BIOINFORMATICS ORIGINAL PAPER

Vol. 30 no. 20 2014, pages 2949–2955
doi:10.1093/bioinformatics/btu405

Databases and ontologies

Advance Access publication June 27, 2014

The Amordad database engine for metagenomics

Ehsan Behnam and Andrew D. Smith*

Molecular and Computational Biology, Department of Biological Sciences, University of Southern California, Los Angeles, CA, USA

Associate Editor: Janet Kelso

ABSTRACT

Motivation: Several technical challenges in metagenomic data analysis, including assembling metagenomic sequence data or identifying operational taxonomic units, are both significant and well known. These forms of analysis are increasingly cited as conceptually flawed, given the extreme variation within traditionally defined species and rampant horizontal gene transfer. Furthermore, computational requirements of such analysis have hindered content-based organization of metagenomic data at large scale.

Results: In this article, we introduce the Amordad database engine for alignment-free, content-based indexing of metagenomic datasets. Amordad places the metagenome comparison problem in a geometric context, and uses an indexing strategy that combines random hashing with a regular nearest neighbor graph. This framework allows refinement of the database over time by continual application of random hash functions, with the effect of each hash function encoded in the nearest neighbor graph. This eliminates the need to explicitly maintain the hash functions in order for query efficiency to benefit from the accumulated randomness. Results on real and simulated data show that Amordad can support logarithmic query time for identifying similar metagenomes even as the database size reaches into the millions.

Availability and implementation: Source code, licensed under the GNU general public license (version 3) is freely available for download from http://smithlabresearch.org/amordad

Contact: andrewds@usc.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on February 21, 2014; revised on June 21, 2014; accepted on June 23, 2014

1 INTRODUCTION

Metagenomics has revolutionized our knowledge of microbial communities. Such studies have uncovered intercommunity rules governing the behavior of the microbial networks and various aspects of their symbiotic or parasitic lives within their host ecosystems (Daniel, 2005; Qin *et al.*, 2010). With technological advancements on the horizon, whole-metagenome shotgun (WMS) sequencing has increasingly become popular. The results of many WMS projects ranging from direct study of simple biofilms in specific niches (Tyson *et al.*, 2004) to complicated microbial consortia like human gut microbiota (Qin *et al.*, 2012) are now publicly accessible. The first phase of the human microbiome project (Turnbaugh *et al.*, 2007) provided an unprecedented opportunity to explore our 'other genome' (Le Chatelier *et al.*, 2013). However, making sense of these data has proven tremendously challenging. These challenges stem from both the extreme size and number of possible metagenomes, which may be regarded as continuous mixtures of species whose boundaries can be poorly defined.

Popular existing frameworks such as MG-RAST (Meyer *et al.*, 2008) and MEGAN (Huson *et al.*, 2011) rely heavily on alignment-based algorithms to analyze metagenomic data. Assembly of raw reads into longer contigs, binning of sequencing reads and homology searching to characterize the predicted operational taxonomic units (OTUs) are among tasks that require sequence alignment in such frameworks (Wooley *et al.*, 2010). The computational requirements of these tasks, however, have reduced their usefulness in large-scale projects. Alignment-free approaches have been successfully applied to overcome the scalability problems associated with alignment and assembly (Chan and Ragan, 2013; Grabherr *et al.*, 2011). A large class of alignment-free algorithms rely on the the frequency of oligonucleotides of fixed length k, called k-mers, to represent biological sequences (Song *et al.*, 2014). In metagenomics, several k-mer-based algorithms have been proposed mostly targeting the phylogenetic characterization of the metagenomes (McHardy *et al.*, 2007; Nalbantoglu *et al.*, 2011). The idea is to estimate the abundance level of different bacterial families in a metagenome by assigning accurate phylogenetic labels to its reads or contigs.

We introduce Amordad, a content-based database engine for metagenomics designed to support rapid indexing and retrieval even as data volumes reach massive scales. In the most basic form, the user makes a query by providing a metagenomic sample sequence and asks for the most similar metagenomes in the database. Similarity between two metagenomes is computed, and query responses are ranked according to similarity with the query. We describe a procedure for assigning an empirical significance score to query responses in this context, which provides additional insight about retrieved metagenomes. Proximity scores also allow, for a given query, efficient identification of the full set of metagenomes sharing some threshold similarity with the query.

Amordad does not attempt any sequence alignment or assembly, and is agnostic of OTUs, focusing on the raw metagenome itself as the fundamental data element. Indexing is based on feature vectors, specifically k-mer sequences. In the next section, we describe the combination of random hashing and regular nearest neighbor graph strategies, which enable low complexity queries, both in terms of time and memory usage. Together, these dual index structures allow the database to reorganize itself continually as new data is added, an essential property that is difficult to achieve in high-dimensional geometric databases. We demonstrate the efficiency of Amordad in handling up to millions of metagenomes.

