1 Introduction

One of the most common aggregation tasks in data analysis is to find a representative for a given data set S, often formulated as an optimization problem. Perhaps the most popular version asks to minimize the sum of distances from all the data points in S (in a metric space relevant to the intended application). More formally, the goal is to find y in the metric space (not necessarily from S) that minimizes the objective function

$$\texttt{Obj}(S, y) := \sum_{x \in S} d(y, x),$$

and an optimal y is called a *median* (or a *geometric median*). For many applications, it suffices to find an *approximate median*, i.e., a point in the metric space whose objective value approximates the minimum (multiplicatively), see Section 2 for a formal definition. The problem of finding an (approximate) median has been studied extensively both in theory and applied domains, over various metric spaces. The most well-studied version is over a Euclidean space (called the Fermat-Weber problem), for which currently the best algorithm finds a $(1+\epsilon)$-approximate median (for any $\epsilon > 0$) in near-linear time [CLM⁺16] (see references therein for an overview). Other metric spaces that have been considered for the median problem include Hamming (folklore), the edit metric [San75, Kru83, NR03], rankings [DKNS01, ACN08], Jaccard distance [CKPV10], and many more [FVJ08, Min15, CCGB⁺17].

The median problem over the *edit metric* (where the edit distance between two strings is the minimum number of character insertion, deletion and substitution operations required to transform one string to the other) is called the *median string* problem [Koh85] (an equivalent formulation is known as *multiple sequence alignment* [Gus97]). It finds numerous applications in many domains, including computational biology [Gus97, Pev00], DNA storage system [GBC⁺13, RMR⁺17], speech recognition [Koh85], and classification [MJC00].

Given a set of m strings each of length n, the median string problem can be solved using standard dynamic programming [San75, Kru83] in time $O(2^m n^m)$, and it is known to be NP-hard [dlHC00, NR03] (even W[1]-hard [NR03]). There is a folklore algorithm that easily computes a 2-approximate (actually, $(2-\frac{1}{m+1})$-approximate) median — simply report the best input string, i.e., $y^* \in S$ that minimizes the objective (we call this algorithm BestFromInput, see Procedure 1) — and in fact this argument holds in every metric space. Although several heuristic algorithms exist [CA97, Kru99, FZ00, PB07, ARJ14, HK16, MAS19], no polynomial-time algorithm is known to break below 2-approximation (i.e., achieve factor $2-\delta$ for fixed $\delta > 0$) for the median string problem. In contrast, over the Hamming metric a median can be computed in linear time by simply taking a coordinate-wise plurality

[*] The full version is available at https://arxiv.org/abs/2011.00868

[†] National University of Singapore. Work partially supported by NUS ODPRT Grant, WBS No. R-252-000-A94-133. Email: diptarka@comp.nus.edu.sg

[‡] Basic Algorithm Research Copenhagen (BARC), University of Copenhagen. Work supported by Basic Algorithms Research Copenhagen (BARC), grant 16582 from the VILLUM Foundation. Email: debaratix710@gmail.com

[§] Weizmann Institute of Science. Work partially supported by ONR Award N00014-18-1-2364, the Israel Science Foundation grant #1086/18, and a Minerva Foundation grant. Email: robert.krauthgamer@weizmann.ac.il vote. One can also compute a $(1+\epsilon)$-approximation in sublinear time using sampling (similarly to [FOR17]).

We focus mostly on approximating the median over the *Ulam metric*, which is a close variant of the edit metric. The Ulam metric of dimension n is the metric space $(\mathcal{S}_n, d)$, where $\mathcal{S}_n$ is the set of all permutations over $[n]$ and $d(x, y)$ is the minimum number of character moves needed to transform x into y [AD99].[1] The importance of studying the Ulam metric is twofold. First, it is an interesting measure of dissimilarity between rankings, which arise in application domains like sports, databases, and statistics. Second, it captures some of the inherent difficulties of the edit metric, and thus, any progress in the Ulam metric may provide insights to tackle the more general edit metric. The Ulam metric has thus been studied from different algorithmic perspectives [CMS01, CK06, AK10, AN10, NSS17, BS19], but unfortunately, no polynomial-time algorithm is currently known to break below the folklore 2-approximation (actually, $(2-\frac{1}{m+1})$-approximation for m input permutations) bound for Ulam median. In contrast, for the median with respect to *Kendall's tau distance* over permutations, which is often used for rank aggregation [Kem59, You88, YL78, DKNS01], a PTAS [KMS07, Sch12] is known, improving upon a polynomial-time 4/3-approximation [ACN08].

Our main result is a deterministic polynomial-time algorithm that breaks below 2-approximation for Ulam median (see Section 3).

Theorem 1.1. *There is a constant $\delta > 0$ and a deterministic algorithm that, given as input a set of m permutations $S \subseteq \mathcal{S}_n$, computes a $(2-\delta)$-approximate median in time $O(nm^2 \log n + n^2 m + n^3)$.*

The running time's quadratic dependence on m comes from a naive subroutine to find the best median among the data set S. We can replace this subroutine with a randomized $(1+\epsilon)$-approximation algorithm, due to [Ind99], to obtain linear dependence on m.

Furthermore, one of our key algorithmic ingredients for the Ulam metric extends to the more general edit metric, albeit with some restrictions on the length of the median string and on the optimal objective value. Specifically, we refer to the following problem: Given a set of strings over Σ^n (for an alphabet Σ), find a string in Σ^n, called a *length-n edit-median*, that attains the minimum objective value under the edit metric. In fact, the improvement is achieved using the folklore algorithm mentioned earlier, i.e., in our restricted setting this algorithm actually beats 2-approximation! (See Section 3.3.)

Theorem 1.2. *Given a set of strings $S \subseteq \Sigma^n$ whose optimal median objective value is at least $|S|n/c$ for some $c > 1$, Procedure* BestFromInput *reports a $(2-\frac{1}{50c^2})$-approximate length-n edit-median in time $O(nm^2 \log n)$.*[2]

Restrictions on the median string's length and on the optimum objective value may be justified in certain applications. For example, in DNA storage system [GBC+13, RMR+17], stored data is retrieved using next-generation sequencing, and as a result several noisy copies of the stored data are generated. (Note, here the noises are in the form of insertions, deletions and substitutions.) Currently, researchers use median-finding heuristics to recover the stored data from these noisy copies. Since third-generation sequencing technology like single molecule real time sequencing (SMRT) [RCS13] involves $12-18\%$ errors, the optimum median objective value is quite large (and matches our restriction). Moreover, since the noise is randomly added at each location during sequencing, it follows from standard concentration inequalities that with high probability the lengths of the noisy strings are "close" to that of the originally stored data (or the median). Thus, a length-restricted median, as in our result, should be a good approximation of the original one.

Motivated by the above application we further investigate a probabilistic model for the Ulam metric, as follows. The input consists of m perturbations of an (unknown) permutation x, each generated by moving every symbol to a random position with probability (a parameter) $\epsilon > 0$. See Section 4 for a formal definition of this input distribution, which we denote by $S(x, \epsilon, m)$. We then provide a $(1+\delta)$-approximate median for this model.

Theorem 1.3. *Fix a parameter $\epsilon \in (0, 1/40)$, a permutation $x \in \mathcal{S}_n$, and $40 \leq m \leq n$. There is an $O(n^3)$-time deterministic algorithm that, given input S drawn from $S(x, \epsilon, m)$, outputs a $(1+\delta)$-approximate median of S, for $\delta = \frac{20}{m} + \frac{3}{\log(n/\epsilon)} + \frac{2e^{-m/40}}{\epsilon}$, with probability at least $1 - 5/m$.*

Our analysis is based on a novel encoding-decoding (information-theoretic) argument, which we hope could also be applied to the more general edit metric (left open for future work).

Even though the Ulam distance is a special case of the edit distance, for the problem of finding an exact

[1] One may also consider one deletion and one insertion operation instead of a character move, and define the distance accordingly [CMS01].

[2] We make no attempt to optimize the constants.

median over the Ulam metric, no algorithm faster than exhaustive search (over the $n!$ permutations) is known. In contrast, for the edit metric, at least with constantly many input strings, one can find an exact median in polynomial time using dynamic programming [San75, Kru83]. The lack of a polynomial-time algorithm for the Ulam metric, even with constantly many inputs, is perhaps not very surprising, because the related problem of rank aggregation, which is the same median problem over permutations (rankings) but with respect to Kendall's tau distance, is NP-hard even for $m = 4$ permutations [DKNS01]. And even for $m = 3$ permutations, the current polynomial-time algorithm achieves only 1.2-approximation [ACN08].

Nevertheless (and in contrast to rank aggregation), we show a polynomial-time algorithm that solves the Ulam median problem for $m = 3$ permutations (see the full version). We further extend this result to show that for m inputs there is an $O(2^{m+1}n^{m+1})$-time algorithm computing a 1.5-approximate median. We refer interested readers to the full version for the details.

Remark 1.4. *Some literature slightly extend the notion of a permutation, and call a string $x \in \Sigma^n$ a permutation if it consists of distinct characters [CMS01, Cor03]. Then the Ulam metric of dimension n is defined over all these permutations, and distances are according to the standard edit distance. All our results hold also for this variant of the Ulam metric as long as the goal is to find (as median) a permutation of length n. However, for the sake of simplicity we present our results only for the standard definitions of a permutation and the Ulam metric (as in [AD99]).*

1.1 Technical Overview

Breaking below 2-approximation (in worst-case) We start with an overview of our main result, a $(2 - \delta)$-approximation for the median under Ulam (Theorem 1.1). It is instructive to understand if and when does the well-known 2-approximation algorithm fail to achieve approximation better than 2. Recall that this algorithm reports the best input permutation $y \in S$ (see BestFromInput in Procedure 1). To analyze it, let x_{med} be an optimal median for S, and let $y^* \in S$ be an input permutation that is closest to x_{med}, i.e., $d(y^*, x_{\text{med}}) = \min_{x \in S} d(x, x_{\text{med}})$. Then by the triangle inequality

$$(1.1) \qquad \sum_{x \in S} d(y^*, x) \leq \sum_{x \in S} [d(y^*, x_{\text{med}}) + d(x_{\text{med}}, x)] \leq 2 \sum_{x \in S} d(x_{\text{med}}, x),$$

and the objective value of the reported $y \in S$ is only better.

