Gazelle: Transcript abundance query against large-scale RNA-seq experiments

Xiaofei Zhang
Ye Yu
xiaofei.zhang@uky.edu
ye.yu@uky.edu
Department of Computer Science,
University of Kentucky
Lexington, Kentucky, USA

Chan Hee Mok
James N. MacLeod
chanhee.mok@uky.edu
jnmacleod@uky.edu
Department of Veterinary Science,
University of Kentucky
Lexington, Kentucky, USA

Jinze Liu
jinze.liu@vcuhealth.org
Department of Biostatistics, Virginia
Commonwealth University
Richmond, Virginia, USA

ABSTRACT

The exponential growth of high throughput sequencing data has been witnessed in almost every sequencing data repository. To date, most of the exploratory analysis on these large datasets requires heavy lifting data processing pipelines that are both resource and labor intensive.

Very recently, various algorithms have been developed to enable arbitrary sequence query over large collections of sequencing data. These algorithms were designed to support presence/absence query, i.e., screening for RNA-seq samples containing a given transcript sequence. Their utility is rather limited as they cannot retrieve abundance information of query sequence. Such abundance information is indeed critical in real applications in order to understand how the variation of transcript expression associates with different biological conditions or disease subtypes.

In this paper, we present Gazelle, a sequence query engine that enables fast and quantified query against large-scale RNA-seq experiments. Gazelle exploits the advantages of two different types of hashing algorithms and seamlessly combines them into one integrated structure to support highly efficient and accurate sequence queries with abundance. We evaluated the performance of Gazelle on three datasets to benchmark its efficiency, accuracy as well as its utility in real-life applications. Our result shows that Gazelle achieves near-perfect k -mer query, supports on-demand sequence query against moderately large sequence database, and renders highly consistent abundance estimation with RT-qPCR as well as traditional transcript quantification method such as Kallisto.

CCS CONCEPTS

• **Applied computing** → **Bioinformatics**.

KEYWORDS

transcript query, RNA-seq, indexing

ACM Reference Format:

Xiaofei Zhang, Ye Yu, Chan Hee Mok, James N. MacLeod, and Jinze Liu. 2021. Gazelle: Transcript abundance query against large-scale RNA-seq experiments. In *12th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '21)*, August 1–4, 2021, Gainesville, FL, USA. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3459930.3469548>

1 INTRODUCTION

The advancement of high throughput sequencing technologies has revolutionized genomic research in the past decade. An enormous amount of sequencing studies have been carried out in almost all disciplines of biological and biomedical research utilizing either RNA-seq and whole genome sequencing technologies. This is evidenced by rapid growth of sequencing data deposited in public archives such as Sequence Read Archive (SRA)[17]. To date, SRA holds about 16.2 petabytes of raw RNA-seq data, among which 9.2 petabytes is openly accessible to the public[27]. Large consortiums such as the Cancer Genome Atlas (TCGA) project [31] and International Cancer Genome Consortium (ICGC) [36] have conducted a large scale sequencing experiments in order to identify populational-level biomarkers in a wide range of tumor types. The availability of these large sequencing data is transforming existing research from focusing on individual or small sets of experiments into global screening of archived sequencing data. For example, Pan-cancer studies have successfully identified the connections and differences between the variety of tumor types which cannot be achieved by studying tumor types individually [23, 32].

As a result, many special-purpose databases have been built to store the intermediate and summarized results from these population studies. The cBioportal [10, 12] is an open-source platform that allows users to interactively explore, visualize and analyze multi-dimensional cancer genomics datasets, such as gene or transcript expression for TCGA datasets. SpliceSeq [26] is a data resource storing percent-splice-in of splicing events from TCGA RNA-seq data. Tumor Fusion Gene Data Portal [33] summarizes the human fusion transcript cross different cancer types in the TCGA dataset. While providing value to the community, these special-purpose databases are often hosted by different research groups and may be inconsistent due to differences in analysis pipelines. Additionally, these databases are often designed to store well-defined genomic features and thus are limited in its utility to serve general inquiry into sequencing experiments. For example, for newly discovered

Permission to make digital or hard copies of all or part 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 components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

BCB '21, August 1–4, 2021, Gainesville, FL, USA

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8450-6/21/08...\$15.00

<https://doi.org/10.1145/3459930.3469548>

fusion junctions or splice junctions, their information may not be included in those well established databases.

In the past few years, efforts have been made to support arbitrary sequence queries directly from sequencing datasets [20]. Sequence Bloom Tree (SBT) and its variants [28–30] were developed to identify sequencing experiments that contain a significant percentage of content in a query sequence. For a given query sequence, SBT returns the subset of the experiments that contain at least θ percent of k -mers in the query sequence, where θ is a user-defined threshold. Mantis [25] provides the same type of outputs as SBT does but without false positives. It uses counting quotient filter's results to speed up the sequence querying time. SeqOthello [35] is an indexing system based on the minimal perfect hash classifier *l*-Othello [34]. It is the first method that supports k -mer level identity search and has achieved significant improvement in sequence query speed over previously mentioned approaches using limited memory and storage.

