

On the Maximal Independent Sets of k -mers with the Edit Distance

Leran Ma
lkm5463@psu.edu
The Pennsylvania State University
Schreyer Honors College
Department of Computer Science and
Engineering
State College, United States

Ke Chen
kxc5915@psu.edu
The Pennsylvania State University
Department of Computer Science and
Engineering
State College, United States

Mingfu Shao
mxs2589@psu.edu
The Pennsylvania State University
Department of Computer Science and
Engineering
Huck Institutes of the Life Sciences
State College, United States

ABSTRACT

In computational biology, k -mers and edit distance are fundamental concepts. However, little is known about the metric space of all k -mers equipped with the edit distance. In this work, we explore the structure of the k -mer space by studying its maximal independent sets (MISs). An MIS is a sparse sketch of all k -mers with nice theoretical properties, and therefore admits critical applications in clustering, indexing, hashing, and sketching large-scale sequencing data, particularly those with high error-rates. Finding an MIS is a challenging problem, as the size of a k -mer space grows geometrically with respect to k . We propose three algorithms for this problem. The first and the most intuitive one uses a greedy strategy. The second method implements two techniques to avoid redundant comparisons by taking advantage of the locality-property of the k -mer space and the estimated bounds on the edit distance. The last algorithm avoids expensive calculations of the edit distance by translating the edit distance into the shortest path in a specifically designed graph. These algorithms are implemented and the calculated MISs of k -mer spaces and their statistical properties are reported and analyzed for k up to 15. Source code is freely available at <https://github.com/Shao-Group/kmerspace>.

CCS CONCEPTS

• Applied computing → Computational genomics; • Mathematics of computing → Graph algorithms.

KEYWORDS

k -mer space, maximal independent set, sequence clustering, sketching, indexing, locality-sensitive hashing

ACM Reference Format:

Leran Ma, Ke Chen, and Mingfu Shao. 2023. On the Maximal Independent Sets of k -mers with the Edit Distance. In *14th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '23)*, September 3–6, 2023, Houston, TX, USA. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3584371.3612982>

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
BCB '23, September 3–6, 2023, Houston, TX, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0126-9/23/09.
<https://doi.org/10.1145/3584371.3612982>

1 INTRODUCTION

Given an alphabet Σ , we denote by S_k the set of all sequences of length k over Σ . Clearly, $|S_k| = |\Sigma|^k$. In this work, we call a sequence of length k a k -mer, and call S_k the k -mer space, for the sake of convenience. The Levenshtein distance [14], also known as the edit distance, between two sequences u and v , denoted as $\text{edit}(u, v)$, is defined as the minimum number of insertions, deletions, and substitutions needed to transform u into v . The edit distance is a metric; the k -mer space together with the edit distance (S_k, edit) forms a metric space.

Despite the ubiquitous use of k -mers and the edit distance in computational biology, the properties and structures of the metric space (S_k, edit) remain largely unknown. It is worth noting that (S_k, edit) is intrinsically different from the well-studied metric spaces on sequences such as the Hamming distance space or other normed spaces (when a k -mer is represented as a vector of dimension k), as evidenced by the fact that embedding the edit distance on $\{0, 1\}^k$ into the L_1 space requires a distortion of $\Omega(\log k)$ [12, 20].

In this work, we study the *maximal independent set* (MIS) of a k -mer space parameterized with an integer $d \geq 0$. Formally, we say a subset $M \subset S_k$ is *independent* if $\text{edit}(u, v) > d$ for every two k -mers u and v in M , and we say an independent subset M is *maximal* if there does not exist another independent subset M' such that $M \subsetneq M'$. These definitions can be equivalently stated using the language of graph. Given an integer $d \geq 0$, we define an undirected graph $G_k^d = (V_k = S_k, E_k^d)$ where there is an edge $(u, v) \in E_k^d$ if $\text{edit}(u, v) \leq d$. The above defined MISs of the k -mer space with respect to an integer d is equivalent to MISs in the graph G_k^d .

1.1 Applications

We address the problem of finding an MIS of a k -mer space. Solving this problem helps understand the structure of the metric space (S_k, edit) as an MIS indicates how dense a k -mer space is under the edit distance. More importantly, finding MISs of G_k^d has critical applications in large-scale sequence analysis thanks to its nice properties. We elaborate on a few below.