*To whom correspondence should be addressed

2 METHODS

We first describe a geometric representation we use for raw metagenomic sequencing data. Then we introduce the two indexing strategies: a randomized hashing strategy based on locality-sensitive hashing [LSH; Gionis *et al.* (1999)] and the regular nearest neighbor graph. Finally, we show how these two indexing strategies are integrated.

2.1 Geometric interpretation of metagenomes

A metagenome is the set of genomes for a population of organisms existing in some defined space at a defined time. This is commonly a complex mixture of microorganisms interacting to form a microbial community with behaviors related to the relative frequencies of different microbes in the community. Our window into this population is by sampling parts of these genomes via sequencing. Following a WMS experiment, the totality of information resides in the unprocessed sequenced reads from a given sample (e.g. in FASTQ files). Our goal is to extract information from these data, but to avoid any kind of sequence alignment, assembly (Luo *et al.*, 2012) or assignment of OTUs (Porter and Beiko, 2013). We instead use representations from alignment-free sequence comparison (Vinga and Almeida, 2003). Specifically, the set of sequenced reads is transformed into a k-mer count vector. This is a $d=4^k$-dimensional vector, where coordinate i gives the frequency of occurrence in the metagenome for the ith k-mer.

The assumption underlying the use of alignment-free methods in metagenomics is that each microorganism contributes a distinct fingerprint when represented by a k-mer count vector, while similar microbial sequences should give vectors that are closer in this d-dimensional space. This also depends on the measure used for comparing these k-mer count vectors.

Assume metagenome $\mathcal{M}$, represented as a set of reads, contains N possible k-mers, which includes all sliding windows of length k in all sequenced reads. We centralize the count of each k-mer to reduce the effect of background noise, as suggested in Kantorovitz *et al.* (2007). If each read were a random sequence, letting p_a denote the probability of drawing letter $a \in \mathcal{A}$ when generating $\mathcal{M}$, then for any $z = z_1 z_2 \dots z_k \in \mathcal{A}^k$,

$$p_z = \prod_{j=1}^{k} p_{z_j},$$

so Np_z is approximately the expected number of occurrences of z in $\mathcal{M}$. The centralized k-mer count vector is as follows:

$$\mathcal{M}(z) = X_z - Np_z,$$

where X_z is the total number of occurrences of z in $\mathcal{M}$. This simple representation accomplishes the transformation from the set of reads into a vector in $d=4^k$ dimensional space. We treat these vectors as points and hereafter, when we refer to a metagenome, we assume it is represented as a point in this space.

In addition to expressing biologically meaningful relationships, appropriate distance measures provide a statistically meaningful foundation for alignment-free sequence comparison. Metric distances are especially appealing, as they allow algorithms to take advantage of the triangle inequality and infer some distance relationships implicitly. Here, we use the angle between two metagenomes as our distance measure. Dot products are computed as usual, the norm of point x is $||x|| = \sqrt{x \cdot x}$, and the angle between points x and y is $\theta_{xy} = \arccos x \cdot y/(||x||||y||)$.

2.2 Indexing metagenomic points with LSH for angles

LSH is among the best known indexing methods for high-dimensional data (Gionis *et al.*, 1999; Lv *et al.*, 2007), and provides a framework for understanding and analyzing a large class of randomized dimension-reduction techniques. Intuitively, a family of hash functions is 'locality sensitive' if any of its members hashes closer points to the same bucket with higher probability. We use an LSH family for which the measure of proximity is the angle between points. Suppose $u \in \mathbb{R}^d$ is a random unit vector. For any $x \in \mathbb{R}^d$ define

$$h_u(x) = \begin{cases} 1 & \text{if } x \cdot u \geq 0, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Then, for any two points x and y, it is straightforward to show that

$$\Pr\left(h_u(x) = h_u(y)\right) = 1 - \theta_{xy}/\pi, \quad (2)$$

where $0 \leq \theta_{xy} \leq \pi$. Let $U = \{u_1, \dots, u_r\}$ be a set of random vectors with norm one and define the hash function as follows:

$$H_U(x) = (h_{u_1}(x), \dots, h_{u_r}(x)). \quad (3)$$

Equation (2) suggests that under the r-bit binary embedding induced by H_U, the 'locality' of points is likely to be preserved. In other words, the closer the two points in $\mathbb{R}^d$, the more likely are they to share the same mapping in the Hamming space, that is, the codomain of H_U.

This function, which is locality sensitive when distances are measured as angles for pairs of points, was first introduced in a theoretical work by Goemans and Williamson (1995) before its first use in the context of LSH by Charikar (2002). Subsequently, this function has been applied in different contexts including natural language processing (Ravichandran *et al.*, 2005) and computational biology (Behnam *et al.*, 2013).