However, the vast majority of existing sequence query algorithms only support sequence identify search, returning whether or not a query sequence is present in sequencing experiments. They cannot deliver quantitative information in terms of abundance level of the query sequence in each experiment the query is present. The abundance details typically are not included during the index construction process in order to minimize memory and storage costs. However, it is well known that quantitative information can be critical for many applications. For example, a lot of transcripts commonly exist in many tissue types, but they may function differently by dialing up or down their expression levels [15, 24, 26]. Splice switching that cause the change of transcript abundance is also identified as a common biomarker in cancer [5, 9].

In this study, we proposed a novel indexing system, called "Gazelle", that allows quantitative sequence query over large scale sequencing datasets. Gazelle utilizes two different hashing techniques that enable indexing and querying large size RNA-seq datasets requiring computational resources available in common desktops. We benchmarked the performance of Gazelle index using 2,586 RNA-seq samples and compared it with other methods. Using Gazelle we managed to index about 5.0 TB raw reads file to 45.56 GB on a computer with 64 GB memory. The average query time per transcript to get the abundance on the Gazelle index is 0.45 seconds, which is at least 2 orders of magnitude faster than existing available methods such as Jellyfish Count function. To demonstrate the utility of Gazelle in real applications, we constructed an index to support on-demand screening the entire TCGA RNA-seq data, totalling 11095 samples. We used the estimated abundance of transcripts queried from Gazelle to calculate percent-splice-in (PSI) of dozens of alternative splicing events. The results exhibit high consistency with PSIs reported by SpliceSeq, a curated database with data processed with traditional RNA-seq pipelines. Finally, we demonstrate that the estimated transcript abundances are comparable to the expression measured by RT-PCR with a collection of Equine RNA-seq datasets. This suggests that Gazelle enables fast on-demand query of transcript abundance on large scale RNA-seq with high accuracy.

2 METHODS

2.1 The k -mer representation of RNA-seq data

Words are the basic units of natural languages. In genomic sequences, k -mers are often considered as the genomic "words" [13]. A k -mer is a nucleotide sequence of length k . k -mers have been used in many bioinformatics applications including genome assembly [14], transcript quantification [8] and metagenomic classification [19]. For example, the basic components of de Bruijn graph for de novo assembly where a sequence is assembled through the presence of overlapping k -mers. Selecting the size k is crucial. As k increases, the dictionary of k -mers increases exponentially in size, but the chance of a random k -mer being unique to a genomic locus also increases significantly.

Our approach stores RNA-seq data in the form of k -mers and their abundances. The sequence query works by retrieving its constituting k -mers for their abundances individually followed by the aggregation of the retrieved k -mer information as a whole. Given billions of unique k -mers arising from thousands of experiments, how to efficiently store k -mer abundances and how to rapidly retrieve the k -mer information during query become the challenging problems.

2.2 Design rationale

Traditional hashing technique [1] is a popular option to store the key-value pairs to support fast queries. It stores the keys themselves (k -mers) and their corresponding values in a large hash table [16]. During query, the value of the key can be quickly looked up in the table through hashing of the keys. However, such conventional hash table is not suitable to store billions of k -mers and their abundances. To store k -mers with $k = 20$, at least 40 bits are needed to store a k -mer since each base requires 2 bits to store and additional 8 or more bits to store k -mer's abundance. Besides, in order to minimize hashing conflict, the real size of the hash table has to be configured several times bigger than the minimum required for the storage, making it infeasible to hold billions of k -mers in memory on a regular PC.

Minimal perfect hashing function (MPFH) is known as an alternative hashing technique that can store the key-value pairs with minimal storage space. Given a dictionary with n key-value pairs, an MPFH can be constructed to map the n keys to the consecutive integers from 1 to n . As a result, it does not need to store the keys themselves in the hash table. Instead, the values of the corresponding keys can be stored in the order based on the hashed values of the keys. Thus MPFH maintains a static data structure using much less memory. Theoretically, about $1.44n$ bits are needed for MPFH to store n keys regardless of the key type [3]. This is very attractive for storing a large scale set of k -mers.

However, the minimal storage associated with MPFH comes with serious limitations. An MPFH is only accurate for in-set queries, i.e., an MPFH is designed for a predefined set of keys and only guarantees accurate query of these keys. Querying any keys outside the predefined set will return a random result without guarantee accuracy. This is not sufficient for arbitrary sequence queries on RNA-seq data because the query sequence can contain a significant number of alien k -mers that may not be present in the provided database.

To alleviate the issue of false positive query while still taking the advantage of the minimal storage offered by MPFH, we have designed Gazelle in a two-tier fashion. The Tier 1 of Gazelle is responsible for identity search, which determines whether or not a k -mer is present in each individual experiment. The Tier 2 query is abundance search and experiment-specific. Only k -mers present in an experiment will be queried to retrieve its abundance. In this way, we can avoid the query of k -mers absent from the experiment, thus eliminating the possibility of false positive queries.

As the Tier 2 query is now restricted to in-set query, it can be implemented using MPFH with minimal storage without false positive rate. Basically, an individual MPFH is constructed to store abundances of k -mers present in each RNA-seq experiment.

Putting them together, Gazelle offers a reference-free, alignment-free, and parameter-free sequence search system that enables quantified sequence query on large-scale datasets.

2.3 Gazelle construction workflow

The entire structure of the Gazelle index is shown in Figure 1. We next present a step-by-step workflow to construct Gazelle index.