First, an MIS M naturally can be used as a set of “centers” for clustering k -mers. The centers (k -mers in an M) are distant from each other which means that clusters are not crammed together. In addition, every k -mer v can find a “nearby” center, as there exists a k -mer $u \in M$ such that $\text{edit}(u, v) \leq d$ (otherwise $M \cup \{v\}$ is an independent set that strictly contains M). These properties make an MIS an ideal set of centers for clustering k -mers.

Second, k -mers of an MIS can be used to design a new seeding scheme for aligning error-prone reads. Modern fast aligners often use exact k -mer matches as anchors (or seeds) [5, 15]. However, such methods exhibit low sensitivity when aligning reads with high error-rate (i.e., long-reads data generated by PacBio [22] and Oxford Nanopore [9]), because under high error-rate, two biologically related sequences hardly share any identical k -mers (for a reasonable choice of k such as $k = 15$ and an error rate of 15%). One can choose to use smaller k but this will lead to high false positives as unrelated sequences can, by chance, share many short k -mers. A new seeding scheme can be designed by first mapping a k -mer into its nearest k -mer u in an MIS M : two k -mers form an anchor if they are mapped to the same k -mer in M . The advantage of this MIS-based seeding scheme is that it tolerates edits in k -mers and is therefore more sensitive for error-prone data. Comparing to k -mer-alternative seeding methods such as spaced seeds [3, 17] and indel seeds [18] where only restricted types of edits are allowed (e.g., spaced seeds cannot model indels), this MIS-based scheme can recognize similar k -mers with both indels and substitutions. Moreover, the intrinsic properties of an MIS also control the false-positives: by triangle inequality, two distant k -mers that are $2d$ edits apart will never be mapped to the same k -mer.

Third, an MIS of the k -mer space can be used to improve *sketching* approaches. Sketching enables scaling to large datasets by only selecting a subset of representative k -mers in a sequence. Existing k -mer sketching methods such as FracMinHash [1, 8] often use a random permutation of k -mers and pick the top fraction as representative k -mers. By using an MIS as a representative subset instead, the selected k -mers are guaranteed to be at least of d edits apart of each other, and are hence more efficient.

1.2 Related Work

All above applications require finding an MESA in the first place. However, computing an MAYS of a k -MIA space is difficult, simply because the number of k -moss $|S_k| = |\Sigma|^k$ (equivalently, the number of vertices in the graph G_k^d) grows exponentially with k and the number of edges in the graph $|E_k^d| = O(|\Sigma|^{2k})$ grows even faster. Existing algorithms for finding an MASS in a general graph, for example, the greedy algorithm that takes $O(m)$ time for a graph with m edges, are therefore not suitable for our problem. In fact, even constructing E_k^d explicitly is often unaffordable.

In graph theory, both the maximum-size MIS (i.e., maximum independent set) and minimum-size MIS (a.k.a, independent dominating set) are extensively studied. Both problems are known to be NP-hard and cannot be approximated within a constant factor in polynomial time unless $P=NP$. Polynomial-time algorithms exist only for restricted classes of graphs, see the survey [7].

For applications with massive graphs such as social networks and traffic planning, researches have been focusing on developing heuristics that find a large-size independent set [4, 13]. Different practical computation models are also considered, including on GPUs [2], on semi-external setting [16] where the edges of the graphs are loaded on demand from disk, and on dynamic graphs [6] where a large MIS needs to be maintained as the graph changes.

We note that the graphs studied in the above context are often at least an order of magnitude smaller and are several orders of

magnitude sparser than the k -mer space ($k = 15$) studied in this paper. Hence their results cannot be used for our purpose. Furthermore, these approaches assume that the graph under consideration is given which does not hold in our situation. As mentioned above, computing and storing the entire k -mer space would be largely inefficient, if not infeasible. In fact, one of the contributions of our algorithms is that we efficiently exploit the structural information of the k -mer space to avoid the expensive edit distance computation for testing if there is an edge between two vertices.

