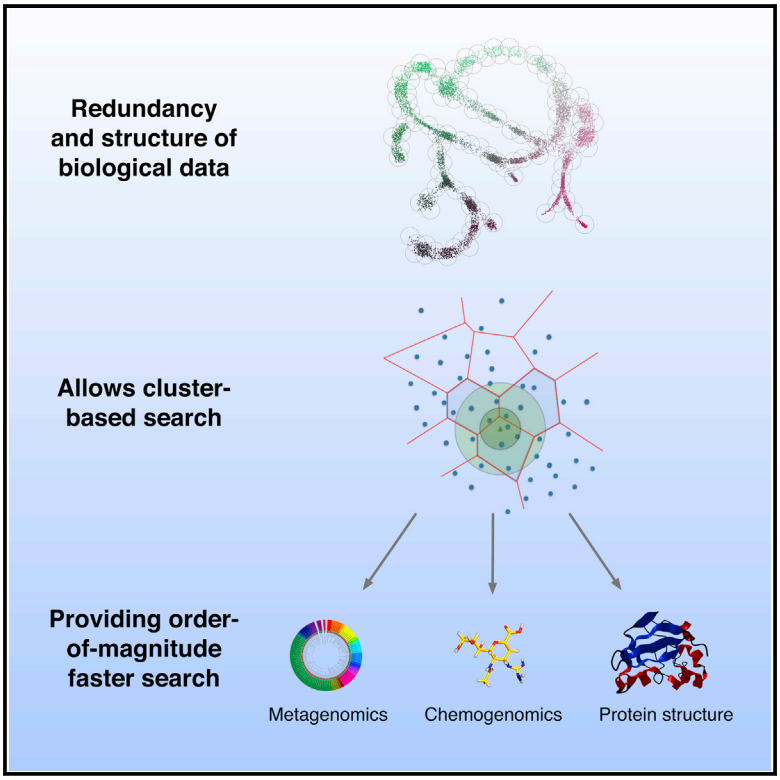
Article

Cell Systems

Entropy-Scaling Search of Massive Biological Data

Graphical Abstract



Authors

Y. William Yu, Noah M. Daniels, David Christian Danko, Bonnie Berger

Correspondence

bab@mit.edu

In Brief

Yu, Daniels et al. describe a general framework for efficiently searching massive datasets having certain properties common in biology.

Highlights

- We describe entropy-scaling search for finding approximate matches in a database
- Search complexity is bounded in time and space by the entropy of the database
- We make tools that enable search of three largely intractable real-world databases
- The tools dramatically accelerate metagenomic, chemical, and protein structure search

Yu et al., 2015, Cell Systems *1*, 130–140
August 26, 2015
http://dx.doi.org/10.1016/j.cels.2015.08.004

CellPress

CellPress

Cell Systems
Article

Entropy-Scaling Search of Massive Biological Data

Y. William Yu,[1,2,3] Noah M. Daniels,[1,2,3] David Christian Danko,[2] and Bonnie Berger[1,2,*]
[1]Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[2]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[3]Co-first author
*Correspondence: bab@mit.edu
http://dx.doi.org/10.1016/j.cels.2015.08.004

SUMMARY

Many datasets exhibit a well-defined structure that can be exploited to design faster search tools, but it is not always clear when such acceleration is possible. Here, we introduce a framework for similarity search based on characterizing a dataset's entropy and fractal dimension. We prove that searching scales in time with metric entropy (number of covering hyperspheres), if the fractal dimension of the dataset is low, and scales in space with the sum of metric entropy and information-theoretic entropy (randomness of the data). Using these ideas, we present accelerated versions of standard tools, with no loss in specificity and little loss in sensitivity, for use in three domains—high-throughput drug screening (Ammolite, 150× speedup), metagenomics (MICA, 3.5× speedup of DIAMOND [3,700× BLASTX]), and protein structure search (esFragBag, 10× speedup of FragBag). Our framework can be used to achieve "'compressive omics," and the general theory can be readily applied to data science problems outside of biology (source code: http://gems.csail.mit.edu).

INTRODUCTION

Throughout all areas of data science, researchers are confronted with increasingly large volumes of data. In many fields, this increase is exponential in nature, outpacing Moore's and Kryder's laws on the respective doublings of transistors on a chip and long-term data storage density (Kahn, 2011). As such, the challenges posed by the massive influx of data cannot be solved by waiting for faster and larger capacity computers but, instead, require the development of data structures and representations that exploit the structure of the dataset.

Here, we focus on similarity search, where the task at hand is to find all entries in a database that are "similar," or approximate matches, to a query item. Similarity search is a fundamental operation in data science and lies at the heart of many other problems, much like how sorting is a primitive operation in computer science. Traditionally, approximate matching has been studied primarily in the context of strings under edit distance metrics (Box 1) (e.g., for a spell-checker to suggest the most similar words to a misspelled word) (Ukkonen, 1985). Several approaches, such as the compressed suffix array and the FM-index (Grossi and Vitter, 2005; Ferragina and Manzini, 2000), have been developed to accelerate approximate matching of strings. However, it has been demonstrated that similarity search is also important in problem domains where biological data are not necessarily represented as strings, including computational screening of chemical graphs (Schaeffer, 2007) and searching protein structures (Budowski-Tal et al., 2010). Therefore, approaches that apply to more general conditions are needed.