Now consider a scenario where this analysis is tight; suppose every input permutation is at the same distance $\ell > 0$ from x_{med}, and the distance between every two input permutations is 2ℓ. Then the objective value for x_{med} is ℓm, but for every input permutation $y \in S$ it is $2\ell m$. In this scenario, a better approximation must exploit the structure of the input permutations.

Somewhat surprisingly, we show that if the optimal objective value is large, say $\Omega(mn)$, then the above scenario cannot occur. To gain intuition, start with a favorable case where all input permutations are at distance at least n/c from x_{med} (for some constant $c > 1$). Then in an optimal alignment of an input $x \in S$ with x_{med}, at least $\ell = n/c$ symbols are not aligned (i.e., are moved). Now a combinatorial bound (based on the inclusion–exclusion principle), implies that every $2c$ input permutations must include a pair $x', x'' \in S$ whose sets of non-aligned symbols (with x_{med}) have a large intersection, specifically of size $\Omega(n/c^2)$. This yields a non-trivial distance bound $d(x', x'') \leq 2n/c - \Omega(n/c^2) < 2\ell$ that contradicts our assumption. Our full argument (in Section 3.1), employs this idea more generally than just the favorable case (i.e., whenever the optimal objective value is large). We use additional steps, like averaging arguments to exclude permutations that are too close or too far from x_{med}, and an iterative "clustering" of the input permutations around at most $2c$ so-called candidate permutations, to infer that for at least one candidate $y^* \in S$, the cluster around it is large. We use this to bound the objective value for this candidate y^*, but with a gain compared to (1.1), due to the cluster around y^* that has many permutations, all within distance $2n/c - \Omega(n/c^2)$ from y. We conclude that reporting the best permutation among the input S is at least as good as y^* and thus breaks below 2-approximation. This argument about large optimal objective value extends to the edit metric, i.e., over general strings (see Section 3.3).

The general case of the Ulam metric (without assuming that the optimal objective value is large) is more difficult and involved, but perhaps surprisingly, reuses the main technical idea from above, although not in a black-box manner. We split this analysis into two cases by considering the contribution of each symbol to the optimal objective value. To be more precise, fix an optimal alignment of each input permutation $x \in S$ with x_{med}, and let the *cost* of a symbol count in how many alignments (equivalently, for how many $x \in S$) this symbol is not aligned.

Informally, one case (called Case 2 in Section 3.2) is when the cost is distributed over a few symbols. Here, by restricting these optimal alignments to these costly symbols we can employ a strategy similar to our first case above (optimal objective value is large). Intuitively, for the costly symbols we obtain approximation better than 2, and approximation factor 2 for the other symbols, and altogether conclude that reporting the best permutation among the input S breaks below 2-approximation. While this plan is quite intuitive, combining these two analyses into one argument is technically challenging because we cannot really analyze these two sets of symbols (denoted therein G and $\overline{G}$) separately. We defer the detailed proof of Case 2 (i.e., Lemma 3.13) to the full version.

In the remaining case (called Case 1 in Section 3.2), the cost is distributed over many symbols; this is a completely different situation and we devise for it an interesting new algorithm (RelativeOrder in Procedure 2). The main idea is that now most of the symbols in $[n]$ must be aligned in many optimal alignments (i.e., for many $x \in S$), and thus for every two such symbols, their relative order in x_{med} can be easily deduced from the input (by taking majority over all $x \in S$). More precisely, call a symbol *good* if it is aligned in at least 0.9-fraction of the input permutations. Observe that every two good symbols must be aligned simultaneously in at least 0.8-fraction of the input permutations, hence their relative order in x_{med} can be computed by checking their order in each $x \in S$ and taking a majority vote. This observation is very useful because in this case most symbols are good, however the challenge is that we cannot identify the good symbols reliably. Instead, our algorithm finds all pairs of symbols with a qualified majority (say, above 0.8 threshold); which is a superset of the aforementioned pairs, and might contains spurious pairs (involving bad symbols) that contradict the relative order between good symbols. We overcome this by iteratively removing symbols that participate in a contradiction: our algorithm builds a directed graph H, whose vertices represent symbols and whose edges represent qualified majority, and then iteratively removes a (shortest) cycle, where removal of a cycle means removing all its vertices (not only edges). We prove that every such cycle consists mostly of bad vertices/symbols, and straightforward counting shows that the final graph $\overline{H}$ contains almost all the good symbols. Moreover, this final $\overline{H}$ contains no cycles, and thus topological sort retrieves the order (according to x_{med}) of almost all good symbols. We then obtain a permutation that is pretty close to x_{med} by simply adding all the missing symbols at the end. We point out that the approximation factor that we get here (Case 1) can in principle be close to 1 (it depends on some parameters). We indeed exploit this in our algorithm for the probabilistic model (as discussed next), however the balance with Case 2 is quite poor, and thus our overall approximation factor is quite close to 2.

Finding median in a probabilistic model. Our next result (in Section 4) deals with a probabilistic model over the Ulam metric, and is motivated by the application to DNA storage system. In this model, the input S consists of m permutations, each generated from an unknown permutation x by moving each symbol independently with probability $\epsilon > 0$ to a randomly chosen location. Let $S(x, \epsilon, m)$ denote the distribution generated in this model. Given an input S drawn from this distribution, the objective is to find a median of S (not the unknown x). We show a polynomial-time algorithm that finds (with high probability) a $(1 + o(1/\epsilon))$-approximate median of S (see Theorem 1.3 for the precise factor). Our argument consists of two parts. First, we show that the unknown x is itself a $(1 + o(1))$-approximate median. Second, we provide an algorithm that computes a permutation $\tilde{x}$ that is "very close" to the unknown x. It then follows by the triangle inequality that $\tilde{x}$ is an approximate median of S.

The first part goes via an information-theoretic (encoding-decoding based) argument. We show that if x is not an approximate median of S, then we can encode the set of (random) move operations used to generate S, using fewer number of bits than that required by the information-theoretic bound. It is evident from the generation process of each permutation $x_i \in S$, that the (Shannon) entropy of this set of random move operations has total entropy about $\sum_{x_i \in S} d(x, x_i)$, which is the median objective value for x. Let x_{med} be an optimal median of S, and denote the optimal median objective value by $\texttt{OPT}(S) = \sum_{x_i \in S} d(x_i, x_{\text{med}})$. To encode all the x_i's (given x), one can first specify a set of move operations to transform x into x_{med}, and then specify the move operations to transform x_{med} to each x_i. The length of this encoding is about $d(x, x_{\text{med}}) + \sum_{x_i \in S} d(x_i, x_{\text{med}}) = d(x, x_{\text{med}}) + \texttt{OPT}(S)$. If the above encoding could be used to recover all the random move operations, then we could conclude, by Shannon's source coding theorem, that the objective value with respect to x is almost equal to $\texttt{OPT}(S)$, and thus x is an approximate median. We do not know if this encoding is indeed sufficient for the said decoding, but we can add to it a little extra information, that suffices to decode the set of random operations; let us elaborate how works.

From the above encoding we know all the x_i's. We show that almost none of the symbols (except about $O(\log n)$ many) that were moved from x to generate x_i, appears in every *longest common subsequence* (lcs) between x_i and x. Therefore by computing an lcs between x_i and x, all but $O(\log n)$ moved symbols can be identified. Note that a random move operation consists of two pieces of information, the moved symbol and the location where it is moved. From the lcs we get back the information about the moved symbols. So the only task remains is to identify the locations where they are moved. Suppose a symbol a is moved to a location right next to another symbol b to generate x_i from x. (Note, since we are dealing with permutations we can identify a location by its preceding symbol). Observe that the symbol b might also be moved, but only with probability ϵ, and so with the remaining probability b just precedes a in x_i. Therefore for each of the moved symbols (except for about an ϵ-fraction) for x_i, just by looking into its preceding symbol in x_i we can identify its moved location. For the remaining ϵ-faction we could encode the moved locations explicitly, but that would worsen the approximation factor. To handle this, we argue that for each of these ϵ-fraction of moved symbols we can identify a $O(\log n)$-sized "set of candidate locations", and thus it suffices to encode the exact location only inside this candidate set using $O(\log \log n)$ bits. Now after a careful calculation we get that the whole encoding is of length $(1 + o(1))\texttt{OPT}(S)$. Then we apply Shannon's source coding theorem and conclude that the objective value with respect to x (which is equal to the total entropy of random move operations) is at most $(1 + o(1))\texttt{OPT}(S)$, and so x is a $(1 + o(1))$-approximate median of S. We defer the exact details of the encoding-decoding argument (i.e., the proof of Theorem 4.3) to the full version.

The second step is to reconstruct the initial unknown permutation x. The task is similar to that in [CDKL14], although their underlying distance is Kendall's tau distance, and their random perturbation model is different. In our case, each symbol of x is moved with probability ϵ to generate a permutation x_i. Hence any particular symbol is moved in expectation in ϵm many x_i's. Further, the total objective value is equally distributed on all the symbols. This scenario is similar to Case 1 in our worst-case approximation algorithm, except that now the underlying permutation is x instead of x_{med}. Thus we can use Procedure RelativeOrder discussed above, and since the objective value is distributed equally among all the symbols, we can find a permutation $\tilde{x}$ that is very close to the unknown x. Moreover, when $m \geq \Omega(\log n)$ we show that for every two symbols $a \neq b \in [n]$ we can decide (with high probability) whether a appears before b in x or not, by observing their relative order in the input permutations. Hence using any sorting algorithm (with slight modification) we can reconstruct x with high probability.