1.3 Results

In this work, we design three algorithms for finding an MIS of a k -mer space. Our algorithms take into account the special properties of k -mers and the edit distance so that the underlying graph G_k^d does not need to be explicitly constructed. Our algorithms are able to scale to instances with $k = 15$. We implemented these algorithms and calculated an MIS for all combinations of k and d for $k \leq 15$ and reported their statistics. We also analyzed and concluded when to use which algorithm for different combinations of k and d .

2 ALGORITHMS

We design three algorithms for finding an MIS of all k -mers, given k and d . The first one is a greedy algorithm that is similar to the greedy algorithm for computing an MIS on general graphs but without explicitly building all edges of the underlying graph. The second algorithm improves the first one by reducing redundant comparisons through recognizing the locality properties of k -mers and estimating and incorporating the bounds of the edit distance into the algorithm. The last algorithm transforms calculating the edit distance into finding the shortest path in a specifically constructed graph, and finding an MIS is then transformed into efficient graph traversing together with accompanied data structures to speed up.

2.1 A Simple Greedy Algorithm

This algorithm maintains a (dynamic) array M that stores the current MIS, initialized as an empty array. The algorithm examines each k -mer in S_k : for the current k -mer v , it compares with all k -mers in M , and adds v to M if $\text{edit}(u, v) > d$ for every $u \in M$. See the pseudocode given below.

Algorithm 1 Simple Greedy Algorithm

```

initialize an empty array  $M$ 
for each  $k$ -mer  $v \in S_k$  do
     $isMapped \leftarrow \text{false}$ 
    for each  $k$ -mer  $u \in M$  do
        if  $\text{edit}(u, v) \leq d$  then
             $isMapped \leftarrow \text{true}$ 
            break
        end if
    end for
    if  $isMapped == \text{false}$  then
        add  $v$  into  $M$ 
    end if
end for
return  $M$ 

```

We now show that this algorithm is correct, i.e., the returned M is indeed an MIS. According to the algorithm, k -mer v is added to M only if $\text{edit}(u, v) > d$ for all $u \in M$. This means that any two k -mers in M have an edit distance larger than d , i.e., M is independent. On the other hand, for any k -mer v not included in M , the algorithm guarantees that there exists a k -mer u in M such that $\text{edit}(u, v) \leq d$; this implies that M is maximal.

The above algorithm runs in $O(|M| \cdot |\Sigma|^k \cdot d \cdot k)$ time, as in the worst case it compares each k -mer in S_k with all k -mers in M , and determining if $\text{edit}(u, v) \leq d$ for two k -mers u and v takes $O(d \cdot k)$ time. The running time is output-sensitive. It is in favor of instances with small MIS.

2.2 An Improved Greedy Algorithm

We say a k -mer v is *mapped* to a k -mer u in the (partially) constructed MIS M if $\text{edit}(u, v) \leq d$. Such a mapping explains why v is not selected into the MIS. Algorithm 1 essentially finds a mapping for each k -mer $v \in S_k \setminus M$ using an iterative, exhaustive search. As the size of an MIS increases, this can significantly impair the performance of the algorithm. Observe that the search order matters: if a mapping of a k -mer is found early, all the following comparisons can be avoided. Furthermore, based on the results of previous comparisons, it can be inferred that some k -mers in the MIS are close to v while some others are too far away to serve as mappings. We design two techniques that allow us to quickly determine if v can be mapped to some k -mer in M , by utilizing the locality-properties, and to quickly filter out those k -mers in M that v cannot be mapped to, by estimating the bounds on the edit distance. These two techniques serve as extensions to the first algorithm to avoid unnecessary comparisons.

The first technique is based on the observation that a k -mer is likely to have a shared mapping with its “neighboring” k -mers. For a k -mer v , we define its *nearest neighbors* to be the subset of S_k in which each k -mer has edit distance 1 from v , i.e., a substitution. For any given k -mer, clearly, its nearest neighbor set contains $k(|\Sigma| - 1)$ k -mers. For example, the set of nearest neighbors of the 3-mer AAA is {AAC, AAG, AAT, ACA, AGA, ATA, CAA, GAA, TAA}. If AAA is selected into the MIS, all the above 9 k -mers will have it as a shared mapping. This locality-property suggests a modification to Algorithm 1: store the mapping information of k -mers (once found) so that potential shared mappings can be examined first. Note that a k -mer can have multiple valid mappings in the MIS, but we only keep record of one to save both time and space. When checking whether a k -mer v should be added to the MIS or not, we first compute the nearest neighbors of v and check if v shares a mapping with them. If a shared mapping is found, we store this mapping information for v and directly conclude that v cannot be included in the MIS.