To achieve our desired accuracy during the query process, we index points in multiple iterations, with each iteration generating a new hash table. Therefore, the first step in our indexing procedure can be summarized as follows:

- Generate U as a set of r random unit-length vectors in $\mathbb{R}^d$ [using an established method (Muller, 1959)].
- For each metagenomic point x, find $H_{U_i}(x)$ and store the result in a hash table T_i.
- Repeat the above two steps L times, producing the set $\mathcal{T} = \{T_1, \dots, T_L\}$ of hash tables.

Although LSH provides a simple indexing scheme and guarantees sublinear query complexity under certain conditions (Gionis *et al.*, 1999), its use in large databases has been hindered because a large number of hash tables can be required to overcome its approximate nature, which is critical for applications that must guarantee query accuracy. Our starting point to address this problem in the metagenomics context was to ask how could we keep some of the useful information from each hash table, without explicitly using all the hash tables or explicitly storing them. The solution was to augment the indexing structure of Amordad by incorporating a proximity graph. This graph encodes important distance relationships that have accumulated through the application of many distinct hash functions, allowing all but a small number of hash tables to be discarded. We show how this graph is constructed and later examine its use in the query process.

2.3 Supplemental indexing with a developing nearest neighbor graph

The family of hash functions defined in Equation (1) corresponds to hyperplanes in $\mathbb{R}^d$, each partition the space of metagenomes. Consider the geometric loci of all points that are hashed to the same bucket as the query point. This 'query region' is the intersection of r half-spaces, defined by r hyperplanes, that contain the query point. The query region is likely to contain the actual nearest neighbor point, but even if it does not, it is reasonable to expect that this point remains close to the query region. We grow this region by searching the neighborhood of all points inside it. Amordad encodes spatial information about the proximities of data points using a graph, and employs the graph when

processing queries to expand the search region. Specifically, for a set of n metagenomic points in the database, we construct a directed graph in which each metagenome is a node, and each node has exactly m outgoing edges pointing to its m nearest neighbors under the assumed distance measure. This type of graph is called an m-regular nearest neighbor graph, abbreviated as m-NNG (Samet, 2006, sec. 4.5.6). Supplementary Figure S1 provides a schematic depicting how this graph is used to expand the query region.

Nearest neighbor graphs have found various applications in different areas including shape recognition (Beis and Lowe, 1997) and computational geometry (Miller *et al.*, 1997), and efficient algorithms have been proposed to construct them (Dong *et al.*, 2011). To understand how we efficiently construct the m-NNG, recall that LSH provides a probabilistic measure of the relative locality of the points. When two points reside in a common hash bucket it is more likely that they should be connected in the m-NNG. This observation suggests an approximate algorithm to construct the m-NNG: Given a point x, (i) for each hash table T_i, examine all points y in the same bucket as x, (ii) keep only the m closest points to x and (iii) connect the corresponding nodes in the m-NNG. As the information from LSH is probabilistic, the probability of misidentifying an edge never reaches zero. However, the theory of LSH guarantees that this 'approximate' graph converges to the actual m-NNG, as we increase the number of hash functions/tables used in this process. In supplementary material, we present results about the rate of convergence of the m-NNG.

There is a major problem with this process as just described: the graph encodes no more information than already exists in the hash tables, which are also used in the query process. This redundancy, however, allows us to 'retire' hash functions, replacing them with new ones. This continually refreshes the randomness in Amordad, while retaining the useful information from retired hash tables, as it is encoded in the m-NNG. In Amordad, as each hash table is generated, it is placed in a queue (i.e. first-in, first-out or FIFO) of size L. At the same time, the oldest hash table in the queue is removed; we elaborate this after defining basic insertion and deletion operations for Amordad.

The nature of any useful large-scale metagenomic database is dynamic: users must be able to add points over time and possibly also remove them. These basic operations are handled as follows:

- *insert*(x) computes $H_{U_i}(x)$ for $1 \le i \le L$ and modifies i-th hash table by adding x to buckets corresponding to each $H_{U_i}(x)$. Then a new vertex is added to the graph and associated with x.
- *delete*(x) computes bucket numbers $H_{U_i}(x)$, for $1 \le i \le L$, and removes x from the appropriate buckets in all hash tables in queue. Removal of x from the graph follows a lazy deletion strategy (Jannink, 1995). This lazy deletion is essential to the efficiency of maintaining the graph because, although each node has a fixed out-degree, the in-degree is not constrained. Nodes have a reference count for in-degree. Upon deletion nodes are marked, and although never again reported in response to a query, each subsequent encounter during query or insertion operations removes an incoming edge and decrements the count; the node itself is deleted when the count reaches zero.

The insertion and deletion operations are both efficient: the complexity of each operation is dominated by evaluation of L hash functions.