Step 1: Preprocessing. Prior to the construction of the Gazelle index, each RNA-seq sample is preprocessed by Jellyfish [21] to extract all the unique k -mers present in the RNA-seq reads and their counts. Here the count of a k -mer refers to the total number of its

Figure 1: An overview of Gazelle’s 2-tier data structure indexing n RNA-seq experiments. The Tier 1 of Gazelle index supports identity search of k -mers in the query. The search returns the experiments containing each individual k -mers. The Tier 1 index leverages a hierarchical othello-based hashing classifier indexing compressed k -mer occurrence map of all n experiments. The Tier 2 of Gazelle index provides the abundance search for k -mers in individual experiments. The Tier 2 index includes n BBhashes, each of which stores k -mers and their abundances pertaining to its own RNA-seq experiment.

occurrences in all RNA-seq reads sequenced in the sample. The sets of k -mers derived from each RNA-seq experiment are combined to establish a SeqOthello index in Tier 1, while the k -mers and their counts in each RNA-seq will be used to establish a quantitative index in Tier 2.

Step 2: Tier-1 index construction. The Tier 1 in Gazelle provides k -mer identity search by storing a global map of all k -mers and their occurrences across all indexed sequencing experiments. Formally, we refer to the collection of n RNA-seq samples as $(E_1, E_2, \dots, E_n)$. Let $\langle S_i, V_i \rangle, 0 \leq i < n$ be the pair of the set of k -mers and their counts for the i th sample. Let S be the union of the k -mer sets $\bigcup_{0 \leq i \leq n} S_i$ from all n RNA-seq experiments. The Tier 1 Gazelle index provides a mapping between any k -mers in S to its individual occurrence map across all the n RNA-seq samples.

The construction algorithm of Tier 1 index of Gazelle is adapted from SeqOthello [35]. The storage requirement of the entire k -mer occurrence map in a matrix format may quickly exceed the memory capacity of regular computers, making it impossible to load the entire occurrence map into the memory for query. SeqOthello tackles this challenge using two approaches. It first tries to compress the k -mer’s occurrence map using encoding methods that are tailored to the different sparsity levels of k -mer occurrence as shown in Figure 1. On top of this, SeqOthello divides the entire k -mer occurrence map into smaller buckets, drastically lowers the limit of run-time memory requirement. SeqOthello nevertheless maintains extremely fast query time due to the design of a hierarchical index enabled by a hashing classifier.

Step 3: Tier-2 index construction. The Tier 2 index of Gazelle supports abundance search using a collection of MPFH, each of which stores the abundance levels of an individual experiment. In each experiment-specific MPFH, it stores the mapping between the set of k -mers present in the experiment to the corresponding abundance.

There are several MPFH implementations available using different algorithms including compressed hash-and-displace algorithm [6], random hypergraph peeling based algorithms such as EMPHF [4], HEM [7] and Cache-Oblivious MPFH [2], and BBhash [18] that is based on a successive hashing based algorithm.

The specific MPFH we adopted in our implementation is BBhash [18]. The advantage of BBhash is its very economic storage cost as well as rapid construction and query speed. The query on BBhash is limited to inset query as only k -mers that are determined to be present in a sequencing experiment by Tier 1. The gamma factor (γ) in BBhash, a parameter that sets the bit array size during the successive hashing was retained to allow users use larger storage for fast construction.

Thus, Tier 2 index of Gazelle consists of a set of BBhash indices $\{B_1, B_2, \dots, B_n\}$, where B_i stores the k -mer and counts pairs $\langle S_i, V_i \rangle$ in the i th RNA-seq experiment. Each BBhash maps the set of k -mers in S_i to a consecutive integer set $[0, |S_i|)$, each of which points to an address that stores the corresponding k -mer count in V_i or its derivatives.

Storing the raw k -mer counts for hundreds of millions of k -mers can be storage consuming as each k -mer needs at least 16 bits (short) or even 32 bits (regular). Biologically, storing exact raw read count may not be necessary for general assessment of gene expression levels as significant changes are often evidenced by double log

fold changes. In Gazelle, we use a logarithm binning method to discretize the raw k -mer counts into a predefined set of levels. This allows users to store only the gene expression levels using much less memory. For example, one can store 2^n k -mer expression levels using n bits of memory per k -mer by using the binning function $D(x)$ shown in the following equation:

$$D(x) = \begin{cases} \lfloor \log_2(x - t) \rfloor & \text{if } x - t < 2^n \\ 2^n - 1 & \text{if } x - t \geq 2^n \end{cases} \quad (1)$$

where the x is the original k -mer count and t is the minimum abundance threshold required to filter out noisy k -mers.

In the default setting, Gazelle takes the input of the original k -mer counts derived from Jellyfish. However, different pre-processing steps may be adopted to cater to different datasets or research focuses. For example, to ensure that the expression levels of k -mers from one sample can be directly compared to those from the other samples, one may store the k -mer abundance normalized by the total reads counts of each sample. Additionally, users may also choose different discretization algorithms so that the binned expression levels reflect biologically meaningful expression ranges.

2.4 Query on a Gazelle index

To query a sequence against a Gazelle index, the sequence is first broken down to k -mers it contains. Then each k -mer's occurrence map is obtained by querying the Tier 1 index. After the occurrence maps of all k -mers are collected, the abundance maps of the k -mers are obtained by querying the k -mers against the Gazelle Tier 2 index. During the query of each experiment, only k -mers that are present according to the occurrence maps are queried. The absent k -mers' abundances are set to 0s. The reverse binning function will be applied to recover an approximate abundance value if any binning method is used when building the Gazelle index.