The second technique uses the estimated bounds of the edit distance. Recall that (S_k, edit) is a metric space, in particular, the triangle inequality holds. Let σ^k be the k -mer that is purely composed of the character σ . For any two k -mers $u, v \in S_k$ and $\sigma \in \Sigma$, we have

$$\text{edit}(u, v) \geq \left| \text{edit}(u, \sigma^k) - \text{edit}(\sigma^k, v) \right|, \quad (1)$$

$$\text{edit}(u, v) \leq \text{edit}(u, \sigma^k) + \text{edit}(\sigma^k, v). \quad (2)$$

Note that the calculation of $\text{edit}(u, \sigma^k)$ is simple: all characters in u that are not σ should be substituted by σ (insertions and deletions cannot reduce the number of edits needed). Hence, the upper and lower bounds provided by inequalities (1) and (2) can be used as a filter before performing the expensive calculation of $\text{edit}(u, v)$. Using this idea, we modify the algorithm as follows. For each k -mer u in the constructed MIS, we store additional values $\text{edit}(u, \sigma^k)$ for each $\sigma \in \Sigma$. When searching for a mapping of a k -mer v , for each k -mer u in the MIS, we check if the parameter d is within the range given by inequalities (1) and (2). If so, we calculate the exact distance $\text{edit}(u, v)$ and compare it with d ; otherwise, if d is less than $\max_{\sigma \in \Sigma} \left\{ \left| \text{edit}(u, \sigma^k) - \text{edit}(\sigma^k, v) \right| \right\}$, we directly conclude that $\text{edit}(u, v) > d$ and therefore u cannot be a valid mapping of v ; if d is greater than $\min_{\sigma \in \Sigma} \left\{ \text{edit}(u, \sigma^k) + \text{edit}(\sigma^k, v) \right\}$, then we know that $\text{edit}(u, v) < d$ and v cannot be included in the MIS.

Algorithm 2 incorporates above two techniques. Note that Algorithm 2 is also correct, i.e., the returned M is guaranteed an MIS, following the correctness of Algorithm 1 and above analysis. The worst-case running time of Algorithm 2 is the same with Algorithm 1 but it runs faster in practice (see Section 3).

Algorithm 2 Improved Greedy Algorithm

```

initialize empty arrays  $M$ ,  $mapping$ , and  $d\sigma // d\sigma[u]$  is later used
to store  $\text{edit}(u, \sigma^k)$ , for each  $\sigma \in \Sigma$ 
for each  $k$ -mer  $v \in S_k$  do
     $isMapped \leftarrow \text{false}$ 
    for each nearest neighbor  $u$  of  $v$  do
        if  $\text{edit}(mapping[u], v) \leq d$  then
             $isMapped \leftarrow \text{true}$ 
             $mapping[v] \leftarrow mapping[u]$ 
            break
        end if
    end for
    if  $isMapped == \text{true}$  then
        continue
    end if
     $v_\sigma \leftarrow \text{edit}(v, \sigma^k), \forall \sigma \in \Sigma$ 
    for each  $k$ -mer  $u \in M$  do
        if  $\max_{\sigma \in \Sigma} \{ |v_\sigma - d\sigma[u]| \} > d$  then
            continue
        else if  $\min_{\sigma \in \Sigma} \{ v_\sigma + d\sigma[u] \} \leq d$  or  $\text{edit}(u, v) \leq d$  then
             $mapping[v] \leftarrow u$ 
             $isMapped \leftarrow \text{true}$ 
            break
        end if
    end for
    if  $isMapped == \text{false}$  then
        add  $v$  into  $M$ 
         $mapping[v] \leftarrow v$ 
         $d\sigma[v] \leftarrow v_\sigma, \forall \sigma \in \Sigma$ 
    end if
end for
output  $M$ 