Once the graph has converged, for each point, its m out-going edges indicate its m nearest neighbors. A problem arises because the insertion and deletion operations eliminate this property. A newly inserted point might be among the m nearest neighbors of any existing point, but to check this requires processing each metagenome in the database. Similarly, when a metagenome is deleted, some metagenomes will effectively have out-degree of $m-1$. The solution is continually applying maintenance operations, which one may imagine as occurring according to a regular schedule, such as once per day. The maintenance consists of l iterations of the following:

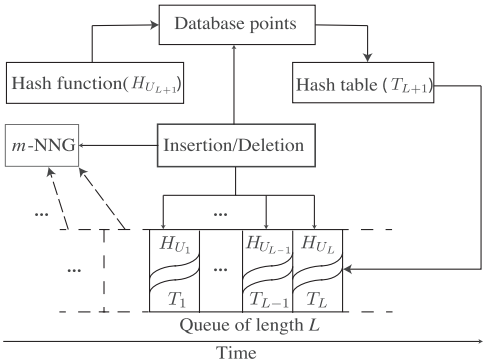

Fig. 1. Schematic showing how the basic operations interact with the different components of the database. The continual procedure for updating the graph is also depicted

- A random hash function is generated and used to create a new hash table T_{L+1}.
- T_1 is examined to update edges in the nearest neighbor graph.
- T_{L+1} is added to the queue, and T_1 is removed.

At any time during this process, queries have access to L hash tables, and T_1 is not discarded until the edges indicated by bucket co-occupancy in T_1 have been added to the graph. The schematic representing this process is depicted in Figure 1. In our current implementation, we set $L = 200$ and $l = 50$.

Amordad queries, or similarity searches, occur in two steps:

(1) For query point y and $1 \le i \le L$, compute $H_{U_i}(y)$ and check the distance between y and all other points hashing to same bucket number $H_{U_i}(y)$ in T_i.

(2) For any x in bucket $H_{U_i}(y)$, evaluate the m neighbors of x in the graph, and check their distance to y.

This procedure is summarized in Algorithm 1, where $\mathcal{N}(x)$ refers to the set of neighbor points of x in the graph, and the function *max_element*(R) gives the furthest point in set R to the query point.

In Supplementary Material, we give a heuristic model to express the influence of the nearest neighbor graph on query accuracies using a formula similar to those commonly used in the context of LSH alone.

Algorithm 1 Similarity search in Amordad

Input: Set $\mathcal{S}$ of points, L hash tables, an m-NNG, query y and parameter t
Output: Set R containing t nearest neighbors of y in $\mathcal{S}$

```
R ← ∅; θ_max ← ∞
for i ← 1 to L do
    compute H_{U_i}(y)
    for all x ∈ H_{U_i}(y) do
        for all x' ∈ N(x) ∪ {x} do
            if θ_{x'y} < θ_max then
                b_max ← max_element(b)
                b ← b ∪ {x'} \ {b_max}
                update θ_max
return b
```

3 RESULTS

We have two objectives in studying the performance of Amordad: (i) investigating its potential for identifying

metagenomic associations in large-scale applications, and (ii) demonstrating the favorable technical aspects of our database engine with particular emphasize on scalability.

3.1 Dataset and implementation details

We obtained WMS data from the European Nucleotide Archive (ENA) (Leinonen *et al.*, 2011), including ~3.5TB of compressed FASTQ files from 133 projects (08/15/13). We extracted 5-mers from each sample and obtained 5073 real metagenomic count vectors in $d = 1024$-dimensional space. We augmented this set with an additional 10^6 randomly generated metagenomes to increase the scale of our evaluations. Each simulated metagenome contained $N = 10^7$ *i.i.d.* reads of length 100. On initiation of the query, the graph and the queue are read into memory (these could be memory residents on a dedicated server). Because of space constraints, we do not load all count vectors but instead an index of their location on disk.

For comparison purposes in evaluating accuracy, we implemented the brute-force (quadratic time) nearest neighbor graph construction and the linear time scan for exact results from queries. The implementation of the database engine and associated utilities was done in C++ and with the assistance of the Boost Graph Library (http://www.boost.org/libs/graph). Our Amordad implementation includes multi-threaded query processing. However, all evaluations described here were done using a single core of a 2.4 GHz Xeon CPU and 16GB of memory.

Building Amordad is a multi-stage process, involving the construction of hash tables and the nearest neighbor graph. The overall time to build the database with 10^6 count vectors was 44.9 h. We also investigated the memory requirement for building the database. Initial construction of the nearest neighbor graph required the largest memory among all of the programs included in Amordad software package. When examining $L = 200$ hash tables to construct a 20-NNG representing 10^6 metagenomes, the peak memory was 10.4GB.

3.2 Assigning significance to query responses

Assigning significance to query results is essential, but there should be no expectation that actual distances between metagenomes (for any reasonable representation) follow a simple statistical distribution. To gain insight into how these distances can be interpreted in the context of Amordad queries, we examined the distribution of distances for relatively small sample of available metagenomes.