Exact median for three permutations. We devise an algorithm that finds an exact median for three permutations. The non-trivial part of this algorithm is that running the conventional dynamic program for a median [San75, Kru83] will compute a string, which need not be a permutation, and in fact even its length need not be equal to n. Therefore, we first use a slight modification of that dynamic program to compute a string x' of length exactly n (but not necessarily a permutation) with the minimum possible median objective value with respect to edit distance (i.e., $x' = \arg\min_{y \in [n]^n} \sum_{i \in [3]} \Delta(y, x_i)$). Crucially, the objective value attained by this x' is at most that of a median permutation x_{med}. Next, we post-process x' to produce a permutation $\tilde{x}$ over $[n]$, by removing multiple occurrences of any symbol and then inserting all the missing symbols (in a careful manner). To complete the analysis we show that $\sum_{i \in [3]} \Delta(\tilde{x}, x_i) = \sum_{i \in [3]} \Delta(x', x_i)$. Interested readers may refer to the full version for the details.

1.2 Conclusion

There is a folklore algorithm that computes 2-approximate median in any metric space, however no better approximation algorithm was known for the Ulam and edit metrics, despite their utter importance. Our main result breaks below 2-approximation for the Ulam metric. Further, we extend our result to the more general edit metric, albeit with certain restrictions on the length of the median and on the optimal median objective value. An exciting future direction, is to beat 2-approximation without these restrictions. In fact, this was recently stated as an open problem [Coh19].

We also consider for the median Ulam problem a probabilistic model, which is motivated by the applications to DNA storage system, and we provide for it a $(1 + o(1))$-approximation algorithm. In achieving our result, we use novel encoding-decoding (information-theoretic) argument, which we hope could also be used for the edit metric (left open for the future work) and perhaps even more general metric spaces.

2 Preliminaries

Notations: Let $[n]$ denote the set $\{1, 2, \cdots, n\}$. We refer the set of all permutations over $[n]$ by $\mathcal{S}_n$. Throughout this paper we consider any permutation x as a sequence of numbers $a_1, a_2, \cdots, a_n$ such that $x(i) = a_i$. For any subset $I \subseteq [n]$, let $x(I) := \{x(i) | i \in I\}$.

The Ulam Metric and the Problem of Finding Median. Given two permutations $x, y \in \mathcal{S}_n$, the *Ulam distance* between them, denoted by $d(x, y)$, is the minimum number of character move operations that is needed to transform x into y.

Given two strings (permutations) x and y of lengths n_1 and n_2 respectively, *alignment* g is a function from

$[n_1]$ to $[n_2] \cup \{*\}$ which satisfies:

- $\forall i \in [n_1]$, if $g(i) \neq *$, then $x(i) = y(g(i))$;
- For any two $i \neq j \in [n_1]$, such that $g(i) \neq *$ and $g(j) \neq *$, if $i > j$, then $g(i) > g(j)$.

For an alignment g between two strings (permutations) x and y, we say g *aligns* a character $x(i)$ with some character $y(j)$ iff $j = g(i)$.

Given a set $S \subseteq \mathcal{S}_n$ and another permutation $y \in \mathcal{S}_n$, we refer the quantity $\sum_{x \in S} d(y, x)$ by the *median objective value* of S with respect to y, denoted by $\texttt{Obj}(S, y)$.

Given a set $S \subseteq \mathcal{S}_n$, a *median* of S is a permutation $x_{\text{med}} \in \mathcal{S}_n$ (not necessarily from S) such that $\texttt{Obj}(S, x_{\text{med}})$ is minimized, i.e., $x_{\text{med}} = \arg\min_{y \in \mathcal{S}_n} \texttt{Obj}(S, y)$. We refer $\texttt{Obj}(S, x_{\text{med}})$ by $\texttt{OPT}(S)$. We call a permutation $\tilde{x}$ a *c-approximate median*, for some $c > 0$, of S iff $\texttt{Obj}(S, \tilde{x}) \leq \texttt{OPT}(S) \leq c \cdot \texttt{Obj}(S, \tilde{x})$.

A Folklore 2-approximation Algorithm. For the problem of finding median (over any metric space, and so for the Ulam), there is a folklore 2-approximation algorithm (actually, a $(2 - \frac{1}{m+1})$-approximation algorithm for m input permutations). We briefly present here this algorithm for a set of permutations. We also refer to this algorithm as Procedure BestFromInput (Procedure 1).

Procedure 1 BestFromInput (S)

```
Input: $S \subseteq \mathcal{S}_n$.
Output: A permutation $y \in S$.
For all pairs of permutations $x_i, x_j \in S$, compute $d(x_i, x_j)$.
return $\arg\min_{y \in S} \sum_{x \in S} d(y, x)$.
```

3 Breaking below 2-approximation (in Worst-case)

In this section, we describe a polynomial-time algorithm that computes, for any given input permutations, a $(2 - \delta)$-approximate median under the Ulam metric. Below we restate Theorem 1.1.

Theorem 3.1. *There is a constant $\delta > 0$ and a deterministic algorithm that, given as input a set of m permutations $S \subseteq \mathcal{S}_n$, computes a $(2 - \delta)$-approximate median in time $O(nm^2 \log n + n^2 m + n^3)$.*

We start with the description of our algorithm and the running time bound. Next we analyze the approximation factor into two parts. First we consider a special case where the objective is large and give a stronger approximation guarantee. After that we discuss the general case. Given as input a set of permutations $S \subset \mathcal{S}_n$, our algorithm runs two procedures, each producing a permutation (candidate median), and returns the better of the two (that has smaller objective value). The first procedure is BestFromInput (see Procedure 1), which reports an input permutation $y \in S$ that has the minimum objective value among all input permutations, i.e., $\arg\min_{y \in S} \sum_{x \in S} d(y, x)$. (We have discussed in previous section that this algorithm is well-known to achieve 2-approximation in every metric space.)

The second procedure, called RelativeOrder, is given a parameter $0 \leq \alpha \leq 1/10$, and works as follows (see also Procedure 2). First, create a directed graph H with vertex set $V(H) = [n]$ and edge set

$$E(H) = \{(i, j) : i \text{ appears before } j \text{ in at least } (1 - 2\alpha)|S| \text{ permutations in } S\}.$$

Next, as long as the current graph H is not acyclic, repeatedly find in it a cycle of minimum length and delete all its vertices (with all their incident edges). Denote the resulting acyclic graph by $\overline{H}$, and use topological sort to compute an ordering $\mathcal{P}$ of its vertex set $V(\overline{H}) \subset V(H) = [n]$. We shall write $i \triangleleft j$ to denote that i precedes j in this ordering $\mathcal{P}$. Let the string $\overline{x}$ be a permutation of (set of symbols) $V(\overline{H})$ by ordering them according to $\mathcal{P}$. Finally, output the permutation $\tilde{x}$ of $[n]$ that is obtained by appending to $\overline{x}$ all the remaining symbols $[n] \setminus V(\overline{H})$ in an arbitrary order.

Procedure 2 RelativeOrder (S, α)

```
Input: $S \subseteq \mathcal{S}_n$ of size $m$, $0 < \alpha \leq 1/10$.
Output: A permutation string $\tilde{x}$ over $[n]$.
$H \leftarrow ([n], E)$ where $E = \{(i, j) : i$ appears before $j$ in $\geq (1 - 2\alpha)|S|$ permutations in $S\}$
while $H$ contains a cycle do
    $\mathcal{C}_{\min} \leftarrow$ cycle of minimum length in $H$
    $H = H - V(\mathcal{C}_{\min})$
end while
$\overline{H} \leftarrow H$
$\overline{x} \leftarrow$ string formed by topological ordering of $\overline{H}$
$\tilde{x} \leftarrow$ string formed by appending to $\overline{x}$ the symbols $[n] \setminus V(\overline{H})$ in an arbitrary order.
return $\tilde{x}$.
```

Running time analysis. Let $m = |S|$. Since $d(x, y)$ can be computed in $O(n \log n)$ time for any $x, y \in \mathcal{S}_n$, Procedure BestFromInput runs in time $O(nm^2 \log n)$.

In Procedure RelativeOrder, given the set S and parameter α we can compute graph H in time $O(n^2 m)$.

Next, we iteratively find a minimum-length cycle $\mathcal{C}_{\min}$ in the current graph $\tilde{H}$ in time $O(n^3)$ (using an All-Pairs Shortest Path algorithm), and delete all the vertices of $\mathcal{C}_{\min}$ (and edges incident on these vertices) in time $O(n^2)$. Hence, each iteration takes time $O(n^3)$, and since the number of iterations required is at most n, the total time to compute $\overline{H}$ is $O(n^4)$. Now since $\overline{H}$ has at most n vertices, computing a topological order of its vertices runs in time $O(n^2)$. Given this ordering, the strings $\overline{x}$ and $\tilde{x}$ are computed in time $O(n)$. Thus, Procedure RelativeOrder runs in time $O(n^2m+n^4)$.

As our main algorithm outputs the string with the minimum median objective value among the two strings returned by Procedure BestFromInput and Procedure RelativeOrder, its total running time is $O(nm^2 \log n + n^2m + n^4)$. In Remark 3.12, we will comment on how to improve the $O(n^4)$ factor of the above time-bound to $O(n^3)$ by slightly modifying Procedure RelativeOrder.

We devote the remaining part of the section to derive the approximation ratio of our algorithm. We will first consider a special case when $\mathtt{OPT}(S)$ is "large", for which the analysis is slightly simpler, and also, we get a stronger approximation guarantee. Then we will turn our attention to the more general case. Although the result for the high regime is independent of that for the general case, one of the main ideas carries forward to the general case, albeit with more complications.