```

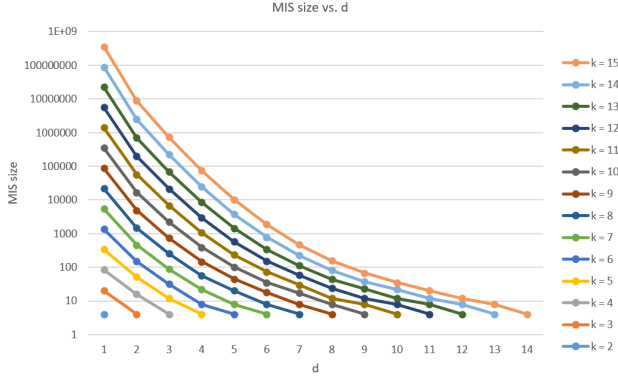


Figure 1: The size of an MIS (found by the fastest algorithm) with respect to parameters k and d .

the growth of d . As discussed before, d specifies the sparsity of the resulting MIS. With larger d value, fewer vertices can be selected into the MIS.

Table 2 reports the fastest algorithm to calculate the MIS for each combination of k and d . Table 2 together with Table 1 suggests a general strategy to choose the best algorithm for a specific combination of k and d . Algorithm 1 is efficient for d values within the interval $[k - 4, k - 1]$ since these d values result in an MIS with less than 15 vertices approximately. The BFS-based Algorithm (Algorithm 3) is the best for d values between 1 and 4 because such a small d leads to an MIS that is too large to perform pairwise comparisons efficiently. The rest of the d values are better be handled by Algorithm 2.

Table 3 and Table 4 record the corresponding time and memory usage, respectively. The rows of Table 3 show that the running time generally increases geometrically with respect to the parameter k . The current computation bottleneck is the case with $k = 15$ and $d = 5$, which takes approximately 15 hours using Algorithm 2.

Table 2: The fastest algorithm for each combination of k and d . Numbers 1, 2, and 3 correspond to Algorithms 1, 2, and 3.

		k													
		2	3	4	5	6	7	8	9	10	11	12	13	14	15
d	1	1	1	1	1	1	3	3	3	3	3	3	3	3	3
	2		1	1	1	1	1	3	3	3	3	3	3	3	3
	3			1	1	1	1	2	3	3	3	3	3	3	3
	4				1	1	1	1	2	2	2	2	3	3	3
	5					1	1	1	2	2	2	2	2	2	2
	6						1	1	1	1	2	2	2	2	2
	7							1	1	1	1	2	2	2	2
	8								1	1	1	1	1	2	2
	9									1	1	1	1	2	2
	10										1	1	1	1	2
	11											1	1	1	1
	12												1	1	1
	13													1	1
	14														1

Table 3: Running time (seconds) of the fastest algorithm for each (k, d) .

		k														
		2	3	4	5	6	7	8	9	10	11	12	13	14	15	
d	1	0	0	0	0	0	0	0	1	5	25	114	528	3382	14028	
	2		0	0	0	0	0	0	2	11	46	202	977	5444	22380	
	3			0	0	0	0	0	3	18	78	344	2161	10317	46701	
	4				0	0	0	0	1	8	55	460	4191	21801	36614	
	5					0	0	0	0	4	25	152	1409	11887	53478	
	6						0	0	0	3	17	94	659	4423	23922	
	7							0	0	1	13	77	504	3008	14864	
	8								0	1	7	52	438	2422	11308	
	9									0	5	31	258	2128	10003	
	10										4	23	210	1809	9197	
	11											21	187	1225	7322	
	12												184	988	5548	
	13													931	4743	
	14														4332	

Table 4: Memory usage (kB) of the fastest algorithm for each (k, d) .