As available data grow exponentially (Berger et al., 2013; Yu et al., 2015) (e.g., genomic data in Figure S1), algorithms that scale linearly (Box 1) with the amount of data no longer suffice. The primary ways in which the literature addresses this problem—locality sensitive hashing (Indyk and Motwani, 1998), vector approximation (Ferhatosmanoglu et al., 2000), and space partitioning (Weber et al., 1998)—involve the construction of data structures that support more efficient search operations. However, we note that, as biological data increase, not only does the redundancy present in the data also increase (Loh et al., 2012) but also internal structure (such as the fact that not all conceivable configurations, e.g., all possible protein sequences, actually exist) also becomes apparent. Existing general-purpose methods such as compressed data structures (Grossi and Vitter, 2005) do not explicitly exploit the particular properties of biological data to accelerate search (see the Theory section in the Supplemental Experimental Procedures).

Previously, our group demonstrated how redundancy in genomic data could be used to accelerate local sequence alignment. Using an approach that we called "compressive genomics," we accelerated BLAST and BLAT (Kent, 2002) by taking advantage of high redundancy between related genomes using link pointers and edit scripts to a database of unique sequences (Loh et al., 2012). We have used similar strategies to obtain equally encouraging results for local alignment in proteomics (Daniels et al., 2013). Empirically, this compressive acceleration appears to scale almost linearly in the entropy of the database, often resulting in orders of magnitude better performance; however, these previous studies neither proved complexity bounds nor established a theory to explain these empirical speedups.

Here, we generalize and formalize this approach by introducing a framework for similarity search of omics data. We prove that search performance primarily depends on a measure of the novelty of new data, also known as entropy. This framework, which we call entropy-scaling search, supports the creation of a data structure that provably scales linearly in both time and space with the entropy of the database, and thus sublinearly with the entire database.

Box 1. Definitions

Edit distance: the number of edits (character insertions, deletions, or substitutions) needed to turn one string into another.

Scale, in time and space: the amount of time or space a task takes as a function of the amount of data on which it must operate. A task requiring time directly proportional to the size of the data is said to scale linearly; for example, searching a database takes twice as long if the database grows by a factor of two.

Distance metric: a measure of distance that obeys several mathematical properties, including the triangle inequality.

Covering spheres: a set of spheres around existing points so that every point is contained in at least one sphere and no sphere is empty.

Metric entropy: a measure of how dissimilar a dataset is from itself. Defined as the number of covering spheres.

Fractal dimension: a measure of how the number of points contained within a sphere scales with the radius of that sphere.

Information-theoretic entropy: often used in data compression as shorthand for the number of bits needed to encode a database or a measure of the randomness of that database.

Pattern matching: refers to searching for matches that might differ in specific ways from a query, such as wildcards or gaps, as opposed to searching for all database entries within a sphere of a specified radius as defined by an arbitrary distance function.

We introduce two key concepts for characterizing a dataset: metric entropy and fractal dimension. Intuitively, metric entropy measures how dissimilar the dataset is from itself, and fractal dimension measures how the number of spheres needed to cover all points in a database scales with the radii of those spheres. Both are rigorously defined later, but note that metric entropy is not to be confused with the notion of a distance metric (Box 1). Using these two concepts, we show that, if similarity is defined by a metric-like distance function (e.g., edit or Hamming distance) and the database exhibits both low metric entropy and fractal dimension, the entropy-scaling search performs much better than naïve and even optimized methods. Through three applications to large databases in chemogenomics, metagenomics, and protein structure search, we show that this framework allows for minimal (or even zero) loss in recall, coupled with zero loss in specificity. The key benefit of formulating entropy-scaling search in terms of metric entropy and fractal dimension is that this allows us to provide mathematically rigorous guidance as to how to determine the efficacy of the approach for any dataset.

RESULTS

Entropy-Scaling Similarity Search

The basic framework for the entropy-scaling search of a database involves four steps. (1) Analyze the database to define a high-dimensional space and determine how to map database entries onto points in this space (this mapping may be one-to-one). (2) Use this space and a measure of similarity between points to group entries in the database into clusters. (3) To search for a particular query item, perform a coarse-grained search to identify the clusters that could possibly contain the query. (4) Do a fine-grained search of the points contained within these clusters to find the closest matches to the query (Figure 1).

Here, we provide conceptual motivation for this process. In the following text, we consider entropy to be nearly synonymous with distance between points in a high-dimensional space; thus, with low entropy, newly added points do not tend to be far from all existing points. For genomic sequences, the distance function can be edit distance; for chemical graphs, it can be Tanimoto distance; and for general vectors, it can be Euclidean or cosine distance. We are interested in the similarity search problem of finding all points in a set that are close to (i.e., similar to) the query point.