3.1 High regime of the optimal objective value

Lemma 3.2. *Given a set of permutations $S \subseteq \mathcal{S}_n$ with $\mathtt{OPT}(S) \geq |S|n/c$ for some $c > 1$, Procedure* BestFromInput *(S) outputs a $(2 - \frac{1}{50c^2})$-approximate median.*

Proof. We first introduce some notation. Let $m = |S|$ and set $\delta = \frac{1}{50c^2}$. Let x_{med} be an (optimal) median of S; then $\mathtt{OPT}(S) = \sum_{x \in S} d(x, x_{\text{med}})$, and for brevity we denote it by $\mathtt{OPT}$. For any subset $S' \subseteq S$, denote $\mathtt{OPT}_{S'} = \sum_{x \in S'} d(x, x_{\text{med}})$. We assume henceforth that

$$\forall x \in S, \qquad d(x, x_{\text{med}}) > (1-\delta)\mathtt{OPT}/m, \tag{3.2}$$

because any $x' \in S$ that violates (3.2) is a $(2-\delta)$-approximate median of S by the triangle inequality, formally $\sum_{z \in S} d(x', z) \leq \sum_{z \in S}[d(x', x_{\text{med}}) + d(x_{\text{med}}, z)] \leq (2-\delta)\mathtt{OPT}$.

For each $x \in S$ fix an optimal alignment (see Section 2 for the definition) between x_{med} and x, and denote by $I_x \subset [n]$ the set of symbols moved (i.e., not aligned) by this alignment. Then by (3.2) we have $|I_x| = d(x, x_{\text{med}}) > (1-\delta)\mathtt{OPT}/m \geq (1-\delta)n/c$. Set $c' = \lceil \frac{c}{1-\delta} \rceil$ and $\xi = \frac{1}{2c'^2}$.

We now partition S into the far and close permutations (from x_{med}). Let $F = \{x \in S : d(x, x_{\text{med}}) \geq (1+\delta)\mathtt{OPT}/m\}$ and $\overline{F} = S \setminus F$. Since $\mathtt{OPT} = \sum_{x \in S} d(x, x_{\text{med}})$, by our assumption (3.2), $|F| \leq |\overline{F}|$. Thus $|\overline{F}| \geq m/2$. It follows that

$$\mathtt{OPT}_{\overline{F}} \geq \tfrac{m}{2} \cdot (1-\delta)\tfrac{\mathtt{OPT}}{m} = \tfrac{1-\delta}{2}\mathtt{OPT}. \tag{3.3}$$

Next, we partition $\overline{F}$ even further using the following procedure. Initialize a set $C = \emptyset$, and then iterate over the permutations $x \in \overline{F}$ in non-decreasing order of $|I_x|$. For each such x, if

$$\forall y \in C, \quad |I_x \cap I_y| < \xi n,$$

then add x to C and create its "buddies set" $B_x = \emptyset$; otherwise, pick $y \in C$ that violates the above, breaking ties arbitrarily, and add x to its buddies set B_y. Note that this partitioning is solely for the sake of analysis. Since $\overline{F}$ is processed in sorted order, it is clear that

$$\forall y \in C, x \in B_y, \qquad |I_y| \leq |I_x|. \tag{3.4}$$

We shall now prove two claims about this partitioning; the first one argues that at least a buddies set B_y (i.e., one "cluster") must be responsible for a large portion of the cost, and the second one bounds the distances from its "center" y.

Claim 3.3. *There exists $y \in C$ such that*

$$\mathit{OPT}_{B_y} \geq \frac{\mathit{OPT}_{\overline{F}}}{|C|} \geq \frac{\mathit{OPT}_{\overline{F}}}{2c'}. \tag{3.5}$$

To prove the claim, we shall need the following upper bound on the size of a family of subsets with small pairwise intersections.

Lemma 3.4. *For every $n, c' \in \mathbb{N}$ and $0 < \xi \leq \frac{2}{2c'^2}$, every family of subsets of $[n]$ in which every subset has size n/c' and every pair of subsets share at most ξn elements, has size at most $2c'$.*

We defer the proof of the above lemma to the end of this subsection. Now assuming the lemma we prove Claim 3.3.

Proof. [Proof of Claim 3.3] Lemma 3.4 applies to the set C because $\xi = \frac{1}{2c'^2}$, and by construction of C, all distinct $y, y' \in C$ satisfy $|I_y \cap I_{y'}| < \xi n$. We thus conclude that

$$|C| \leq 2c'.$$

Now since $\overline{F} = \bigcup_{x \in C} B_x$, a straightforward averaging implies the claim. ◻

Claim 3.5. *Suppose $y \in C$ satisfies (3.5). Then its distance to every $x \in S$ is bounded by:*

$$\forall x \in F, \quad d(x,y) \le 2d(x, x_{med}). \tag{3.6}$$

$$\forall x \in \overline{F}, \quad d(x,y) \le (2+4\delta)d(x, x_{med}) \tag{3.7}$$

$$\forall x \in B_y, \quad d(x,y) \le (2-\rho)d(x, x_{med}), \tag{3.8}$$

where $\rho = \frac{(1-\delta)(c'-1)}{2(1+\delta)c'^2}$.

Proof. To prove (3.6), consider $x \in F$. Since $y \in C \subseteq \overline{F}$ we have $d(y, x_{\text{med}}) \le d(x, x_{\text{med}})$, and thus by the triangle inequality, $d(x,y) \le d(x, x_{\text{med}}) + d(y, x_{\text{med}}) \le 2d(x, x_{\text{med}})$.

To prove (3.7), consider $x \in \overline{F}$. Since $y \in C \subseteq \overline{F}$ and using our assumption (3.2), we have $d(y, x_{\text{med}}) \le (1+\delta)\mathtt{OPT}/m \le \frac{1+\delta}{1-\delta} d(x, x_{\text{med}})$. Using $\delta \le 1/2$ and the triangle inequality, we obtain $d(x,y) \le d(x, x_{\text{med}}) + d(y, x_{\text{med}}) \le 2(1+2\delta)d(x, x_{\text{med}})$.

To prove (3.8), consider $x \in B_y$. Then

$$\begin{aligned} d(x,y) &\le |I_x| + |I_y| - |I_x \cap I_y| \\ &\le 2|I_x| - \xi n \\ &\qquad\qquad \text{[by (3.4)]} \\ &\le \left(2 - \frac{\xi n}{d(x, x_{\text{med}})}\right) d(x, x_{\text{med}}) \\ &\qquad\qquad \text{[since } |I_x| = d(x, x_{\text{med}})\text{]} \\ &\le \left(2 - \frac{(1-\delta)(c'-1)}{2(1+\delta)c'^2}\right) d(x, x_{\text{med}}) \end{aligned}$$

where the last inequality follows because $d(x, x_{\text{med}}) \le (1+\delta)\mathtt{OPT}/m \le (1+\delta)n/c$ and $c' = \lceil \frac{c}{1-\delta} \rceil$. □

We can now complete the proof of the lemma. Let $y \in C$ be as in Claims 3.3 and 3.5

$$\begin{aligned} \sum_{x \in S} d(x,y) &\le \sum_{x \in F} d(x,y) + \sum_{x \in \overline{F} \setminus B_y} d(x,y) + \sum_{x \in B_y} d(x,y) \\ &\le 2\mathtt{OPT}_F + (2+4\delta)\mathtt{OPT}_{\overline{F} \setminus B_y} + (2-\rho)\mathtt{OPT}_{B_y} \\ &\qquad\qquad \text{[by Claim 3.5]} \\ &\le 2\mathtt{OPT} + 4\delta\mathtt{OPT}_{\overline{F}} - \rho\mathtt{OPT}_{B_y} \\ &\le 2\mathtt{OPT} + 4\delta\mathtt{OPT}_{\overline{F}} - \rho\frac{\mathtt{OPT}_{\overline{F}}}{c'} \\ &\qquad\qquad \text{[by (3.5)]} \\ &\le 2\mathtt{OPT} - (\frac{\rho}{c'} - 4\delta)(1-\delta)\frac{\mathtt{OPT}}{2} \\ &\qquad\qquad \text{[by (3.3)]} \\ &\le \left(2 - \frac{(\frac{\rho}{c'} - 4\delta)(1-\delta)}{2}\right)\mathtt{OPT} \\ &\le (2 - \frac{1}{50c^2})\mathtt{OPT} \\ &\qquad\qquad \text{[for } \delta = \frac{1}{50c^2}\text{].} \end{aligned}$$

This concludes the proof of Lemma 3.2. □

It only remains to prove Lemma 3.4.

Proof. [Proof of Lemma 3.4] For contradiction sake, assume that there are $2c'$ subsets $Z_1, \cdots, Z_{2c'} \subseteq [n]$, such that

$$\forall i \in [2c'], |Z_i| \ge n/c', \tag{3.9}$$

$$\forall i \ne j \in [2c'], |Z_i \cap Z_j| \le n/2c'^2. \tag{3.10}$$

Clearly, $\left|\bigcup_{i \in [2c']} Z_i\right| \le n$. Now from simple inclusion-exclusion principle together with (3.9) and (3.10), we get

$$\begin{aligned} \Big|\bigcup_{i \in [2c']} Z_i\Big| &\ge \frac{n}{c'}2c' - \frac{n}{2c'^2}\binom{2c'}{2} \\ &= n + \frac{n}{2c'} > n \end{aligned}$$

which leads to a contradiction. Now the lemma follows. □

3.2 The general case We now argue the below 2-approximation guarantee for general input. Let us first recall a few notations from the last subsection and introduce a few more. Let x_{med} be an (arbitrary) median of S; then $\mathtt{OPT}(S) = \sum_{x \in S} d(x, x_{\text{med}})$, and for brevity we denote it by $\mathtt{OPT}$. For any subset $S' \subseteq S$ let $\mathtt{OPT}_{S'} = \sum_{x \in S'} d(x, x_{\text{med}})$. Let us take parameters $\delta, \alpha, \beta, \gamma, \xi, \eta$, the value of which will be set later. (Note, the parameters δ, ξ were also used in the last subsection, but their values will be set differently in this subsection.)