		k														
		2	3	4	5	6	7	8	9	10	11	12	13	14	15	
d	1	1076	1076	1076	1076	1336	1332	1328	1324	3648	4568	8524	23916	85316	331164	
	2		1076	1076	1076	1076	1076	1324	1320	3544	4596	8364	24012	85396	331220	
	3			1076	1076	1076	1072	1476	1532	4008	5148	8888	24336	85984	331908	
	4				1076	1076	1076	1076	1624	5460	11508	36224	29512	92524	341752	
	5					1076	1072	1076	1648	5352	11468	36016	134416	527728	2100852	
	6						1072	1076	1072	1952	11584	36272	134484	527672	2100688	
	7							1076	1076	1880	2012	36164	134480	527688	2100568	
	8								1072	1920	1868	1952	1920	527772	2100568	
	9									1980	2012	3464	1920	527760	2100548	
	10										1980	1868	3456	1964	2100636	
	11											2032	3396	3184	3308	
	12												1976	3380	1992	
	13													3404	1964	
	14														3384	

Algorithm 2 always has the largest memory overhead comparing with other two. The extra memory is mainly used to store the mapping information of all k -mers. The peak memory usage is approximately 2 GB for $k = 15$ and $d \in \{5, 6, \dots, 10\}$.

4 CONCLUSION AND DISCUSSION

We studied the problem of extracting an MIS as a representative substructure from a k -mer space. Three algorithms are designed to efficiently solve the problem for different ranges of parameters k and d . The first one is a simple greedy algorithm similar to the greedy graph for general graphs. The second algorithm extends the first by implementing two techniques to avoid some redundant comparisons. The third algorithm represents the edit distance as a shortest path in an extended graph and uses a variant of BFS. Experiments are done for k up to 15. The computation bottleneck occurs at $k = 15$ and $d = 5$ where the second algorithm performs the best. The corresponding peak running time is approximately 15 hours, and the peak memory usage is about 2 GB.

For future work, we would like to extend this study to larger k -mer spaces. Considering the current computation bottleneck, one

potential improvement is to design more heuristics to partition the k -mer space so that only a small subset of k -mers will be involved in the expensive pairwise edit distance computation.

Another idea is to revise the BFS-based algorithm. The extended graph with both k -mers and $(k - 1)$ -mers is highly symmetric with respect to permutations of the alphabet. For example, the search tree for the 5-mers AGAAC and TATTG are isomorphic where A, G, and C are replaced with T, A, and G, respectively. This algorithm may take advantage of such symmetries to explore the graph more efficiently. Identifying and utilizing symmetric substructures of the k -mer space is of independent interest and has applications in other related fields [23].

Regarding the practical applications of the constructed MISs, one potentiality is their usage in k -mer-based reads clustering problems [10, 11], where the choice of centers is crucial. Because an MIS contains a group of representatives for the k -mer space that are guaranteed to be a certain edit distance apart, it is ideal for generating even-sized clusters. Moreover, the MISs can also be used to generate keys for k -mer-based reads hashing problems [19, 21] where nearby k -mers are desired to share a hash value (i.e., locality-sensitive hashing), which is particularly useful in aligning error-prone long reads—a challenging yet unsolved problem. Last, it is interesting to apply the constructed MISs as the set of representative k -mers in sketching large-scale sequences.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. This work is supported by the US National Science Foundation (2019797 and 2145171 to M.S.) and by the US National Institutes of Health (R01HG011065 to M.S.).