Let us first consider what it means for a large biological dataset, considered as points in a high-dimensional space, to be highly redundant. Perhaps many of the points are exact duplicates; this easy scenario is trivially exploited by de-duplication and is already standard practice with datasets such as the NCBI's non-redundant (NR) protein database (Pruitt et al., 2005). Maybe the points mostly live on a low-dimensional subspace; statistical tools such as principal-component analysis (PCA) exploit this property in data analysis. Furthermore, if the dimension of the subspace is sufficiently low, it can be divided into cells, allowing quick similarity searches by looking only at nearby cells (Weber et al., 1998). However, when the dimensionality of the subspace increases, cell search time grows exponentially; additionally, in sparse datasets, most of the cells will be empty, which wastes search time.

More importantly, biological datasets generally do not live in low-dimensional subspaces. Consider the instructive case of genomes along an evolutionary "tree of life" (Figure 2). Such a tree has many branches (although admixture merges branches back together) and looks nearly one-dimensional locally, but it is globally of higher dimension. Additionally, because of differences due to mutation, each of the branches is also "thick" (high dimensional) when looked at closely. Viewing this example as a low-dimensional subspace, as in PCA, is incorrect.

However, the local low dimensionality can be exploited by looking on the right scales: a coarse scale in which the tree looks one-dimensional locally and a fine scale where the branch width matters. We cover the tree with spheres (Box 1) of radius r_c, where r_c is on the order of the branch width; these spheres determine our clusters, and the number of them is the metric entropy of the tree (Tao, 2008). Because all the points within a sphere are close to each other, they are highly redundant and can be encoded in terms of one another, saving space.

By the triangle inequality, in order to search for all points within distance r of a query, we only need to look in nearby spheres with centers (i.e., representatives) within a distance $r + r_c$ of the query (Figure 1D). However, because each sphere has a radius comparable to branch width, the tree is locally one dimensional on the coarse scale; that is, spheres largely tend to extend along the branches of the tree rather than in all directions. We will call this property of local scaling the fractal dimension d of the tree at the scale r_c (Falconer, 1990), where r_c is essentially our ruler size and $d = 1$. Thus, increasing the search radius for coarse search only linearly increases the number of points that need to be searched in a fine search.

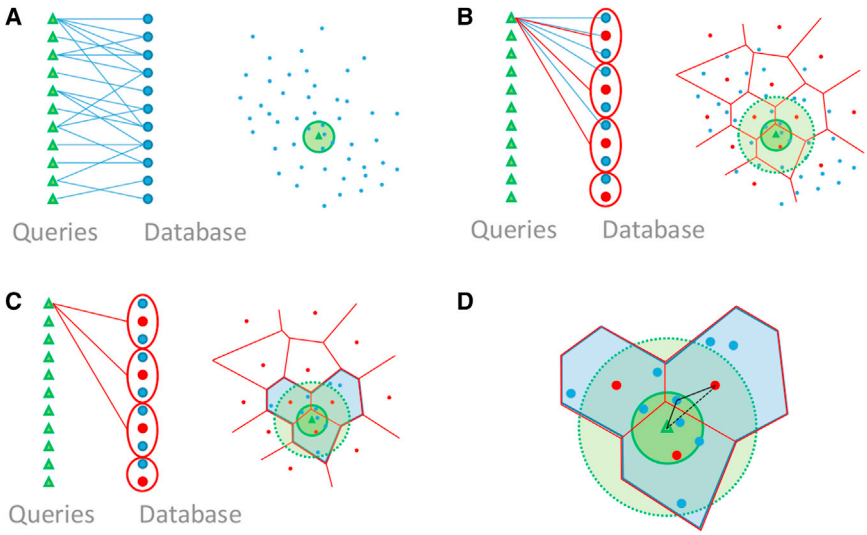


Figure 1. Entropy-Scaling Framework for Similarity Search

(A–D) As shown, (A) the naive approach tests each query against each database entry to find entries within distance r of the query (inside the small green disc). (B) By selecting appropriate cluster centers with maximum radius, r_c, to partition the database, we can (C) first do a coarse search to find all cluster centers within distance $r + r_c$ of a query (larger green disc), and then the (D) triangle inequality guarantees that a fine search over all corresponding cluster entries (blue polygonal regions) will suffice.

A similar analysis holds in the more general case where $d \neq 1$. The entropy-scaling frameworks we introduce can be expected to provide a boost to approximate search when fractal dimension d of a dataset D is low (i.e., close to 1) and metric entropy k is low. Specifically, the ratio $|D|/k$ provides an estimate of the acceleration factor for just the coarse search component compared to a full linear search of a database D. Local fractal dimension around a data point can be computed by determining the number of other data points within two radii, r_1 and r_2, of that point; given those point counts (n_1 and n_2, respectively), fractal dimension d is simply

$$d = \frac{log\ (n_2/n_1)}{log\ (r_2/r_1)}.$$

Sampling this property over a dataset can provide a global average fractal dimension. When we search a larger radius around a query, the number of points we encounter grows exponentially with the fractal dimension; low fractal dimension implies that this growth will not obviate the gains provided by an entropy-scaling data structure.