From now on we assume that

$$\forall x \in S, \quad d(x, x_{\text{med}}) > (1-\delta)\mathtt{OPT}/m, \tag{3.11}$$

because any $x' \in S$ that violates (3.11) is a $(2-\delta)$-approximate median of S (by the triangle inequality).

For each $x \in S$ consider an (arbitrary) optimal alignment g_x between x_{med} and x, and let I_x denote the set of symbols that are moved (i.e., not aligned) by this alignment. Note, $|I_x| = d(x, x_{\text{med}})$. For any $x \in S$ and subset of symbols $Z \subseteq [n]$, let $I_x(Z) = I_x \cap Z$.

For each symbol $a \in [n]$ and any subset $S' \subseteq S$, let

$$c_{S'}(a) = |\{x \in S' : a \text{ is moved by the alignment } g_x\}|.$$

For brevity when $S' = S$ we drop the subscript S' and simply use $c(a)$. For any subset $Z \subseteq [n]$ and $S' \subseteq S$, let $\mathtt{OPT}_{S'}(Z) = \sum_{a \in Z} c_{S'}(a)$. Again for brevity when $Z = [n]$ we only use $\mathtt{OPT}_{S'}$.

We call a symbol $a \in [n]$ *good* if $c(a) \le \alpha m$; otherwise *bad*. Let

$$G = \{a \in [n] : a \text{ is a good symbol}\},$$

and $\overline{G} = [n] \setminus G$. Now we divide our analysis into two cases depending on the size of $\overline{G}$.

Case 1: $|\overline{G}| \le \beta\frac{\texttt{OPT}}{m}$

LEMMA 3.6. *Let $\alpha \in (0, 1/10]$ and $\beta \in (0, 1)$. Given a set $S \subseteq \mathcal{S}_n$ of size m such that the set of bad symbols $\overline{G}$ is of size at most $\beta\frac{\texttt{OPT}}{m}$, Procedure* RELATIVEORDER*(S, α) outputs a $(1+\beta(1+8\alpha))$-approximate median.*

In this section we show Procedure RELATIVEORDER(S, α) finds a string $\tilde{x}$ such that $d(\tilde{x}, x_{\text{med}}) \le \frac{1}{1-4\alpha}|\overline{G}|$. Given set S, Procedure RELATIVEORDER() starts with the construction of the alignment graph $H = (V(H), E(H))$. Call a vertex *good* if its corresponding symbol is good; otherwise call it *bad*. We first make the following observation.

OBSERVATION 3.7. *Given a set $S \subseteq \mathcal{S}_n$ of size m and a parameter $0 < \alpha \le 1/10$, let G be the set of good symbols. For each pair of symbols $i, j \in G$ there exists either a directed edge (i, j) or (j, i) in $E(H)$.*

Proof. As both i and j are good symbols, $c(i)$ and $c(j)$ are at most αm. Now for the sake of contradiction, assume the observation is not correct. Then neither i precedes j, nor j precedes i in at least $(1-2\alpha)m$ strings of S. In this case, irrespective of the order of i and j in x_{med}, together they can be aligned in less than $(1-2\alpha)m$ strings of S. Hence $c(i)+c(j) > 2\alpha m$. But then at least one of $c(i)$ and $c(j)$ is strictly larger than αm and we get a contradiction. $\square$

Next we take the graph H and repetitively delete the shortest length cycle until the resultant graph $\overline{H} = (V(\overline{H}), E(\overline{H}))$ becomes acyclic. We make the following claim.

CLAIM 3.8. *Given a set $S \subseteq \mathcal{S}_n$ of size m and a parameter $0 < \alpha \le 1/10$, let H be the associated alignment graph. Let H^k be the graph obtained from H after k deletion steps and $\mathcal{C}^k_{min}$ be a shortest length cycle in H^k. Then for any $k \ge 0$, we claim the following.*

1. *Each cycle $\mathcal{C}^k$ of H^k has length at least $\frac{1}{2\alpha}$.*

2. *There exist at most two good vertices in $\mathcal{C}^k_{min}$.*

Proof. Consider a cycle $\mathcal{C}^k$ in H^k. Let i be some vertex and (j, i) be some edge contained in $\mathcal{C}^k$. Without loss of generality assume $\mathcal{C}^k$ be the shortest cycle containing i. Let p_{ij} be the path from i to j in $\mathcal{C}^k$. Note, p_{ij} is indeed the shortest path from i to j. To prove the first part we show if length of p_{ij} is ℓ then in at most $2\ell\alpha m$ strings of S, j precedes i. We prove this by induction on the length of p_{ij}. As a base case, consider the scenario when the path length is just one, that is there is a directed edge from i to j. Hence, in at least $(1-2\alpha)m$ strings of S, i precedes j and therefore in at most $2\alpha m$ strings j precedes i. Let the claim be true for path of length $\ell-1$. Now consider a shortest path $i = i_1 \to i_2 \to \cdots \to i_\ell \to i_{\ell+1} = j$ of length ℓ. Notice the length of the shortest path between i and i_ℓ is $\ell-1$. Hence in at most $2(\ell-1)\alpha m$ strings, i_ℓ precedes i. Now as there is a directed edge from i_ℓ to j, in at least $(1-2\alpha)m$ strings i_ℓ precedes j. Together we claim in at most $2(\ell-1)\alpha m + 2\alpha m = 2\ell\alpha m$ strings, j precedes i. Now as there is a directed edge from j to i, $2\ell\alpha m \ge (1-2\alpha)m$. So, $\ell \ge \frac{1-2\alpha}{2\alpha}$. Hence length of the cycle is at least $\ell + 1 \ge \frac{1}{2\alpha}$.

Fix two consecutive vertices i, j in $\mathcal{C}^k_{\min}$ such that the directed edge (j, i) is part of $\mathcal{C}^k_{\min}$. To prove the second part assume there are more than two good vertices, namely $v_1, v_2, \ldots v_{\ell'}$ appearing on p_{ij}. Moreover, assume they appear on the path p_{ij} in the above order. By Observation 3.7 between any v_q and v_r there is an edge. First we claim for each pair $q, r \in [\ell']$ where $q < r$, except both $v_q = i$ and $v_r = j$, the direction of the edge is from v_q to v_r. As otherwise let $\exists q < r$ where either $v_q \ne i$ or $v_r \ne j$ or both, the edge is from v_r to v_q. This gives rise to a cycle $(v_q, \ldots, v_r, v_q)$ which has length strictly smaller than the length of $\mathcal{C}^k_{\min}$, and thus we get a contradiction. Next we divide the proof into two cases.

Case i: (When at least one of i and j is a bad symbol) We have already seen, $\forall q, r \in [\ell']$ where $q < r$ the edge is from v_q to v_r. Following this there exists a directed edge from v_1 to $v_{\ell'}$. Hence, the concatenation of the path from i to v_1, the edge (v_1, v'_ℓ) and the path from $v_{\ell'}$ to j creates a path from i to j of length $|p_{ij}|-(\ell'-2) < |p_{ij}|$ as $\ell' > 2$, and we get a contradiction as we assumed $\mathcal{C}^k_{\min}$ to be the shortest length cycle.

Case ii: (When both i and j are good symbols) In this case, as we have already argued there must be an edge from $v_1 = i$ to v_2 and an edge from v_2 to $v_{\ell'} = j$. That implies there is a cycle (i, v_2, j, i) of length 3, which contradicts the first part of our claim that says each cycle must be of length at least $\frac{1}{2\alpha} \ge 5$ (for $\alpha \le 1/10$). $\square$

Recall, G is the set of good symbols (vertices). As a corollary of Claim 3.8 we have the following.

COROLLARY 3.9. $|G \setminus V(\overline{H})| \le \frac{4\alpha}{1-4\alpha}|\overline{G}|$.

Proof. By Claim 3.8, any cycle that we remove has at least $\frac{1}{2\alpha}$ vertices. Moreover, among them at most two are good. So the number of bad vertices in each removed cycle is at least $\frac{1-4\alpha}{2\alpha}$. Hence, total number of good vertices we remove is at most $\frac{4\alpha}{1-4\alpha}|\overline{G}|$. $\square$

Next we consider a topological ordering of $\overline{H}$. Using this we define an ordering $\mathcal{P}$ among the symbols of $V(\overline{H})$

as follows: For each $i, j \in V(\overline{H})$, i precedes j, denoted by $i \lhd j$ if i occurs before j in the topological sorted ordering. Let $\overline{x}$ be the string over the symbols of $V(\overline{H})$ obeying the ordering of $\mathcal{P}$. Note, $V(\overline{H})$ may not contain all the n vertices (or symbols). Create a permutation $\tilde{x}$ over $[n]$ by appending the symbols of $[n] \setminus V(\overline{H})$ at the end of string $\overline{x}$ in any arbitrary order. We claim the following.

LEMMA 3.10. *$d(\tilde{x}, x_{med}) \le \frac{1}{1-4\alpha}|\overline{G}|$.*

Before proving the lemma, we make the following claim.