When a significant amount of k -mers are not present in an experiment, it is a signal that the query sequence may not be expressed in the experiment. In this case, it is unnecessary to query Tier 2 index to identify the abundance of the sparsely present k -mers. Thus, we have set up a threshold called k -mer hit ratio. The threshold will be applied after the Tier 1 query. The abundance of a sequence within an experiment will be extracted at Tier 2 only if the fraction of present k -mers in the query exceeds the minimum k -mer hit rate in the experiment. This threshold significantly reduces the unnecessary tier-2 queries especially when we expect the majority of k -mers to be present in real applications, such as the transcript query.

2.5 Expression level estimation

The expression level of the query sequence in each experiment can be estimated using the abundance map from Gazelle query results. As mentioned above, the expression level of a query sequence in an experiment is only meaningful if the sequence is present in the experiment. This is indicated by the presence of significant fraction of k -mers in the experiment. Inspired by how Reads Per Kilobase Million (RPKM) was calculated as gene expression in RNA-seq [22], we use the average of the k -mer counts of all k -mers in a query sequence to estimate the expression of that sequence. However, the k -mers contained in the query sequence may not be exclusive to query sequence, potentially resulting in outlier k -mers with much

higher abundance than the other k -mers. Instead of using the mean of all the k -mer counts, we apply the interquartile mean (IQM) of the k -mer counts for the expression estimation. The IQM considers the k -mer counts within the interquartile range of the distribution of k -mer counts, thus removing the effect of the outlier k -mers who may not exclusive for the query sequence and have counts significantly larger than others.

2.6 Complexity of Gazelle

The time complexity to construct the Gazelle Tier 1 map is $O(|S|)$, where S is the set of unique k -mers across all the experiments. The time complexity of Tier 2 index construction for experiment E_i is $O(e^{\frac{1}{\gamma}} * |S_i|)$, where S_i is the set of k -mers in experiment E_i . In practice, $\gamma = 1$ is used since it has the lowest disk space cost.

To query a sequence with m k -mers in the Tier 1 search, Gazelle computes at most two hash values and accesses two memory locations for each k -mer in the query. The time complexity is $O(m)$. In the Tier 2 search, a search on one k -mer in one BHash is $O(1)$. The query of entire sequence across all n experiments is $O(mn)$.

3 RESULTS

In this section we first evaluate the accuracy of k -mer queries from Gazelle index. This is followed by benchmarking Gazelle's scalability in index construction and sequence query. We further demonstrate Gazelle's performance in two real applications where it is used to extract transcript abundances from a large set of RNA-seq experiments.

All experiments were conducted on a Linux machine (Ubuntu 18.04) with an Intel(R) Core(TM) i7-7700 CPU (@3.60GHz with 8 cores and 8MB L3 cache) and 64GB RAM.

3.1 Performance Evaluation of Gazelle on 2,586 SRA Sequencing Experiments

Our first set of experiments were conducted using 2,586 RNA-seq experiments from human blood, breast, and brain tissues deposited at SRA[17]. Originally, the dataset had 2,652 samples and was firstly introduced and used by research work of Solomon and Kingsford[28]. Among those 66 samples were removed since their read length is shorter than 21. This dataset provides a diverse set of samples in terms of tissue types and sequencing depth. It has also been used in numerous existing studies to benchmark the performance of sequence query tools [25, 28, 35].

3.1.1 Construction of Gazelle index. We followed the steps described in method section 2.3 to construct the Gazelle index. The *count* function in the Jellyfish [21] was called to generate the k -mer occurrence and abundance files from raw sequencing reads in each RNA-seq experiment, taking about 2 minutes per sample. The Gazelle index was constructed from the k -mer files generated in the first step.

The total size of the Gazelle index of the 2,586 RNA-seq experiments is 45.58 GB, which is less than 1% of size of the compressed raw FASTQ read files (5.0 TB) and 15% of the size of the raw k -mer files generated by Jellyfish (300 GB). The abundance was compressed using 4-bit encoding of k -mer expression levels. Using 2-bit encoding further decreases the total index size to 36.65 GB. The

Table 1: Comparison of k -mer query accuracy between SeqOthello, BBhash and Gazelle

Index Algorithm	Query Type	BBhash Gamma Factor	False Positive Rate(%)
SeqOthello	presence	-	0.018
Gazelle	abundance	1	0.018
Gazelle	abundance	2	0.018
Gazelle	abundance	5	0.015
Gazelle	abundance	10	0.009
BBhash	abundance	1	99.97
BBhash	abundance	2	99.44
BBhash	abundance	5	79.25
BBhash	abundance	10	47.98

construction of Gazelle index took around 9.23 hours with the input of individual k -mer files. The peak memory of the construction process is about 29 GB, well under 32GB that is available in many desktop computers.

3.1.2 Gazelle achieves near perfect k -mer query. In this experiment, we investigated to what extent we can accurately report the presence and abundance of k -mers. The entire set of k -mers from each RNA-seq experiment generated by Jellyfish served as ground truth. We built four different Gazelle indices with different gamma factors (1,2,5,10) of BBhash in Tier 2 to examine the effects of gamma on the accuracy of query result. We focused the evaluation on the query results of 150 RNA-seq experiments randomly selected out of the 2,586 RNA-seq experiment and compared them with.