More formally, given a database with fractal dimension d and metric entropy k at the scale r_c, we show in the Supplemental Experimental Procedures that the time complexity of similarity search on database D for query q with radius r is:

$$O\left(\underbrace{k}_{\text{metric entropy}} + \overbrace{|B_D(q,r)|}^{\text{output size}} \underbrace{\left(\frac{r+2r_c}{r}\right)^d}_{\text{scaling factor}}\right).$$

Thus, for small fractal dimension and output size, similarity search is asymptotically linear in metric entropy. Additionally, because the search has to look at only a small subset of the clusters, the clusters can be stored in compressed form and only decompressed as needed, giving space savings that also scale with entropy. The space complexity scales with the sum of metric and information-theoretic entropy, rather than just metric entropy (see the Theory section in Supplemental Experimental Procedures).

Practical Application of Entropy-Scaling Search

We have presented the simplest such data to analyze for clarity of exposition. However, real data are generally messier. Sometimes the distance function is not a metric, so we lose the triangle inequality guarantee of 100% sensitivity; sometimes, different distance functions can be used for the clustering versus search; and sometimes even what counts as a distinct data point is not entirely clear without domain knowledge (for example, long genomic sequences might be better broken into shorter subsequences).

To show that entropy-scaling frameworks are robust to the variations presented by real data, we explored a diversity of applications from three major biological "big challenges of big data"—pharmaceuticals, metagenomics, and protein structure (Marx, 2013). We demonstrate that the general scheme results in order-of-magnitude improvements in running time in these different contexts, promising to enable new workflows for practitioners (e.g., fast first-pass computational drug screens and local analyses of sequencing data in remote field sites for real-time epidemic monitoring). These applications are enabled by augmenting the framework with domain-specific distance functions in different stages of the process, as well as preprocessing to take advantage of domain-specific knowledge. We expect that as long as the dataset exhibits both low entropy and low fractal dimension—and this is empirically true in biological systems—our entropy-scaling framework has the potential to achieve massive speedup over more naïve methods and significant speedup even over other highly optimized methods.

Source code for the applications discussed here is available at http://gems.csail.mit.edu and in the Supplemental Information.

Application to High-Throughput Drug Screening

Chemogenomics is the study of drug and target discovery by using chemical compounds to probe and characterize proteomic functions (Bredel and Jacoby, 2004). Particularly in the field of drug discovery and drug repurposing, prediction of biologically active compounds is a critical task. Computational high-throughput screening can eliminate many compounds from wet-lab consideration, but even this screening can be time consuming. PubChem (Bolton et al., 2008), a widely used repository of molecular compound structures, has grown greatly since 2008. In July 2007, PubChem contained 10.3 million compounds. In October 2013, PubChem contained roughly 47 million

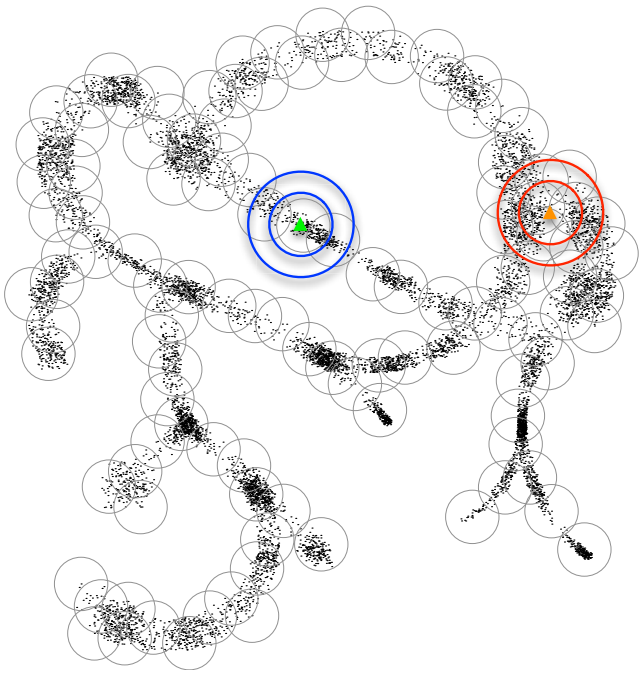


Figure 2. Cartoon Depiction of Points in a High-Dimensional Space
This cartoon depicts points in an arbitrary high-dimensional space that live close to a one-dimensional tree-like structure, as might arise from genomes generated by mutation and selection along an evolutionary tree of life. Although high dimensional at a fine scale, at the coarser scale of covering spheres, the data cloud looks nearly one-dimensional, which enables entropy scaling of similarity search. The cluster center generation was performed using the same method we used for protein structure search. The blue circles around the green query point illustrate low fractal dimension: the larger radius circle contains only linearly more points than the smaller one, rather than exponentially more. In contrast, the red circles around the orange query point illustrate higher local fractal dimension.

compounds, while in December 2014, it contained 61.3 million compounds.

We designed a compression and search framework around one of the standard techniques for high-throughput screening of potential drug compounds, the use of maximum common subgraph (MCS) to identify similar motifs among molecules (Cao et al., 2008; Rahman et al., 2009). We introduce Ammolite, a method for clustering molecular databases such as PubChem and for quickly searching for similar molecular structures in compressed space. Ammolite demonstrates that entropy-scaling methods can be extended to data types that are not inherently sequence based. Ammolite is a practical tool that provides approximately a factor of 150 speedup with greater than 92% accuracy compared to the popular small molecule subgraph detector (SMSD) (Rahman et al., 2009).