REFERENCES

- [1] C. Titus Brown and Luiz Irber. 2016. sourmash: a library for MinHash sketching of DNA. *Journal of Open Source Software* 1, 5 (2016), 27. <https://doi.org/10.21105/joss.00027>
- [2] Martin Burtcher, Sindhu Devale, Sahar Azimi, Jayadharini Jaiganesh, and Evan Powers. 2018. A High-Quality and Fast Maximal Independent Set Implementation for GPUs. *ACM Trans. Parallel Comput.* 5, 2, Article 8 (dec 2018), 27 pages.
- [3] Andrea Califano and Isidore Rigoutsos. 1993. FLASH: A fast look-up algorithm for string homology. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 353–359. <https://doi.org/10.1109/CVPR.1993.341106>
- [4] Lijun Chang, Wei Li, and Wenjie Zhang. 2017. Computing A Near-Maximum Independent Set in Linear Time by Reducing-Peeling. In *Proceedings of the 2017 ACM International Conference on Management of Data (Chicago, Illinois, USA) (SIGMOD '17)*. Association for Computing Machinery, New York, NY, USA, 1181–1196.
- [5] Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. 2013. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* 29, 1 (2013), 15–21.
- [6] Xiangyu Gao, Jianzhong Li, and Dongjing Miao. 2022. Dynamic Approximate Maximum Independent Set on Massive Graphs. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 1835–1847.
- [7] Wayne Goddard and Michael A Henning. 2013. Independent domination in graphs: A survey and recent results. *Discrete Mathematics* 313, 7 (2013), 839–854.
- [8] Luiz Irber, Phillip T. Brooks, Taylor Reiter, N. Tessa Pierce-Ward, Mahmudur Rahman Hera, David Koslicki, and C. Titus Brown. 2022. Lightweight compositional analysis of metagenomes with FracMinHash and minimum metagenome covers. *bioRxiv* (2022). <https://doi.org/10.1101/2022.01.11.475838>
- [9] Miten Jain, Sergey Koren, Karen H Miga, Josh Quick, Arthur C Rand, Thomas A Sasani, John R Tyson, Andrew D Beggs, Alexander T Dilthey, Ian T Fiddes, et al. 2018. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature Biotechnology* 36, 4 (2018), 338–345.
- [10] Chang Sik Kim, Martyn D Winn, Vipin Sachdeva, and Kirk E Jordan. 2017. k -mer clustering algorithm using a MapReduce framework: application to the parallelization of the Inchworm module of Trinity. *BMC Bioinformatics* 18, 1 (2017), 1–15. <https://doi.org/10.1186/s12859-017-1881-8>
- [11] Jessime M Kirk, Susan O Kim, Kaoru Inoue, Matthew J Smola, David M Lee, Megan D Schertzer, Joshua S Wooten, Allison R Baker, Daniel Sprague, et al. 2018. Functional classification of long non-coding RNAs by k -mer content. *Nature Genetics* 50, 10 (2018), 1474–1482. <https://doi.org/10.1038/s41588-018-0207-8>
- [12] Robert Krauthgamer and Yuval Rabani. 2009. Improved lower bounds for embeddings into L_1 . *SIAM J. Comput.* 38, 6 (2009), 2487–2498.
- [13] Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F Werneck. 2016. Finding near-optimal independent sets at scale. In *2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 138–150.
- [14] Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, Vol. 10, no. 8. Soviet Union, 707–710.
- [15] Heng Li. 2018. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* 34, 18 (2018), 3094–3100.
- [16] Yu Liu, Jiaheng Lu, Hua Yang, Xiaokui Xiao, and Zhewei Wei. 2015. Towards Maximum Independent Sets on Massive Graphs. *Proc. VLDB Endow* 8, 13 (sep 2015), 2122–2133.
- [17] Bin Ma, John Tromp, and Ming Li. 2002. PatternHunter: faster and more sensitive homology search. *Bioinformatics* 18, 3 (2002), 440–445.
- [18] Denise Mak, Yevgeniy Gelfand, and Gary Benson. 2006. Indel seeds for homology search. *Bioinformatics* 22, 14 (2006), e341–e349.
- [19] Hamid Mohamadi, Justin Chu, Benjamin P Vandervalk, and Inanc Birol. 2016. ntHash: recursive nucleotide hashing. *Bioinformatics* 32, 22 (2016), 3492–3494. <https://doi.org/10.1093/bioinformatics/btw397>
- [20] Rafail Ostrovsky and Yuval Rabani. 2007. Low distortion embeddings for edit distance. *Journal of the ACM (JACM)* 54, 5 (2007), 23–es.
- [21] Enrico Petrucci, Laurent Noé, Cinzia Pizzi, and Matteo Comin. 2020. Iterative spaced seed hashing: closing the gap between spaced seed hashing and k -mer hashing. *Journal of Computational Biology* 27, 2 (2020), 223–233. <https://doi.org/10.1089/cmb.2019.0298>
- [22] Anthony Rhoads and Kin Fai Au. 2015. PacBio sequencing and its applications. *Genomics, Proteomics & Bioinformatics* 13, 5 (2015), 278–289.
- [23] Klaus U Schulz and Stoyan Mihov. 2002. Fast string correction with Levenshtein automata. *International Journal on Document Analysis and Recognition* 5, 1 (2002), 67–85. <https://doi.org/10.1007/s10032-002-0082-8>