CLAIM 3.11. *For any pair of symbols $i, j \in G \cap V(\overline{H})$, if $i \lhd j$, then i precedes j in x_{med}; otherwise j precedes i in x_{med}.*

Proof. For any pair of symbols $i, j \in G \cap V(\overline{H})$, as both the symbols $i, j \in G$, by Observation 3.7 there exists an edge between i and j in $\overline{H}$. So if $i \lhd j$, then there must exist a directed edge from i to j, and therefore in at least $(1-2\alpha)m$ strings i appears before j. As both i and j are aligned together in at least $(1-2\alpha)m$ strings and $1 - 2\alpha > 2\alpha$ (for $\alpha \le 1/10$), i precedes j in x_{med}. We can prove the other direction in a similar way. □

Proof. [Proof of Lemma 3.10] Following Claim 3.11, between $\tilde{x}$ and x_{med} there exists a common subsequence of length at least $|G \cap V(\overline{H})|$. Hence

$$\begin{aligned} d(\tilde{x}, x_{\text{med}}) &\le n - |G \cap V(\overline{H})| \\ &= |G| + |\overline{G}| - |G \cap V(\overline{H})| \\ &= |\overline{G}| + |G \setminus V(\overline{H})| \\ &\le \frac{1}{1-4\alpha}|\overline{G}| \qquad \text{[by Corollary 3.9]}. \end{aligned}$$

□

Proof. [Proof of Lemma 3.6] Procedure RELATIVEORDER(S, α) outputs a string $\tilde{x}$ such that $d(\tilde{x}, x_{\text{med}}) \le \frac{1}{1-4\alpha}|\overline{G}|$. Hence by triangle inequality,

$$\begin{aligned} \sum_{y \in S} d(\tilde{x}, y) &\le \sum_{y \in S} \Big(d(y, x_{\text{med}}) + d(x_{\text{med}}, \tilde{x}) \Big) \\ &\le \texttt{OPT} + \frac{m}{1-4\alpha}|\overline{G}| \qquad \text{[by Lemma 3.10]} \\ &\le \texttt{OPT} + \frac{\beta}{1-4\alpha}\texttt{OPT} \qquad [\text{as } |\overline{G}| \le \beta\frac{\texttt{OPT}}{m}] \\ &\le (1 + \beta(1 + 8\alpha))\texttt{OPT} \qquad [\text{as } \alpha \le 1/10]. \end{aligned}$$

□

REMARK 3.12. *We can improve the running time of Procedure* RELATIVEORDER *from $O(n^2m + n^4)$ to $O(n^2m + n^3)$ by slightly modifying it, without losing much on the approximation guarantee. Currently, Procedure* RELATIVEORDER *runs a while loop until there is no cycle in the graph H, and at each iteration computes a shortest cycle on the whole graph and delete all the vertices of that cycle (with all their incident edges). Instead of this while loop, we can enumerate over all the vertices and while enumerating a vertex v compute a shortest cycle that contains v and then delete all its vertices (with all their incident edges). Now each iteration takes only $O(n^2)$ time, and so the enumeration over all the vertices takes $O(n^3)$ time. Hence, the overall running time is $O(n^2m + n^3)$.*

The issue with this modification is that Claim 3.8 can get violated as each deleted cycle may contain more than two good vertices. However, we can claim that for any vertex v, in a shortest cycle $\mathcal{C}$ containing it, the ratio between the number of good and bad vertices is at most $3/(\frac{1}{2\alpha} - 2)$. The claim follows from two observations. First, for any two good vertices u_1, u_2 in $\mathcal{C}$, either they are consecutive in $\mathcal{C}$, or there are at least $\frac{1}{2\alpha} - 2$ bad vertices between them. To see this, take any two non-consecutive vertices u_1, u_2 in $\mathcal{C}$, and without loss of generality assume u_1 appears before u_2 in $\mathcal{C}$. By an argument similar to that in the proof of Claim 3.8, if there are at most $\frac{1}{2\alpha} - 3$ bad vertices between u_1, u_2 then there must be a directed edge from u_1 to u_2 in H, contradicting the cycle $\mathcal{C}$ being a shortest cycle containing v. Our second observation is that, if three good vertices u_1, u_2, u_3 form a 3-length subpath $u_1 \to u_2 \to u_3$ in $\mathcal{C}$ (which is a shortest cycle containing v), then $u_2 = v$. This is because, since u_1, u_2, u_3 all are good vertices and there is an edge from u_1 to u_2 and from u_2 to u_3, there must be an edge also from u_1 to u_3, and therefore if $u_2 \neq v$, we will get a shorter cycle by following the edge u_1 to u_3 contradicting $\mathcal{C}$ being a shortest cycle containing v. Our claimed bound on the ratio of good and bad symbols now follows from these two observations. Hence, at the end of enumeration, the number of good symbols (or vertices) that got deleted is at most $\frac{6\alpha}{1-4\alpha}|\overline{G}|$ (which is slightly worse than that in Corollary 3.9). The rest of the argument will remain the same, and therefore finally, we will get a $(1+\beta(1+12\alpha))$-approximate median.

Case 2: $|\overline{G}| > \beta\frac{\texttt{OPT}}{m}$ Recall, we take parameters $\delta, \alpha, \beta, \gamma, \xi, \eta$, the value of which will be set later.

LEMMA 3.13. *Let $\alpha \in (0, 1/10]$ and $\beta \in (0, 1)$. Given $S \subseteq \mathcal{S}_n$ of size m such that the number of bad symbols is $|\overline{G}| \ge \beta\frac{OPT}{m}$, Procedure* BESTFROMINPUT*(S) outputs a $(2 - \zeta)$-approximate median, where $\zeta =$*

$\frac{(1-\alpha/2)\alpha^5\beta^2}{2^{20}\log^2_{(1+\frac{3\alpha}{64})}(8/3\alpha)}$.

We would like to mention that the constant $1/2^{20}$ in the above lemma is not optimal, and one can improve this significantly by optimizing various parameters. The proof of the above lemma resembles that of Lemma 3.2, however it is much more intricate. We defer the proof to the full version.

Proof of Theorem 1.1. Recall, if our input set S violates assumption (3.11), then we get a $(2-\delta)$-approximate median using Procedure BestFromInput. Set $\delta = \frac{\alpha^6\beta^2}{2^{19}\log^2_{(1+\frac{3\alpha}{64})}(8/3\alpha)}$. Next, set $\alpha = 1/10$ and $\beta = 1/2$. Now Theorem 1.1 follows from Lemma 3.6 and 3.13.

3.3 Generalization to Edit Distance (for the High Regime)

So far, all our results are only for the Ulam metric. In this section, we will describe how to extend our result of Section 3.1 to the edit metric space, which is a generalization of the Ulam. The edit distance between two strings is defined as the minimum number of insertion, deletion and character substitution operations required to transform one string into another. For the simplicity in exposition, we start with a special variant of the edit distance, where character substitution is not allowed. (Originally, Levenshtein [Lev65] defined both the variants, with and without the substitution operation.) In this section, we refer this special variant also as the edit distance. For any two strings x, y, their edit distance, denoted by $\Delta(x, y)$, is the minimum number of insertion and deletion operations to transform x into y. So $\Delta(x) = |x| + |y| - |\mathrm{lcs}(x, y)|$.

We now define the median under the edit distance metric, requiring it has the same length as the input strings. Formally, the *length-n edit-median* of a set of strings $S \subseteq \Sigma^n$ is a string $x_{\mathrm{med}} \in \Sigma^n$ such that $\sum_{x\in S}\Delta(x, x_{\mathrm{med}})$ is minimized. A c-approximate length-n edit-median is defined analogous to that for the Ulam metric.

Theorem 3.14. *Given a set of strings $S \subseteq \Sigma^n$ whose optimal median objective value is at least $|S|n/c$ for some $c > 1$, Procedure* BestFromInput *reports a $(2 - \frac{1}{50c^2})$-approximate length-n edit-median in time $O(nm^2 \log n)$.*[3]

Let $x_{\mathrm{med}} \in \Sigma^n$ be an (arbitrary) median of S; then $\mathtt{OPT}(S) = \sum_{x\in S}\Delta(x, x_{\mathrm{med}})$. We use the argument used in the proof of Lemma 3.2, but change the definition of I_x for $x \in S$ as follows: Fix an optimal alignment (or a lcs) between x_{med} and x, and let I_x be the set of positions $i \in [n]$ such that $x_{\mathrm{med}}(i)$ is not aligned by this alignment. Notice that $|I_x| = \Delta(x, x_{\mathrm{med}})/2$ since x and x_{med} have the same length n. Furthermore, for all $x \neq y \in S$,

$$|\mathrm{lcs}(x, y)| \geq |\overline{I_x} \cap \overline{I_y}|,$$

because the positions in $\overline{I_x} \cap \overline{I_y}$ define a subsequence of x_{med} that is common to both x and y. Thus,

$$\begin{aligned}\Delta(x, y) &= 2(n - |\mathrm{lcs}(x, y)|)\\ &\leq 2(n - |\overline{I_x} \cap \overline{I_y}|)\\ &= 2|I_x \cup I_y|\\ &= 2(|I_x| + |I_y| - |I_x \cap I_y|)\\ &\leq \Delta(x, x_{\mathrm{med}}) + \Delta(y, x_{\mathrm{med}}) - |I_x \cap I_y|. \end{aligned} \tag{3.12}$$

Then we follow the argument as in the proof of Lemma 3.2 to identify a point $y \in \Sigma^n$ (a cluster) as in Claim 3.3, and bound the distance from y to all $x \in S$ as in Claim 3.5. To prove the bound (3.8), we use (3.12), and the rest of the arguments will remain the same.