An MCS-based search of molecule databases typically matches pairs of molecules by Tanimoto distance (Rahman et al., 2009). Tanimoto distance obeys the triangle inequality and is more useful in the domain of molecular graphs than other distance metrics such as graph distance (Bunke and Shearer, 1998).

To compress a molecule database, we project the space of small molecules onto a subspace by removing nodes and edges that do not participate in simple cycles (Figure S2); note that a molecule without cycles will collapse to a single node. Clusters are exactly pre-images of this projection operator (i.e., all molecules that are isomorphic after simplification form a cluster). Coarse search is performed by finding the MCS on this much smaller projection subspace. This step increases speed by reducing both the required number of MCS operations and the time required for each MCS operation, which scales with the size of the molecule. Further reduction in search time is accomplished by grouping clusters according to size of the molecules within; because Tanimoto distance relies on molecule size, clusters containing molecules that are significantly larger or smaller than the query need not be searched at all.

The time required to cluster a large database such as PubChem is, nonetheless, significant; clustering the 306-GB PubChem required approximately 400 hr on a 12-core Xeon X5690 running at 3.47 GHz, and required 128 GB RAM. However, this database can easily be appended as new molecules become available, and the clustering time can be amortized over future queries. It is worth noting that this preprocessing of molecular graphs can cause the triangle inequality to be violated; while the distance function is a metric, the clustering does not respect that metric. Ammolite can be readily incorporated into existing analysis pipelines for high-throughput drug screening.

Our entropy-scaling framework can be applied to PubChem because it has both low fractal dimension and low metric entropy. In particular, we determined the mean local fractal dimension of PubChem to be approximately 0.2 in the neighborhood between 0.2 and 0.4 Tanimoto distance and approximately 1.9 in the neighborhood between 0.4 and 0.5. The expected speedup is measured by the ratio of database size to metric entropy, which, for PubChem, is approximately 11:1. This is not taking into account the clustering according to molecule size, which further reduces the search space.

Because SMSD is not computationally tractable on the entire PubChem database, we benchmarked Ammolite against SMSD on a subset of 1 million molecules from PubChem. Since SMSD's running time should scale linearly with the size of the database, we extrapolated the running time of SMSD to the entire PubChem database. Benchmarking Ammolite and SMSD required 60 GB RAM and used 12 threads, although Ammolite's search, used normally, requires <20 GB RAM. For these benchmarks, we used five randomly chosen query molecules with at least two rings (PubChem IDs 1504670, 19170294, 28250541, 4559889, and 55484477), as well as five medically interesting molecules chosen by hand (adenosine triphosphate [atp], clindamycin, erythromycin, teixobactin, and thalidomide). We also used SMSD as a gold standard against which we measured Ammolite's recall.

Ammolite achieves an average of 92.5% recall with respect to SMSD (Table 1). This recall is brought down by one poorly performing compound, PubChem ID 1504670, with only 62.5% recall, but is otherwise over 80%. Furthermore, Ammolite's speed gains with respect to SMSD grow as the database grows (Table 1).

Application to Metagenomics

Metagenomics is the study of genomic data sequenced directly from environmental samples. It has led to improved

Table 1. Benchmarks of Ammolite versus SMSD on Databases of 1 Million Molecules and 47 Million Molecules

PubChem ID and Benchmark	SMSD (Hours)	Ammolite (Hours)	Speedup	Recall (%)
Ammolite Benchmark on Database of 1 Million Molecules				
5957 (atp)	4.4	0.14	31	81
446598 (clindamycin)	18.7	1.5	11.7	90
12560 (erythromycin)	849.6	3.0	279.2	91
86341926 (teixobactin)	618.5	2.3	265.5	100
5426 (thalidomide)	48.9	0.81	60.4	100
1504670	8.1	0.8	10.3	62.5
19170294	31.3	0.8	39.7	100
28250541	43.3	4.8	9.0	100
4559889	108.8	2.7	41.0	100
55484477	23.3	2.5	9.1	100
Ammolite Benchmark on Entire PubChem Database as of October 2013				
5957 (atp)		4.1	51.3	
446598 (clindamycin)		28.4	14.5	
12560 (erythromycin)		79.1	512.9	
86341926 (teixobactin)		96.5	305.9	
5426 (thalidomide)		29.2	80.0	
1504670		4.6	84.4	
19170294		6.0	247.4	
28250541		38.9	53.2	
4559889		57.3	90.7	
55484477		35.5	31.4	

See also Figure S2.

Table 2. MICA Running Time in Minutes

BLASTX	RapSearch2	DIAMOND	MICA-DIAMOND	MICA-BLASTX
58215 (1561.8)	206 (5.4)	54 (1.1)	15.6 (0.5)	21.9 (1.7)

SDs are in parentheses. Dataset is the American Gut Microbiome Project read sets ERR335622, ERR335625, ERR335631, ERR335635, and ERR335636.

understanding of how ecosystems recover from environmental damage (Tyson et al., 2004) and how the human gut responds to diet and infection (David et al., 2014). Metagenomics has even provided some surprising insights into disorders such as autism spectrum disorder (Macfabe, 2012).