To set up the query experiment, we extracted the entire set of unique k -mers totalling 124,920,978 from Gencode(v29) transcripts annotated for human transcriptome [11]. These k -mers were queried against the Gazelle index for their presence across all RNA-seq experiments.

A k -mer query is false positive if it returns a positive presence of the k -mer in an experiment which is actually absent.

We compared three methods SeqOthello, BBhash, as well as Gazelle. The result was summarized in Table 1. The result shows that both SeqOthello and Gazelle achieve very low false positive rate, i.e., less than 2 out of 10,000 k -mer queries. But using BBhash alone results in a false positive rate as high as 47.98% even when the gamma factor increases to 10. This is not quite suitable for general query requirements. This validates our design rationale of Gazelle, where BBhash is utilized at the second level only to support query abundances of k -mers that are known to be present in an experiment. We also expect that the false positive rate at transcript level is even lower. For a transcript sequence to be present in an experiment, a significant fraction of its k -mers should be present. Given the extremely low false positive rate of Gazelle k -mer query, the chance of a significant amount of k -mers in a query returning false positive results should be very low. In fact, it is expected to be much lower than the false positive rate of individual k -mer query.

3.1.3 Gazelle enables fast quantified sequence query. In this experiment, we evaluated the query performance of Gazelle on the 2586

RNA-seq experiments. We compared the performance of three approaches, Gazelle, SeqOthello, and Jellyfish’s ‘Query’ functionality.

As with any other index structures, the efficiency of Gazelle depends on the amount of disk access. The fastest version of Gazelle is to load all indices into memory prior to query. The disadvantage is that this will incur a one-time loading cost and higher peak memory. Note that during the query, a sequence may not be present in all experiments. This can be determined at the end of Tier 1 index. Thus we may only need to load the indices of experiments where the query sequence are present. This avoids the pre-loading of all the Tier 2 indices into memory, thus minimizing the peak memory required during query. This approach strikes a good balance between memory and query efficiency. We call the first as Gazelle pre-loading approach and the latter as Gazelle active-loading approach.

3.2 Gazelle enables quantitative transcript expression screening

3.2.1 Expression level retrieved by Gazelle is comparable to other methods. To evaluate the query performance of Gazelle under different configurations, three sets of query sequences each containing 10,000 transcripts were randomly selected from the Gencode(v29) transcript database (Frankish et al., 2019). These sequences were queried against the Gazelle index built upon the 2,586 RNA-seq samples with 4-bits encoding in the Tier 2 index. We also evaluated the effect of hit rate threshold for the query performance. The hit rate defines the fraction of k -mers of the query sequence that are present in an experiment. If the hit rate is higher than the cutoff threshold, the sequence is deemed to be present in an experiment. This will trigger the query against the Tier 2 index to obtain the abundance information. The SeqOthello was compared as an presence/absence sequence query tool. The same sets of transcripts were also queried against the SeqOthello index. As a typical method without indexing, Jellyfish’s ‘count’ functionality was also compared. However, due to the extreme long query time, only one randomly selected sequence of 1,584 bases (the average size of query sequences used to evaluate Gazelle and SeqOthello) were queried using Jellyfish on 2,586 RNA-seq samples.

The average query time of three query sequence sets using different hit rate cutoff and index loading options were collected and summarized in Table 2. With the pre-loading option, Gazelle achieves an average of 0.42 second per transcript query. The amount of query time only increases slightly when lowering the hit rate cutoffs. This is because all indices are in memory and accessing more experiments is not quite expensive. The average loading time of the whole index is about 330 seconds, adding the average of 0.033 second for the average time cost by each query. The peak memory usage is 46.75 GB in this case. In contrast, when the active loading option is selected, Gazelle uses less than 29 GB memory during the entire experiment. The average query time per transcript is between 0.37 and 0.68 second with higher cutoff rendering shorter query response time. Nevertheless, the query time is not significantly higher especially with a reasonable hit rate cutoff which is 50% and above. The peak memory in this case is 28.5GB, significantly less than the peak memory required with the preloading approach.

Table 2: Comparison of query response time per transcript (seconds) under different hit rate cutoff

Method	Hit rate cutoffs used in Gazelle					
	1	0.9	0.75	0.5	0.25	0
Gazelle-pre loading	0.37±0.0014	0.38±0.0027	0.39±0.0036	0.41±0.0020	0.42±0.0036	0.45±0.0035
Gazelle-active loading	0.37±0.0065	0.41±0.0035	0.47±0.0084	0.56±0.0073	0.68±0.0123	-
SeqOthello (detailed) ¹	-	-	-	-	-	0.06±0.0002
Jellyfish	-	-	-	-	-	415.76

¹ The 'detailed' option in SeqOthello outputs the presence/absence information of individual k -mers across all the indexed experiments.

Figure 2: Correlations of expressions of 90 genes from 28 horse samples obtained between Gazelle and Kallisto (a), Gazelle and rt-PCR (b), Kallisto and rt-PCR (c).

Our result shows that Gazelle pre-loading is a faster option if the memory size is large enough to hold the entire gazelle index. However, Gazelle active-loading offers very competitive query response time while cutting down the memory requirement to half. The query time of Gazelle is about 7 times larger compared to the presence/absence query conducted by SeqOthello because of more computations are required to obtain abundance information from individual experiments. However it is at least 2 orders of magnitude faster than existing available methods such as Jellyfish Count function.