Figure 2 shows the empirical distribution of distances between the 5073 metagenomes from the 133 projects. Denoting F as the cumulative function of this distribution, for any query point y with response x, we defined $F(\theta_{xy})$ as the significance score of the query answer. This score shows the relative ranking of the distance between x and y among all of the distance measurements. If $F(\theta_{xy})$ is large, the relatedness between x and y might be questionable. In fact, if x and y are two unrelated metagenomes, the centralization of count vectors and the linearity of expectations assert that $E(\theta_{xy}) = \pi/2$. This important observation suggests that any meaningful θ_{xy} must belong to the tail of the empirical distribution.

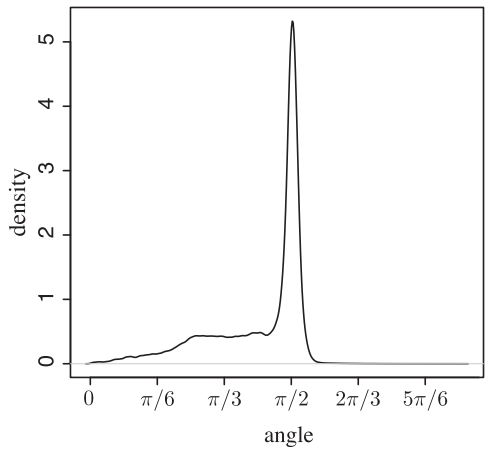

Fig. 2. Probability density function of angles resulted from real metagenomes

We decided to define the maximum proximity radius as the maximum distance between the query point and the retrieved data point such that the response is assumed relevant to the query. We denote the maximum proximity radius by θ_0, and for a query y, we filter out any response x if $\theta_{xy} > \theta_0$. In our evaluations, a liberal threshold is $F^{-1}(0.2)$, which results in $\theta_0 = 0.987$ radians or $\theta_0 \approx 57^\circ$. A more conservative cut-off is $F^{-1}(0.1)$ which is equivalent to $\theta_0 = 0.752$ radians or approximately 43°. The conservative and liberal cutoff thresholds are respectively selected as the first and second deciles of the empirical cumulative distribution of distances within the database.

3.3 Evaluation of the graph effect on the query space requirement

We can view the m-NNG as having two effects: increasing accuracy when using a fixed number of hash tables or reducing the number of hash tables required to achieve a particular accuracy. We conducted a series of experiments to compare the performance of the various instances of Amordad with the basic LSH algorithm when they are both using the same sets of hash tables. For LSH alone, we exclude the graph. The evaluation metric used here is average recall on a set of queries. Given a query y, denoting the t nearest neighbors as $N_t(y)$, if $R(y)$ is the set of points reported by the algorithm, then

$$\text{recall}(y) = |N_t(y) \cap R(y)|/|N_t(y)|.$$

The detailed experimental set up is as follows: We randomly selected 10 of 133 projects and regarded their 94 samples as the query set. Different instances of Amordad were then constructed considering all other samples as database points and m-NNG for $m \in \{1, 10, 20\}$. Next, using $1 \leq L \leq 200$ hash tables, we compared the average recall value of different instances of Amordad with the basic LSH algorithm. For simplicity, we only considered the first nearest neighbor (*i.e.* $t = 1$), and performed experiments separately for both the conservative and liberal-maximum proximity radius. If the distance between any query point and its nearest neighbor exceeded the maximum proximity radius in an experiment, it was discarded from the query set. As hash functions are randomly generated, we repeated each experiment

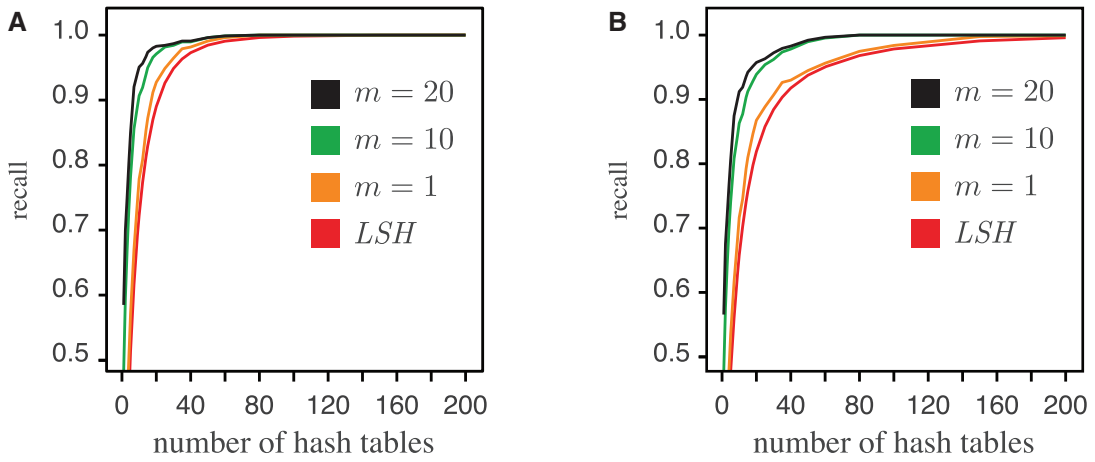

Fig. 3. Average recall for a query set, showing (**A**) the conservative and (**B**) liberal maximum proximity radius