BLASTX (Altschul et al., 1990) is widely used in metagenomics to map reads to protein databases such as KEGG (Kanehisa and Goto, 2000) and NCBI's NR (Sayers et al., 2011). This mapping is additionally used as a primitive in pipelines such as MetaPhlAn (Segata et al., 2012), PICRUSt (Langille et al., 2013), and MEGAN (Huson et al., 2011) to determine the microbial composition of a sequenced sample. Unfortunately, BLASTX's runtime requirements scale linearly with the product of the size of the full read dataset and the targeted protein database and, thus, each year require exponentially more runtime to process the exponentially growing read data. These computational challenges are, at present, a barrier to widespread use of metagenomic data throughout biotechnology, which constrains genomic medicine and environmental genomics (Frank and Pace, 2008). For example, Mackelprang et al. (2011) reported that using BLASTX to map 246 million reads against KEGG required 800,000 CPU hours at a supercomputing center.

Although this is already a problem for major research centers, it is especially limiting for on-site analyses in more remote locations. In surveying the 2014 Ebola outbreak, scientists physically shipped samples on dry ice to Harvard University for sequencing and analysis (Gire et al., 2014). Even as sequencers become more mobile and can thus be brought on site, lack of fast Internet connections in remote areas can make it impossible to centralize and expedite processing (viz., the cloud); local processing on resource-constrained machines remains essential. Thus, a better scaling and accurate version of BLASTX raises the possibility of not only faster computing for large research centers but also of performing entirely on-site sequencing and desktop metagenomic analyses.

Recently, approaches such as RapSearch2 (Zhao et al., 2012) and DIAMOND (Buchfink et al., 2015) have provided faster alternatives to BLASTX. We have applied our entropy-scaling framework to the problem of metagenomic search and demonstrate MICA, a method whose software implementation provides an acceleration of DIAMOND by a factor of 3.5, and BLASTX by a factor of up to 3,700. This application illustrates the potential of entropy-scaling frameworks while providing a useful tool for metagenomic research. It can be readily incorporated into existing analysis pipelines (e.g., for microbial composition analysis using MEGAN). MICA clustering of the September 17, 2014, NCBI NR database (containing 49.3 million sequences) required 39 hr on a 12-core Xeon X5690 running at 3.47 GHz; it used approximately 84 GB of resident memory.

Our entropy-scaling framework can be applied to the NCBI's NR database because it, like PubChem, exhibits low fractal dimension and metric entropy. We determined the mean local fractal dimension of the NCBI's NR database, using sequence identity of alignment as a distance function, to be approximately 1.6, in the neighborhood between 70% and 80% protein sequence identity. The ratio of database size to metric entropy, which gives an indicator of expected speedup, is approximately 30:1. Indeed, the notion that protein sequence space exhibits structure, and lends itself to clustering, has precedent (Linial et al., 1997).

To evaluate the runtime performance of MICA, we tested it against BLASTX, RapSearch2 (Zhao et al., 2012), and DIAMOND (Buchfink et al., 2015). On five read sets (ERR335622, ERR335625, ERR335631, ERR335635, and ERR335636) totaling 207,623 151-nucleotide (nt) reads from the American Gut Microbiome Project, we found that MICA provides measurable runtime improvements over DIAMOND with no further loss in accuracy (Table 2) and substantial runtime improvements over BLASTX. Notably, the mean running time for BLASTX was 58,215 min, while MICA took an average of 15.6 min, a speedup of 3,724×. MICA uses DIAMOND for its coarse search and can use either DIAMOND or BLASTX for its fine search.

We also evaluated MICA using BLASTX for both the coarse and the fine search; this approach performed slightly slower

Table 3. MICA Accuracy against BLASTX

RapSearch2	DIAMOND	MICA-DIAMOND	MICA-BLASTX
79.5% (1.63)	90.4% (3.10)	90.4% (3.10)	90.4% (3.10)

SDs are in parentheses. Dataset is the American Gut Microbiome Project read sets ERR335622, ERR335625, ERR335631, ERR335635, and ERR335636.

than DIAMOND, requiring an average of 89 min, though it was somewhat more accurate, at 95.9% recall compared to DIAMOND's 90.4% recall. MICA using BLASTX for both coarse and fine searches relied on a query-side clustering (discussed in Supplemental Experimental Procedures); we note that the time spent performing query-side clustering is included here; without query-side clustering, this variant of MICA takes 2,278 min, a speedup of 25x over BLASTX.

MICA accelerates DIAMOND with no further loss in accuracy: 90.4% compared to unaccelerated BLASTX (Table 3). Experiments validating accuracy treated BLASTX as a gold standard. Since MICA accelerates DIAMOND using entropy-scaling techniques, false-positives with respect to DIAMOND are not possible, but false-negatives are. We report as accuracy the fraction of BLASTX hits that are also returned by MICA.