Remark 3.15. *We can further extend our proof to a more generalized edit distance notion, with character substitution also as a valid edit operation. In this case, the proof will be slightly more involved (by considering different cases depending on whether the unaligned index positions are for substitutions or deletions). However, if we allow the median string to be of arbitrary length (not necessarily the same as that of input strings), our proof will fail. Indeed, in this case, there exists an input set S with $\mathtt{OPT} \geq \Omega(n|S|)$ such that Procedure* BestFromInput *(S) does not achieve approximation better than factor 2.*

4 Approximate Median in a Probabilistic Model

Consider a permutation $x \in \mathcal{S}_n$. Then take a set of "noisy" copies of x, where each noisy copy is generated from x by moving "a few" randomly chosen symbols in randomly chosen positions. Formally, for any $\epsilon \in (0, 1)$ define $S(x, \epsilon, m)$ as a set of m permutations $x_1, \cdots, x_m \in \mathcal{S}_n$ such that for each $i \in [m]$ x_i is generated from x in the following way: Select each symbol in $[n]$ independently with probability ϵ. Let the set of selected symbols be Σ_i. For each symbol $a \in \Sigma_i$ choose another symbol $b_i(a)$ independently uniformly at random from $[n]$, and then move the symbol a from its original position (in x) to right next to $b_i(a)$. Let $\Sigma_i^b = \{b_i(a) : a \in \Sigma_i\}$.

Denote the set of all move operations performed to generate x_i by the set of tuples $(a, b_i(a))$. Let $\Sigma_i^e = \{(a, b_i(a)) : a \in \Sigma_i\}$. For each $i \in [m]$, define set $\Sigma_i^r = \{a \in \Sigma_i : b_i(a) \in \Sigma_i\}$.

[3] We make no attempt to optimize the constants.

Given S drawn from $S(x,\epsilon,m)$, the objective is to find its median. Throughout this section, all the probabilities are over the randomness used to generate this set S. Now we state the main theorem of this section (which is a restatement of Theorem 1.3).

THEOREM 4.1. *Fix a parameter $\epsilon \in (0,1/40)$, a permutation $x \in \mathcal{S}_n$, and $40 \le m \le n$. There is an $O(n^3)$-time deterministic algorithm that, given input S drawn from $S(x,\epsilon,m)$, outputs a $(1+\delta)$-approximate median of S, for $\delta = \frac{20}{m} + \frac{3}{\log(n/\epsilon)} + \frac{2e^{-m/40}}{\epsilon}$, with probability at least $1-5/m$.*

Next, we state an important observation about permutations in $S(x,\epsilon,m)$, following the simple application of Chernoff bound.

OBSERVATION 4.2. *For any $\epsilon \in (0,1)$, any $n \in \mathbb{N}$, a permutation $x \in \mathcal{S}_n$ and any $m \in \mathbb{N}$, let $S = S(x,\epsilon,m)$. Then the followings hold.*

1. *For any $i \in [m]$, $\Pr[|\Sigma_i| \notin (1 \pm \frac{1}{\sqrt{\log n}})\epsilon n] \le e^{-\epsilon n/4\log n}$.*
2. *For any two $x_i \neq x_j \in S$, $\Pr[|\Sigma_i \cap \Sigma_j| \notin (1 \pm \frac{1}{\sqrt{\log n}})\epsilon^2 n] \le e^{-\epsilon^2 n/4\log n}$.*

4.1 Hidden Permutation and Approximate Median

To prove Theorem 1.3 we design an algorithm that given a set S drawn from $S(x,\epsilon,m)$, finds a "good approximation" of x. Recall, our main goal is to find a median permutation x_{med} for S. The following theorem explains why it suffices to find x instead of an actual median.

THEOREM 4.3. *For every $\epsilon \in (0,1/12)$, any large enough $n \in \mathbb{N}$, a permutation $x \in \mathcal{S}_n$, $20 \le m \le n$ and $\delta = \frac{20}{m} + \frac{3}{\log(n/\epsilon)}$, for a set of permutations S drawn from $S(x,\epsilon,m)$,*

$$\Pr[\texttt{Obj}(S,x) \le (1+\delta)\texttt{OPT}(S)] \ge 1 - mn^{-1.5}.$$

Our proof will go via an information-theoretic (encoding-decoding based) argument. First, we will argue that one can encode the set S by specifying the move operations to produce x_i's from a median x_{med}. Then we will show that given x, using x_i's and extra "few" bits, one can decode all the random move operations of Σ_i^e's. Now using Shannon's source coding theorem we will get a lower bound on the optimum median objective value $\texttt{OPT}(S) = \sum_{x_i \in S} d(x_{\text{med}}, x_i)$. Then compare that with the value obtained by x, i.e., $\texttt{Obj}(S,x) = \sum_{x_i \in S} d(x,x_i)$ to get the claimed approximation guarantee. We defer the detailed proof to the full version.

4.2 Finding the Hidden Permutation

In the last section, we have seen that to find an approximate median of a set S drawn from $S(x,\epsilon,m)$, it suffices to find the permutation x. So from now on, we will focus only on finding x (approximately).

When m is large Apparently the task of finding the unknown x becomes much easier when $m \ge \Omega(\log n)$.

LEMMA 4.4. *For any $\epsilon \in (0,1/16)$, a large enough $n \in \mathbb{N}$, a permutation $x \in \mathcal{S}_n$, and any $m \ge 32\log n$, let S be drawn from $S(x,\epsilon,m)$. There is an $O(n\log^2 n)$ time algorithm that given S outputs x with probability at least $1-1/n$.*

Note, running time of the algorithm is independent of m. The reason is that our algorithm will take an arbitrary $\Theta(\log n)$-sized subset of S and compute x.

Proof. Finding x is nothing but sorting the numbers in $[n]$ according to the order specified by x. Before proceeding further, let us introduce a notation that we will use henceforth. For any two distinct symbols $a,b \in [n]$ if a appears before b in x we use the notation $a <_x b$. Below we describe our algorithm.

Without loss of generality, assume that $m = 32\log n$; otherwise, take an arbitrary subset $S' \subseteq S$ of size $32\log n$ and perform our algorithm with S' instead of S. To sort the symbols according to the ordering of x, we use the Mergesort [4] with additional query access to the set S. While performing the Mergesort whenever two elements $a,b \in [n]$ will be compared to check whether $a <_x b$, we will use the following query algorithm.

Query algorithm (a,b)**:** Compare a,b in all $x_i \in S$. If at least in $m/2$ many x_i's a appears before b, then return $a <_x b$; else return $b <_x a$.

It follows from the time complexity of the Mergesort that the algorithm will make at most $O(n\log n)$ queries to our query algorithm. Each such query takes $O(m)$ time. So the total running time of our algorithm is $O(n\log^2 n)$, since by our assumption $m = 32\log n$.

Now it only remains to prove the correctness of our algorithm. For each $a \in [n]$ let $B_a = \{x_i \in S : a \in \Sigma_i\}$. Take a parameter $\delta = \frac{1}{4\epsilon} - 2$. We call a symbol $a \in [n]$ *bad* if $|B_a| \ge (1+\delta)\epsilon m$. (Note, here the definition of a bad symbol is similar to that used in Section 3.2. The only difference is that here our "unknown reference" is x instead of a median string x_{med}.) Consider any symbol $a \in [n]$. Then $\mathbb{E}[|B_a|] = \epsilon m$. Since Σ_i's are generated independently of each other, by Chernoff bound

$$\Pr[a \text{ is bad}] \le e^{-\frac{\delta^2\epsilon m}{2+\delta}}.$$

[4] One may take any comparison-based sorting algorithm instead of the Mergesort; the running time will change accordingly.

Then by a union bound over all symbols,

$$\Pr[\text{None of symbols is bad}] \geq 1 - ne^{-\frac{\delta^2 \epsilon m}{2+\delta}} \geq 1 - 1/n$$

where the last inequality holds for $\epsilon < 1/16$, $\delta = \frac{1}{4\epsilon} - 2$ and $m = 32 \log n$.

Observe, for any two distinct symbols $a, b \in [n]$ if $a <_x b$ and none of them is bad, then the number of x_i's in S in which in a appears before b is at least $(1 - 2(1+\delta)\epsilon)m > m/2$ for $\delta = \frac{1}{4\epsilon} - 2$. Thus our query algorithm always outputs a correct order among two symbols (given none of them is bad). The correctness now follows from the correctness of the Mergesort. $\square$

When m is small

Lemma 4.5. *For any $\epsilon \in (0, 1/40)$, a large enough $n \in \mathbb{N}$, a permutation $x \in \mathcal{S}_n$, and any m, let S be drawn from $S(x, \epsilon, m)$. There is a (deterministic) algorithm that given S, outputs a permutation $\tilde{x} \in \mathcal{S}_n$ such that $d(x, \tilde{x}) \leq \frac{5}{3}(e^{-m/40} + 2\sqrt{\log n/n})n$ in time $O(n^3 + mn^2)$ with probability at least $1 - 1/n$.*

Proof. Before describing the algorithm let us introduce a few notations to be used in this proof. For each $a \in [n]$ let $B_a = \{x_i \in S : a \in \Sigma_i\}$. Take a parameter $\delta = \frac{1}{10\epsilon} - 2$. We call a symbol $a \in [n]$ *bad* if $|B_a| \geq \alpha|S| = \alpha m$, where $\alpha = (1+\delta)\epsilon$. (Note, here the definition of a bad symbol is similar to that used in Section 3.2. The only difference is that here our "unknown reference" is x instead of a median string x_{med}.) Let

$$G = \{a \in [n] : a \text{ is not bad}\},$$

and $\overline{G} = [n] \setminus G$.

Now we run the procedure RelativeOrder (described in Section 3) with S and α as input. Next we show that this procedure will return a $\tilde{x} \in \mathcal{S}_n$ with the desired distance bound from x.