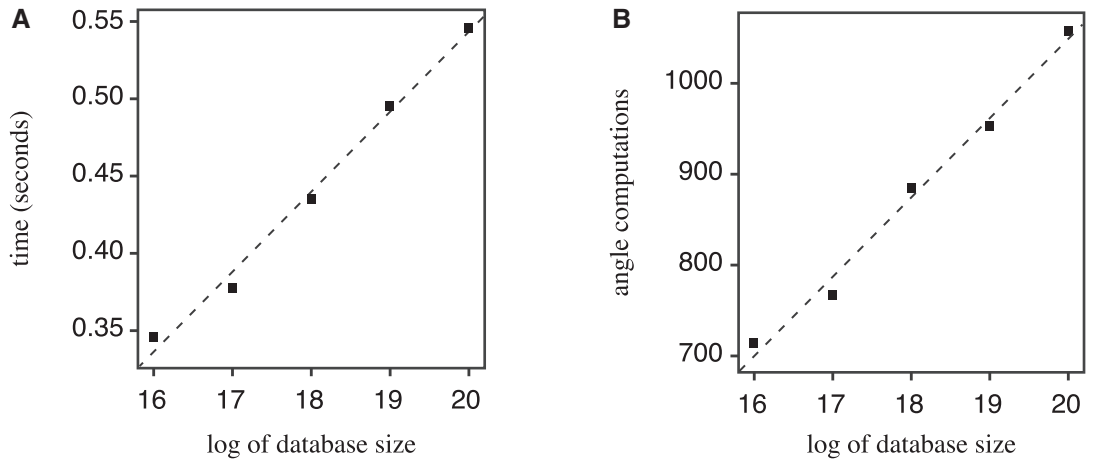

Fig. 4. (**A**) Average wall-time and (**B**) required distance computations per query as a function of (log) database size

10 times with different set of hash functions and reported average recall.

Based on the results presented in Figure 3, for a fixed accuracy, Amordad requires considerably fewer hash tables than LSH. When the graph is constructed for $m = 10$ or $m = 20$, Amordad already exhibited no query errors in any experiment. However, LSH is not able to reach perfect recall even when $L = 200$ hash tables are used under the liberal-maximum proximity radius.

3.4 Measuring query time in a large-scale database

We conducted a set of experiments to investigate the scalability of the Amordad engine. Similar to the previous experiment, we randomly selected 10 projects and used their metagenomes (183) for queries. All $n_d = 4890$ remaining metagenomic points were used, along with n_s simulated points, to construct the different instances of the database. We set the value of n_s such that $n_s + n_d = 2^i$ for i increasing from 16 to 20. Then, we created an instance of the database with n points as follows. We set the number of hash functions to $r = \log n$. We used the conservative threshold $\theta_0 = 0.752$ to determine the number of hash tables required by the LSH to bound the probability of missing a nearest neighbor below $\sigma = 0.05$. Finally, we constructed the graph with 10 outgoing edge from each node.

To evaluate the relation between the query time and the number of points in the database, we calculated the average wall-time and number of angle computations required for responding to all queries. As shown in Figure 4, the query time grows almost linearly with $\log n$. This establishes the scalability of Amordad, validating its ability to efficiently process queries in large database instances. In each experiment conducted here, the query accuracy was 100%, again demonstrating the effect of the graph to drive the probability of query error to virtually zero.

3.5 Relationships captured in the nearest neighbor graph

Our original purpose for including the m-NNG in the design of Amordad was to eliminate the space that would otherwise be required to explicitly use enough hash tables to ensure the desired accuracy. But it can also be leveraged as a persistent and adaptive knowledge base that links close points and therefore permits the detection of biologically similar metagenomes. We expect

Table 1. Top 10 most similar pairs of metagenomes from different projects, with scope of corresponding projects, distance between metagenomic points (i.e. the angle in degrees) and associated significance scores

Accession I	Accession II	Project scope I	Project scope II	Distance	Relative ranking
ERP001737	ERP001736	Sea water	Sea water	4°	0.0016
ERP001736	ERP001227	Sea water	North sea water	4.6°	0.0020
SRP009476	ERP001736	Sea water	Sea water	5.3°	0.0023
ERP001068	SRP013944	Low and neutral PH soil	Forest soil	5.3°	0.0023
SRP021115	ERP001736	Marine matter	Sea water (different depth)	5.4°	0.0023
SRP009476	SRP021115	Sea water	Marine matter	5.5°	0.0024
SRP017582	ERP001568	Oil sands	Polluted water	6.5°	0.0028
ERP001737	SRP021115	Sea water	Marine matter	6.6°	0.0029
SRP016067	ERP002469	Gut microbiota[a]	Gut microbiota[b]	6.8°	0.0030
SRP019913	SRP021115	Red sea water	Marine matter	7.3°	0.0033

[a]From a cohort of normal and atherosclerosis samples.[b]Metagenomes are sampled from European women with normal, impaired and diabetic glucose control.

many edges in the graph will link between biological replicates, the precise definition of which might depend on the goals of the project. For example, a cohort representing sea water samples taken at slightly different depths, or gut microbiota samples obtained at slightly different times. Recovering biological replicates within a cohort is an easy task under our framework and could be potentially useful. However, we claim our approach can cluster inherently similar metagenomes together even when the methodologies for obtaining, preparing and sequencing samples vary.