DIAMOND's clever indexing and alphabet reduction provide excellent runtime performance already, though its running time still scales linearly with database size. In contrast, as an entropy-scaling search, MICA will demonstrate greater acceleration as database sizes grow (Daniels et al., 2013). Moreover, MICA can use standard BLASTX for its fine search, which allows the user to pass arbitrary parameters to the underlying BLASTX call but which also comes at a small runtime penalty (40% in our testing). This option allows for additional BLAST arguments that DIAMOND does not support, such as XML output, which may be useful in some pipelines. Thus, MICA with BLASTX may be suitable for a wider variety of existing analysis pipelines.

Application to Protein Structure Search

The relationship between protein structure and function has been a subject of intense study for decades, and this strong link has been used for the prediction of function from structure (Hegyi and Gerstein, 1999). Specifically, given a protein of solved (or predicted) structure but unknown function, the efficient identification of structurally similar proteins in the Protein Data Bank (PDB) is critical to function prediction. Finding structural neighbors can also give insight into the evolutionary origins of proteins of interest (Yona et al., 1999; Nepomnyachiy et al., 2014).

One approach to finding structural neighbors is to attempt to align the query protein to all the entries in the PDB using a structural aligner, such as STRUCTAL (Subbiah et al., 1993), ICE (Shindyalov and Bourne, 1998), or Matt (Menke et al., 2008). However, performing a full alignment against every entry in the PDB is prohibitively expensive, especially as the database grows. To mitigate this, Budowski-Tal et al. (2010) introduced the tool FragBag, which avoids performing full alignments but rather describes each protein as a "bag of fragments," where each fragment is a small structural motif. FragBag has been reported as comparable to structural aligners such as STRUCTAL or ICE, and its bag-of-fragments approach allows it to perform comparisons much faster than standard aligners. Importantly for us, the bag of fragments is just a frequency vector, making FragBag amenable to acceleration through entropy scaling.

By first verifying that the local fractal dimension of PDB FragBag frequency vectors is low in most regimes ($d \approx 2 - 3$; Figure S3), we are given reason to think that this problem is amenable to entropy-scaling search. As an estimate of potential speedup, the ratio of PDB database size to metric entropy at the chosen cluster radii is, on average, ~10:1. We directly applied our entropy-scaling framework without any additional augmentation: esFragBag (entropy-scaling FragBag) is able to achieve an average speedup factor of 10 of the highly optimized FragBag with <0.2% loss in sensitivity and no loss in specificity.

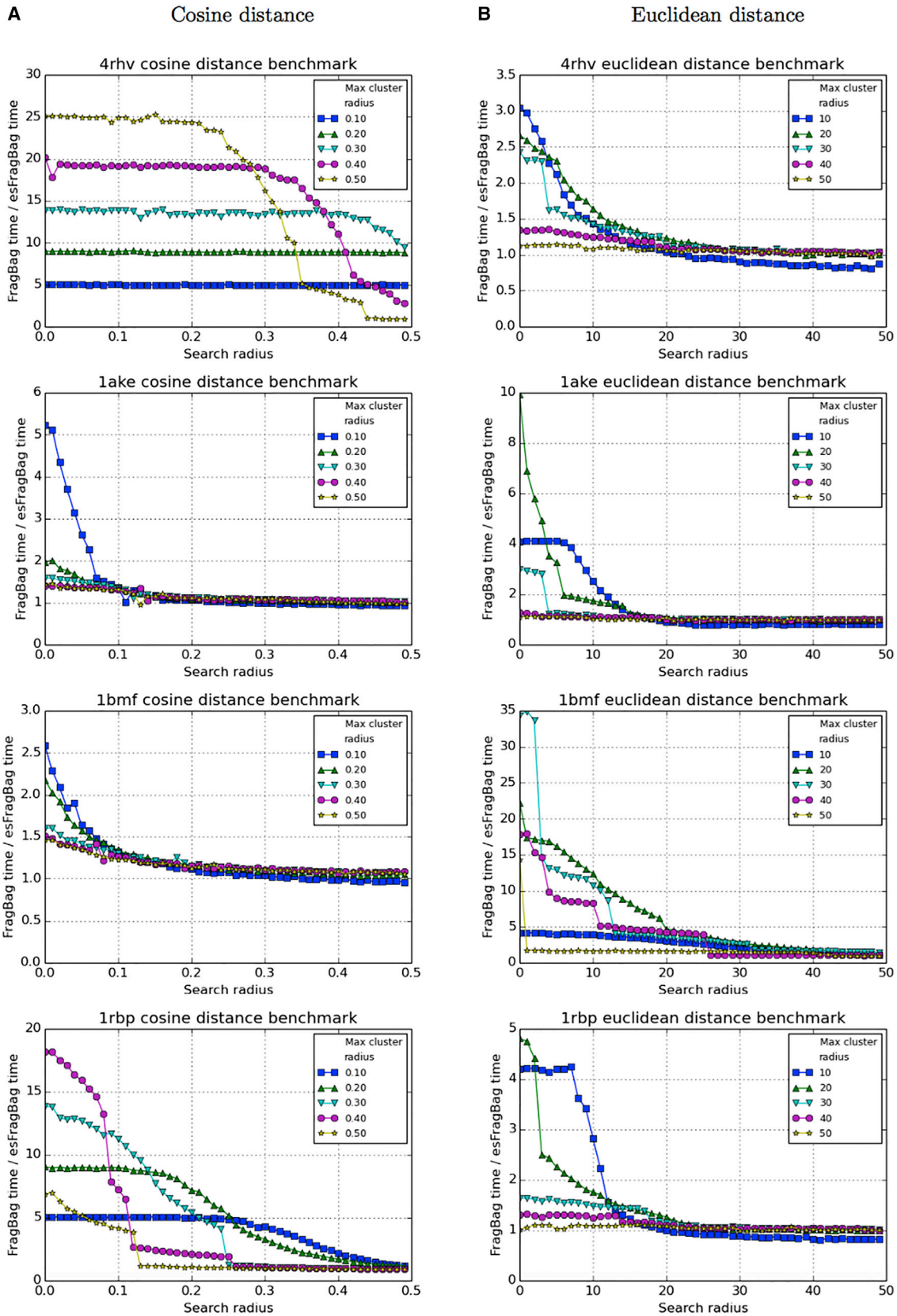
For this last example, we intentionally approach the application of entropy-scaling frameworks to FragBag in a blind manner, without using any domain-specific knowledge. Instead, we use the very same representation (bag of fragments) and distance functions (Euclidean and cosine distances) as FragBag, coupled with a greedy k-centers algorithm to generate the clustered representation. Note that this is in contrast to MICA and Ammolite, which both exploit domain knowledge to further improve performance. Thus, esFragBag only involves extending an existing codebase with new database generation and similarity search functions.