In this experiment, we evaluate how well the Gazelle query estimation of transcript abundance compares to the gold standard measurement obtained by reverse transcription-quantitative polymerase chain reaction (RT-qPCR) and transcript quantification tool, Kallisto.

We obtained RNA-seq data from 28 different equine tissue and cell culture samples. The expression of 90 targeted gene loci of those samples were quantified by RT-qPCR, a traditional method for gene expression quantification. Meanwhile, the RNA-seq data of those 28 equine samples were also generated. In the meantime, the sequences of the amplicons of the 90 target genes were used as target sequences to quantify their expression by applying Kallisto on raw RNA-seq reads. A Gazelle index was built on those 28 RNA-seq samples using 4-bits for expression encoding. The same RT-qPCR target

sequences were used to query Gazelle to retrieve the corresponding RNA abundances. The expressions obtained by Gazelle were compared to those quantified using Kallisto on the RNA-seq data, and also the measurements obtained independently from RT-qPCR.

As shown in Figure 2, there exists a strong correlation between the expression level estimated by Gazelle and Kallisto as high as 0.98. The correlation between Gazelle and rt-PCR is also very competitive to the correlation between Kallisto and rt-PCR, both over 0.88 in pearson correlation. This is reasonably high as RNA-seq and rt-PCR are two independent experimental approaches. These results show that even with compression when storing k -mer expression, Gazelle is able to rapidly and accurately query transcript abundances from RNA-seq.

3.2.2 Percent-splice-in level retrieve based on sequence query results.

In this experiment, Gazelle was used to evaluate the percent-splice-in (PSI) from RNA-seq data of cancer samples in TCGA. The TCGA maintains more than 11,000 primary cancers and matched normal RNA-seq samples spanning 33 cancer types. We have built a Gazelle index on 11095 TCGA samples based on 21-mers with an Intel Xeon 6130 CPU(@2.10 GHz with 32 cores and 2.2M cache) and 192GB RAM. The total size of the Gazelle index for the TCGA dataset is 487 GB.

PSI characterizes the involvement of a particular transcript at the alternative splicing (AS) events. Traditionally, such analysis

Figure 3: An illustration of inclusive and exclusive sequences used to query different types of alternative splicing events. The boxed region has the length of 20 since the 21-mers are used to build the Gazelle index.

relies on accurate spliced read alignment. The PSI is calculated by the fraction of the number of reads crossing the transcript-specific junctions over all the transcripts involved in the AS events.

With the availability of Gazelle, we leveraged Gazelle query to estimate the abundance of the k -mers that cross the transcript-specific junctions to calculate a similar value to PSI. For each splice event we need to verify, the inclusive splice query sequence and exclusive splice query sequence were constructed based on the gene structures and splice graphs that SpliceSeq used. Figure 3 shows the details of how the inclusive and exclusive query sequences of different types of splice events were designed. After querying both inclusive and exclusive query sequence, the predicted PSI levels were calculated using the abundances of the inclusive query sequence (A_i) and exclusive query sequence (A_e):

$$PSI = \frac{A_i}{A_i + A_e} \quad (2)$$

, where A_i and A_e were obtained using the the quantification of the expressions introduced in the method part.

A total of 1146 Lung squamous cell carcinoma (LUSC) and lung adenocarcinoma (LUAD) samples from TCGA database were selected for the comparison. A total of 66 splice events were selected based on their ability to distinguish the samples (cancer and normal) from two subtypes of lung cancer, LUAD and LUSC, according to the reference PSI in SpliceSeq database. The Pearson correlation test was performed on the reference PSIs of those splice events and PSI obtained based on Gazelle's query results. The average Pearson correlation coefficient is 0.7879. We further conducted T-SNE embeddings of all patients by taking either the reference PSI or the PSI level calculated by Gazelle query results as each patient's features. The visualization is shown in Figure 4. PSIs calculated from expression estimated by Gazelle demonstrate very similar clustering structures when compared to using PSIs calculated by the traditional RNA-seq processing pipelines. In summary, these PSI events clearly separate LUAD and LUSC samples with normal samples bearing more similarity with LUAD.

4 DISCUSSION AND CONCLUSION

The large amount of high throughput sequencing data accumulated in both public and private sectors becomes a unique and rich data

resource. However, its utility has been severely hindered by the difficulty in data access. A biologically meaningful "google" tool to screen information from large collection of sequencing dataset will be invaluable and may benefit biologists in many research disciplines. However, research to develop algorithms to enable such search engine is still in its infancy.

In this study, we designed and implemented a query system, named Gazelle, to support on-demand query of transcripts with abundance over large collection of RNA-seq datasets. The key advantages of Gazelle is its extremely fast query response, low memory footprint, and high fidelity to raw k -mer abundance. These features make it a viable tool in real applications as demonstrated in the calculation of PSIs characterizing alternative splicing events as well as transcript quantification compatible to RT-qPCR.

The two tiered structures of Gazelle offers additional benefits when indexing a very large collection of RNA-seq data. Index built by any of the existing algorithms function often function as one inseparable hashing data structure whose size grows as more datasets are included. As they require the entire index to be loaded into memory during query, the algorithm does not scale as the database continue to grow. In contrast, Both Tier1 and Tier 2 indices of Gazelle can be partially disk-based, significantly reducing memory footprint. When the query is on a sub-population of the entire index, Gazelle may easily direct the query towards the selected samples without loading all indexes into memory. Such flexibility may become increasing important as HTS data continues to grow.