To demonstrate this type of relationship captured within Amordad, we constructed the least informative graph (*i.e.* the 1-NNG) using metagenomes from the 133 projects. We identified any edge connecting metagenomes that belong to two distinct projects. Among the 194 such edges, 58 showed the significance score <0.01. Table 1 shows the top 10 most similar pairs of samples (projects are listed by accession number from ENA). To avoid redundancy, any pair of projects is reported at most once in the table (Supplemetary Material contains a complete list). In general, we expect the number of applications that can benefit from mining this graph is extremely broad.

4 DISCUSSION

We presented Amordad, a database engine for whole-metagenome sequencing data. Amordad uses alignment-free principles, representing metagenomes as points in a high-dimensional geometric space. LSH is used to efficiently index the database points. To improve accuracy and efficiency beyond what is practical from LSH alone, Amordad augments this indexing with a regular nearest neighbor graph. The randomness in Amordad is continually refreshed by resampling new random hash functions. Although only a fixed number of hash functions is alive within Amordad at any given time, those that are extinct have contributed to optimizing connectivity in the graph, and thus continue to assist queries even if they are no longer used for hashing. Results from a series of experiments have demonstrated this approach to have a significant effect on query efficiency and accuracy.

The ability to cope with the rapid accumulation of data was a central design goal. LSH is a well-known approach to index high-dimensional data, but it is also a randomized method. A major drawback is the number of hash tables one must maintain to provide guarantees on the accuracy of queries, imposing time and space constraints. This is a well-known problem, and solutions have been proposed to overcome this pitfall (Lv *et al.*, 2007; Panigrahy, 2006). These solutions mostly rely on the fact that each hash function (i.e. each bit in our scheme) provides a ranking based on the proximity of points to a certain query. Therefore, even if the closest neighbor is not hashed to the same bucket as the query, it is highly probable that buckets close to the query bucket contain the nearest neighbor. Consequently, one may avoid generating many hash tables at the cost of checking more buckets from each table. This clever strategy has a direct effect on the indexing memory consumption (Lv *et al.*, 2007). However, as the distance between the query and its nearest neighbors increases, there is a rapid growth in the number of buckets that must be checked. This presents a challenge in our applications because of the inherent diversity of microbial communities in many circumstances. In fact, if we view a set of biologically related metagenomes as a cluster in high-dimensional space, the diameter of some complex clusters like human gut microbiota (Arumugam *et al.*, 2011), might be large enough to make the process of searching close buckets to the query bucket time-consuming.

Our approach, augmenting the LSH indexing with a graph structure, avoids having to explicity search additional buckets, and depends on transitivity of relationships encoded in the graph. Intuitively, if the underlying distance measure between metagenomes satisfies the triangle inequality, the graph should guide the search in the most appropriate directions. Additionally, the LSH provides a means of streamlining the database maintenance process. Hash tables are maintained in a fixed size queue that is continually updated, and each new hash table contributes to refining edges in the graph. As a consequence, we are able to keep the graph in a near ideal state without requiring explicit operations to maintain the graph after metagenomes are added to (or removed from) the database. This design decision acknowledges that in the near future large (and distributed) metagenome databases will be updated frequently, and efficiency of queries will be more important to users of Amordad.

It is clear that many applications of Amordad will involve maintaining large metagenomic data clusters so that new metagenomes can be understood through their neighborhoods in the graph. From this perspective, the graph becomes the primary structure, and the LSH serves simply to find the right neighborhood without exhaustive search.

Finally, we address some limitations of this work. We represent each metagenome by a feature vector that included the frequencies of all k-mers for a fixed k. We did not use feature selection, and were bound to relatively small values of k. Many of the k-mers were almost certainly irrelevant in our experiments on real data. More flexibility could be obtained if a metagenome is represented by a 'bag of features' (Salton, 1991) composed of a variety of words (i.e. subsets of k-mers for varying k). From statistical point of view, this is equivalent to shifting the paradigm from a full Markov chain of order k toward a variable length Markov chain (Bühlmann and Wyner, 1999). The main challenge is to find a parsimonious algorithm for feature extraction at large scale. More important than irrelevant features, however, are features representing technical artifacts, for example, of the sequencing experiments.

Funding: This work was supported by the National Institute of Health [P50 HG002790].

Conflict of interest: none declared.