Consider any symbol $a \in [n]$. Then $\mathbb{E}[|B_a|] = \epsilon m$. Since Σ_i's are generated independently of each other, by Chernoff bound

$$\Pr[a \text{ is bad}] \leq e^{-\frac{\delta^2 \epsilon m}{2+\delta}}.$$

Let $p = e^{-\frac{\delta^2 \epsilon m}{2+\delta}} \leq e^{-m/40}$ for any $\epsilon \in (0, 1/40)$. So $\mathbb{E}[|G|] \geq (1-p)n$. Since a symbols is bad independent of any other symbol being bad, by Chernoff bound for any $\delta' \in (0,1)$,

$$\begin{aligned} \Pr[|G| \geq (1-\delta')\mathbb{E}[|G|]] &\geq 1 - e^{-\frac{\delta'^2 \mathbb{E}[|G|]}{2}} \\ &\geq 1 - e^{-\frac{\delta'^2 (1-p)n}{2}}. \qquad (4.13) \end{aligned}$$

Note, by our choice of parameter δ for any $\epsilon \in (0, 1/40)$, $\alpha \in (0, 1/10)$. So by an argument exactly the same as that used in the proof of Lemma 3.10, we get that

$$\begin{aligned} d(x, \tilde{x}) &\leq \frac{1}{1-4\alpha}|\overline{G}| \\ &\leq \frac{1}{1-4\alpha}(p + \delta'(1-p))n \\ &\leq \frac{5}{3}(e^{-m/40} + \delta')n \qquad \text{since } \alpha < 1/10 \end{aligned}$$

where the second inequality holds with probability at least $1 - e^{-\frac{\delta'^2(1-p)n}{2}}$ by (4.13). Now to finish the proof set $\delta' = 2\sqrt{\log n/n}$. $\square$

Proof of Theorem 1.3. Now we are ready to finish the proof of Theorem 1.3. For $m \geq 32 \log n$, Theorem 4.3 together with Lemma 4.4 shows that in time $O(n \log^2 n)$ we can find a $(1+\delta)$-approximate median of S drawn from $S(x, \epsilon, m)$, for $\delta = \frac{20}{m} + \frac{3}{\log(n/\epsilon)}$ with probability at least $1 - mn^{-1.5}$.

For any $m < 32 \log n$ by Lemma 4.5 we get a $\tilde{x}$ such that $d(x, \tilde{x}) \leq \frac{5}{3}(e^{-m/40} + 2\sqrt{\log n/n})n$ with probability at least $1 - 1/n$. Let $\gamma = e^{-m/40} + 2\sqrt{\log n/n}$.

$$\begin{aligned} \texttt{Obj}(S, \tilde{x}) &= \sum_{x_i \in S} d(x_i, \tilde{x}) \\ &\leq \sum_{x_i \in S} d(x_i, x) + md(x, \tilde{x}) \\ &\qquad \text{[by the triangle inequality]} \\ &\leq \texttt{Obj}(S, x) + \frac{5}{3}\gamma nm \\ &\qquad \text{[by Lemma 4.5]} \\ &\leq \left(1 + \frac{\gamma}{\epsilon}\right)\texttt{Obj}(S, x) \\ &\qquad \text{[by Observation 4.2 w.p.} \geq 1 - 1/m] \\ &\leq \left(1 + \frac{\gamma}{\epsilon}\right)\left(1 + \frac{20}{m} + \frac{3}{\log(n/\epsilon)}\right)\texttt{OPT}(S) \\ &\qquad \text{[by Theorem 4.3]} \\ &\leq \left(1 + \frac{20}{m} + \frac{3}{\log(n/\epsilon)} + \frac{2e^{-m/40}}{\epsilon}\right)\texttt{OPT}(S) \\ &\qquad \text{[by replacing the value of } \gamma]. \end{aligned}$$

This concludes the proof of Theorem 1.3.

Acknowledgements. Diptarka would like to thank Djordje Jevdjic for some useful discussion on [RMR+17].

References

[ACN08] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, 2008.

[AD99] D. Aldous and P. Diaconis. Longest increasing subsequences: from patience sorting to the Baik-Deift-Johansson theorem. *Bulletin of the American Mathematical Society*, 36(4):413–432, 1999.

[AK10] A. Andoni and R. Krauthgamer. The computational hardness of estimating edit distance. *SIAM J. Comput.*, 39(6):2398–2429, 2010.

[AN10] A. Andoni and H. L. Nguyen. Near-optimal sublinear time algorithms for Ulam distance. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 76–86, 2010.

[ARJ14] J. Abreu and J. R. Rico-Juan. A new iterative algorithm for computing a quality approximate median of strings based on edit operations. *Pattern Recognition Letters*, 36:74–80, 2014.

[BS19] M. Boroujeni and S. Seddighin. Improved MPC algorithms for edit distance and Ulam distance. In *The 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019, Phoenix, AZ, USA, June 22-24, 2019.*, pages 31–40, 2019.

[CA97] F. Casacuberta and M. Antonio. A greedy algorithm for computing approximate median strings. In *Proc. of National Symposium on Pattern Recognition and Image Analysis*, pages 193–198, 1997.

[CCGB$^+$17] H. Cardot, P. Cénac, A. Godichon-Baggioni, et al. Online estimation of the geometric median in Hilbert spaces: Nonasymptotic confidence balls. *Annals of Statistics*, 45(2):591–614, 2017.

[CDKL14] F. Chierichetti, A. Dasgupta, R. Kumar, and S. Lattanzi. On reconstructing a hidden permutation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014).* Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[CK06] M. Charikar and R. Krauthgamer. Embedding the Ulam metric into l_1. *Theory of Computing*, 2(11):207–224, 2006.

[CKPV10] F. Chierichetti, R. Kumar, S. Pandey, and S. Vassilvitskii. Finding the Jaccard median. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 293–311. SIAM, 2010.

[CLM$^+$16] M. B. Cohen, Y. T. Lee, G. Miller, J. Pachocki, and A. Sidford. Geometric median in nearly linear time. In *Proceedings of the forty-eighth annual ACM Symposium on Theory of Computing*, pages 9–21, 2016.

[CMS01] G. Cormode, S. Muthukrishnan, and S. C. Sahinalp. Permutation editing and matching via embeddings. In *International Colloquium on Automata, Languages, and Programming*, pages 481–492. Springer, 2001.

[Coh19] V. Cohen-Addad. Fine grained approximation algorithms and complexity (FG-APX 2019). 2019.

[Cor03] G. Cormode. *Sequence distance embeddings*. PhD thesis, Department of Computer Science, 2003.

[DKNS01] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10*, pages 613–622, 2001.

[dlHC00] C. de la Higuera and F. Casacuberta. Topology of strings: Median string is NP-complete. *Theor. Comput. Sci.*, 230(1-2):39–48, 2000.

[FOR17] D. Feldman, S. Ozer, and D. Rus. Coresets for vector summarization with applications to network graphs. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, pages 1117–1125, 2017.

[FVJ08] P. T. Fletcher, S. Venkatasubramanian, and S. Joshi. Robust statistics on riemannian manifolds via the geometric median. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[FZ00] I. Fischer and A. Zell. String averages and self-organizing maps for strings. *Proceedings of the neural computation*, pages 208–215, 2000.

[GBC$^+$13] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature*, 494(7435):77–80, 2013.

[Gus97] D. Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology.* Cambridge University Press, 1997.

[HK16] M. Hayashida and H. Koyano. Integer linear programming approach to median and center strings for a probability distribution on a set of strings. In *BIOINFORMATICS*, pages 35–41, 2016.

[Ind99] P. Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 428–434, 1999.

[Kem59] J. G. Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

[KMS07] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 95–103, 2007.

[Koh85] T. Kohonen. Median strings. *Pattern Recognition Letters*, 3(5):309–313, 1985.

[Kru83] J. B. Kruskal. An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM review*, 25(2):201–237, 1983.

[Kru99] F. Kruzslicz. Improved greedy algorithm for computing approximate median strings. *Acta Cybernetica*, 14(2):331–339, 1999.

[Lev65] V. Levenshtein. Binary codes capable of correcting spurious insertions and deletion of ones. *Problems of information Transmission*, 1(1):8–17, 1965.

[MAS19] P. Mirabal, J. Abreu, and D. Seco. Assessing the best edit in perturbation-based iterative refinement algorithms to compute the median string. *Pattern Recognition Letters*, 120:104–111, Apr 2019.

[Min15] S. Minsker. Geometric median and robust estimation in banach spaces. *Bernoulli*, 21(4):2308–2335, 2015.

[MJC00] C. D. Martínez-Hinarejos, A. Juan, and F. Casacuberta. Use of median string for classification. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 2, pages 903–906. IEEE, 2000.

[NR03] F. Nicolas and E. Rivals. Complexities of the centre and median string problems. In *Combinatorial Pattern Matching, 14th Annual Symposium, CPM 2003, Morelia, Michocán, Mexico, June 25-27, 2003, Proceedings*, pages 315–327, 2003.

[NSS17] T. Naumovitz, M. E. Saks, and C. Seshadhri. Accurate and nearly optimal sublinear approximations to Ulam distance. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2012–2031, 2017.

[PB07] O. Pedreira and N. R. Brisaboa. Spatial selection of sparse pivots for similarity search in metric spaces. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 434–445. Springer, 2007.

[Pev00] P. Pevzner. *Computational molecular biology: an algorithmic approach.* MIT press, 2000.

[RCS13] R. J. Roberts, M. O. Carneiro, and M. C. Schatz. The advantages of smrt sequencing. *Genome Biology*, 14(7):405, 2013.

[RMR$^+$17] C. Rashtchian, K. Makarychev, M. Z. Rácz, S. Ang, D. Jevdjic, S. Yekhanin, L. Ceze, and K. Strauss. Clustering billions of reads for DNA data storage. In *Advances in Neural Information Processing Systems 30*, pages 3360–3371. Curran Associates, Inc., 2017.

[San75] D. Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28(1):35–42, 1975.

[Sch12] W. Schudy. Approximation schemes for inferring rankings and clusterings from pairwise data. *Ph.D. Thesis*, 2012.

[YL78] H. P. Young and A. Levenglick. A consistent extension of condorcet's election principle. *SIAM Journal on applied Mathematics*, 35(2):285–300, 1978.

[You88] H. P. Young. Condorcet's theory of voting. *American Political science review*, 82(4):1231–1244, 1988.