We expect the abundance sequence query provided by Gazelle may benefit exploratory analysis arising from a myriad applications. It can be extremely useful in a rapid screening of novel biological features on large collection of sequencing databases. We will continue to explore novel applications to demonstrate the efficacy of sequence query tools in the near future.

ACKNOWLEDGMENTS

We thank Qi Sun for her valuable comments and suggestions on this study.

REFERENCES

- [1] Nikolas Askitis. 2009. Fast and compact hash tables for integer keys. In *Proceedings of the Thirty-Second Australasian Conference on Computer Science-Volume 91*. Australian Computer Society, Inc., 113–122.
- [2] Djamal Belazzougui, Paolo Boldi, Giuseppe Ottaviano, Rossano Venturini, and Sebastiano Vigna. 2014. Cache-oblivious peeling of random hypergraphs. In *2014 Data Compression Conference*. IEEE, 352–361.
- [3] Djamal Belazzougui, Paolo Boldi, Rasmus Pagh, and Sebastiano Vigna. 2011. Theory and practice of monotone minimal perfect hashing. *Journal of Experimental Algorithmics (JEA)* 16 (2011), 3–2.
- [4] Djamal Belazzougui, Fabiano C Botelho, and Martin Dietzfelbinger. 2009. Hash, displace, and compress. In *European Symposium on Algorithms*. Springer, 682–693.
- [5] Cláudia Bessa, Paulo Matos, Peter Jordan, and Vânia Gonçalves. 2020. Alternative Splicing: Expanding the Landscape of Cancer Biomarkers and Therapeutics. *International Journal of Molecular Sciences* 21, 23 (2020), 9032.
- [6] Fabiano C Botelho, Rasmus Pagh, and Nivio Ziviani. 2007. Simple and space-efficient minimal perfect hash functions. In *Workshop on Algorithms and Data Structures*. Springer, 139–150.
- [7] Fabiano C Botelho, Rasmus Pagh, and Nivio Ziviani. 2013. Practical perfect hashing in nearly optimal space. *Information Systems* 38, 1 (2013), 108–131.
- [8] Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. 2016. Near-optimal probabilistic RNA-seq quantification. *Nature biotechnology* 34, 5 (2016), 525.
- [9] Sara A Byron, Kendall R Van Keuren-Jensen, David M Engelthaler, John D Carpten, and David W Craig. 2016. Translating RNA sequencing into clinical diagnostics: opportunities and challenges. *Nature Reviews Genetics* 17, 5 (2016), 257–271.

Figure 4: T-SNE visualization of LUAD (Red), LUSC (Green) and Normal (Blue) samples from TCGA databases using PSI values based on Gazelle query (a) and PSI values downloaded from SpliceSeq database (b).