REFERENCES

Arumugam,M. *et al.* (2011) Enterotypes of the human gut microbiome. *Nature*, **473**, 174–180.

Behnam,E. *et al.* (2013) A geometric interpretation for local alignment-free sequence comparison. *J. Comput. Biol.*, **20**, 471–485.

Beis,J.S. and Lowe,D.G. (1997) Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. Conference on Computer Vision and Pattern Recognition, Puerto Rico, pp. 1000–1006.

Bühlmann,P. and Wyner,A.J. (1999) Variable length Markov chains. *Ann. Stat.*, **27**, 480–513.

Chan,C.X. and Ragan,M.A. (2013) Next-generation phylogenomics. *Biol. Direct*, **8**, 1–6.

Charikar,M.S. (2002) Similarity estimation techniques from rounding algorithms. In: *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*. ACM, Las Vegas, Nevada, pp. 380–388.

Daniel,R. (2005) The metagenomics of soil. *Nat. Rev. Microbiol.*, **3**, 470–478.

Dong,W. *et al.* (2011) Efficient k-nearest neighbor graph construction for generic similarity measures. In: *Proceedings of the 20th International Conference on World Wide Web*. ACM, Hyderabad, India, pp. 577–586.

Gionis,A. *et al.* (1999) Similarity search in high dimensions via hashing. In: *VLDB*. Vol. 99, pp. 518–529.

Goemans,M.X. and Williamson,D.P. (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, **42**, 1115–1145.

Grabherr,M.G. *et al.* (2011) Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat. Biotechnol.*, **29**, 644–652.

Huson,D.H. *et al.* (2011) Integrative analysis of environmental sequences using megan4. *Genome Res.*, **21**, 1552–1560.

Jannink,J. (1995) Implementing deletion in B+-trees. *ACM Sigmod Rec.*, **24**, 33–38.

Kantorovitz,M. *et al.* (2007) Asymptotic behavior of k-word matches between two uniformly distributed sequences. *J. Appl. Probab.*, **44**, 788–805.

Le Chatelier,E. *et al.* (2013) Richness of human gut microbiome correlates with metabolic markers. *Nature*, **500**, 541–546.

Leinonen,R. *et al.* (2011) The european nucleotide archive. *Nucleic Acids Res.*, **39**, D28–D31.

Luo,R. *et al.* (2012) SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, **1**, 18.

Lv,Q. *et al.* (2007) Multi-probe LSH: efficient indexing for high-dimensional similarity search. In: *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, Vienna, Austria, pp. 950–961.

McHardy,A.C. *et al.* (2007) Accurate phylogenetic classification of variable-length DNA fragments. *Nat. Methods*, **4**, 63–72.

Meyer,F. *et al.* (2008) The metagenomics rast server–a public resource for the automatic phylogenetic and functional analysis of metagenomes. *BMC Bioinformatics*, **9**, 386.

Miller,G.L. *et al.* (1997) Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, **44**, 1–29.

Muller,M.E. (1959) A note on a method for generating points uniformly on n-dimensional spheres. *Commun. ACM*, **2**, 19–20.

Nalbantoglu,O.U. *et al.* (2011) RAIphy: phylogenetic classification of metagenomics samples using iterative refinement of relative abundance index profiles. *BMC Bioinformatics*, **12**, 41.

Panigrahy,R. (2006) Entropy based nearest neighbor search in high dimensions. In: *Proceedings of the seventeenth annual ACM-SIAM Symposium on Discrete Algorithm*. ACM, Philadelphia, PA, pp. 1186–1195.

Porter,M.S. and Beiko,R.G. (2013) SPANNER: Taxonomic assignment of sequences using pyramid matching of similarity profiles. *Bioinformatics*, **29**, 1858–1864.

Qin,J. *et al.* (2010) A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, **464**, 59–65.

Qin,J. *et al.* (2012) A metagenome-wide association study of gut microbiota in type 2 diabetes. *Nature*, **490**, 55–60.

Ravichandran,D. *et al.* (2005) Randomized algorithms and NLP: using locality sensitive hash function for high speed noun clustering. In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Ann Arbor, MI, pp. 622–629.

Salton,G. (1991) Developments in automatic text retrieval. *Science*, **253**, 974–980.

Samet,H. (2006) *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, Burlington, MA.

Song,K. *et al.* (2014) New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Brief. Bioinformatics*, **15**, 343–53.

Turnbaugh,P.J. *et al.* (2007) The human microbiome project. *Nature*, **449**, 804–810.

Tyson,G.W. *et al.* (2004) Community structure and metabolism through reconstruction of microbial genomes from the environment. *Nature*, **428**, 37–43.

Vinga,S. and Almeida,J. (2003) Alignment-free sequence comparison–a review. *Bioinformatics*, **19**, 513–523.

Wooley,J.C. *et al.* (2010) A primer on metagenomics. *PLoS Comput. Biol.*, **6**, e1000667.