- [10] Ethan Cerami, Jianjiong Gao, Ugur Dogrusoz, Benjamin E. Gross, Selcuk Onur Sumer, Bülent Arman Aksoy, Anders Jacobsen, Caitlin J. Byrne, Michael L. Heuer, Erik Larsson, Yevgeniy Antipin, Boris Reva, Arthur P. Goldberg, Chris Sander, and Nikolaus Schultz. 2012. The cBio Cancer Genomics Portal: An Open Platform for Exploring Multidimensional Cancer Genomics Data. *Cancer Discovery* 2, 5 (2012), 401–404. <https://doi.org/10.1158/2159-8290.CD-12-0095> arXiv:<https://cancerdiscovery.aacrjournals.org/content/2/5/401.full.pdf>
- [11] Adam Frankish, Mark Diekhans, Anne-Maud Ferreira, Rory Johnson, Irwin Junge, Jane Loveland, Jonathan M Mudge, Cristina Sisu, James Wright, Joel Armstrong, et al. 2019. GENCODE reference annotation for the human and mouse genomes. *Nucleic acids research* 47, D1 (2019), D766–D773.
- [12] Jianjiong Gao, Bülent Arman Aksoy, Ugur Dogrusoz, Gideon Dresdner, Benjamin Gross, S Onur Sumer, Yichao Sun, Anders Jacobsen, Rileen Sinha, Erik Larsson, et al. 2013. Integrative analysis of complex cancer genomics and clinical profiles using the cBioPortal. *Sci. Signal* 6, 269 (2013), p1–p11.
- [13] Manfred G Grabherr, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, Qiangdong Zeng, et al. 2011. Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nature biotechnology* 29, 7 (2011), 644.
- [14] Brian J Haas, Alexie Papanicolaou, Moran Yassour, Manfred Grabherr, Philip D Blood, Joshua Bowden, Matthew Brian Couger, David Eccles, Bo Li, Matthias Lieber, et al. 2013. De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. *Nature protocols* 8, 8 (2013), 1494.
- [15] André Kahles, Kjong-Van Lehmann, Nora C Toussaint, Matthias Hüser, Stefan G Stark, Timo Sachsenberg, Oliver Stegle, Oliver Kohlbacher, Chris Sander, Samantha J Caesar-Johnson, et al. 2018. Comprehensive analysis of alternative splicing across tumors from 8,705 patients. *Cancer cell* 34, 2 (2018), 211–224.
- [16] Yeddyah Langsam, Moshe J. Augenstein, and Aaron M. Tenenbaum. 1996. *Data Structures Using C and C++ (2Nd Ed.)*. Prentice Hall Press, Upper Saddle River, NJ, USA.
- [17] Rasko Leinonen, Hideaki Sugawara, Martin Shumway, and International Nucleotide Sequence Database Collaboration. 2010. The sequence read archive. *Nucleic acids research* 39, suppl_1 (2010), D19–D21.
- [18] Antoine Limasset, Guillaume Rizk, Rayan Chikhi, and Pierre Peterlongo. 2017. Fast and scalable minimal perfect hashing for massive key sets. *arXiv preprint arXiv:1702.03154* (2017).
- [19] Xinan Liu, Ye Yu, Jinpeng Liu, Corrine F Elliott, Chen Qian, and Jinze Liu. 2017. A novel data structure to support ultra-fast taxonomic classification of metagenomic sequences with k-mer signatures. *Bioinformatics* 34, 1 (2017), 171–178.
- [20] Camille Marchet, Christina Boucher, Simon J Puglisi, Paul Medvedev, Mikael Salson, and Rayan Chikhi. 2021. Data structures based on k-mers for querying large collections of sequencing data sets. *Genome Research* 31, 1 (2021), 1–12.
- [21] Guillaume Marçais and Carl Kingsford. 2011. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics* 27, 6 (2011), 764–770.
- [22] Ali Mortazavi, Brian A Williams, Kenneth McCue, Lorian Schaeffer, and Barbara Wold. 2008. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature methods* 5, 7 (2008), 621.
- [23] Larsson Omberg, Kyle Ellrott, Yuan Yuan, Cyriac Kandoth, Chris Wong, Michael R Kellen, Stephen H Friend, Josh Stuart, Han Liang, and Adam A Margolin. 2013. Enabling transparent and collaborative computational analysis of 12 tumor types within The Cancer Genome Atlas. *Nature genetics* 45, 10 (2013), 1121–1126.
- [24] Qun Pan, Ofer Shai, Leo J Lee, Brendan J Frey, and Benjamin J Blencowe. 2008. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature genetics* 40, 12 (2008), 1413.
- [25] Prashant Pandey, Fatemeh Almodaresi, Michael A Bender, Michael Ferdman, Rob Johnson, and Rob Patro. 2018. Mantis: A fast, small, and exact large-scale sequence-search index. *Cell systems* 7, 2 (2018), 201–207. e4.
- [26] Michael C Ryan, James Cleland, RyangGuk Kim, Wing Chung Wong, and John N Weinstein. 2012. SpliceSeq: a resource for analysis and visualization of RNA-Seq data on alternative splicing and its functional impacts. *Bioinformatics* 28, 18 (2012), 2385–2387.
- [27] Eric W Sayers, Jeff Beck, J Rodney Brister, Evan E Bolton, Kathi Canese, Donald C Comeau, Kathryn Funk, Anne Ketter, Sunghwan Kim, Avi Kimchi, et al. 2020. Database resources of the national center for biotechnology information. *Nucleic acids research* 48, D1 (2020), D9.
- [28] Brad Solomon and Carl Kingsford. 2016. Fast search of thousands of short-read sequencing experiments. *Nature biotechnology* 34, 3 (2016), 300.
- [29] Brad Solomon and Carl Kingsford. 2018. Improved search of large transcriptomic sequencing databases using split sequence bloom trees. *Journal of Computational Biology* 25, 7 (2018), 755–765.
- [30] Chen Sun, Robert S Harris, Rayan Chikhi, and Paul Medvedev. 2018. Allsome sequence bloom trees. *Journal of Computational Biology* 25, 5 (2018), 467–479.
- [31] Katarzyna Tomczak, Patrycja Czerwińska, and Maciej Wiznerowicz. 2015. The Cancer Genome Atlas (TCGA): an immeasurable source of knowledge. *Contemporary oncology* 19, 1A (2015), A68.
- [32] John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M Stuart, Cancer Genome Atlas Research Network, et al. 2013. The cancer genome atlas pan-cancer analysis project. *Nature genetics* 45, 10 (2013), 1113.
- [33] Kosuke Yoshihara, Qianghu Wang, Wandaliz Torres-Garcia, Siyuan Zheng, Rahul-simham Vegesna, Hoon Kim, and Roel GW Verhaak. 2015. The landscape and therapeutic relevance of cancer-associated transcript fusions. *Oncogene* 34, 37 (2015), 4845–4854.
- [34] Ye Yu, Djamel Belazzougui, Chen Qian, and Qin Zhang. 2018. Memory-efficient and ultra-fast network lookup and forwarding using Othello hashing. *IEEE/ACM Transactions on Networking* 26, 3 (2018), 1151–1164.
- [35] Ye Yu, Jinpeng Liu, Xinan Liu, Yi Zhang, Eamonn Magner, Erik Lehnert, Chen Qian, and Jinze Liu. 2018. SeqOthello: querying RNA-seq experiments at scale. *Genome biology* 19, 1 (2018), 167.
- [36] Junjun Zhang, Joachim Baran, Anthony Cros, Jonathan M Guberman, Syed Haider, Jack Hsu, Yong Liang, Elena Rivkin, Jianxin Wang, and Brett Whitty. 2011. International Cancer Genome Consortium Data Portal—a one-stop shop for cancer genomics data. *Database* 2011 (2011).