We investigated the increases in speed resulting from directly applying the entropy-scaling framework for both Euclidean and cosine distances and found that the acceleration is highly dependent on both the search radius and cluster radius (Figure 3). For cosine distance, we generated databases with maximum cluster radii of 0.1, 0.2, 0.3, 0.4, and 0.5. Then, for each query protein from the set {PDB: 4rhv, 1ake, 1bmf, 1rbp} (identified by PDB IDs), we ran both naïve and accelerated similarity searches with radii of $0.02i$, $\forall i \in \{0, \ldots, 49\}$. This test was repeated five times for each measurement, and the ratio of average accelerated versus naïve times is shown in Figure 3A. For Euclidean distance, we generated databases with maximum cluster radii of 10, 20, 25, 50, and 100.

Again, for each query protein drawn from the same set, we compared the average over five runs of the ratio of average accelerated versus naïve times (Figure 3B). The cluster generation required anywhere from 65 to 23,714 s, depending on the choice of radii (Tables 4 and 5) and no more than a small constant (<3) times as much memory as it takes to simply load the PDB database (no more than 2 GB RAM). Clustering used 20 threads on a 12-core Xeon X5690, while search used only one thread.

Not only is the acceleration highly dependent on both the search radius r and the maximum cluster radius r_c, but the choice of query protein also affects the results. We suspect that this effect is due to the geometry of protein fragment frequency space being very "spiky" and "star-like." Proteins that are near the core (and, thus, similar to many other proteins) show very little acceleration when our framework is used because the majority of the database is nearby, whereas proteins in the periphery have fewer neighbors and are, thus, found much more quickly. Changing the maximum cluster radius effectively makes more proteins peripheral proteins but at the cost of overall acceleration.

Naturally, as the search radius expands, it quickly becomes necessary to compare against nearly the entire database, destroying any acceleration. For the cosine space in particular,



(legend on next page)

Table 4. Cluster Generation Times for esFragBag: Cosine Distance

Radius	0.1	0.2	0.3	0.4	0.5
Times	21,037	11,088	7,409	5,288	3,921

Table 5. Cluster Generation Times for esFragBag: Euclidean Distance

Radius	10	20	30	40	50
Times	23,714	3,062	483	144	65

note that the maximum distance between any two points is 1, so once the coarse search radius of $r + r_c \geq 1.0$, there cannot ever be any acceleration as the fine search encompasses the entire database. Similarly, once the coarse search encompasses all (or nearly all) the clusters in Euclidean space, the acceleration diminishes to a factor of 1, and the overhead costs make the entropy-scaling framework perform worse than a naïve search. However, as we are most interested in proteins that are very similar to the query, the low-radius behavior is of primary interest. In the low-radius regime, esFragBag demonstrates varying though substantial acceleration (2–30×, averaging >10× for both distance functions for the proteins chosen) over FragBag.

It is instructive to note that because of the very different geometries of Euclidean versus cosine space, acceleration varies tremendously for some proteins, such as 4rhv and 1bmf, which display nearly opposite behaviors. Whereas there is nearly 30× acceleration for 4rhv in cosine space for low radius, and the same for 1bmf in Euclidean space, neither achieves better than ~2.5× acceleration in the other space.

Finally, while Euclidean distance is a metric—for which the triangle inequality guarantees 100% sensitivity—cosine distance is not. Empirically, however, for all of the queries we performed, we achieve >99.8% sensitivity (Table 6).

Application to Other Domains

We anticipate that our entropy-scaling approach will be useful to other kinds of biological datasets; applying it to new datasets will require several steps. Here, we provide a "cookbook" for applying our entropy-scaling framework to a new dataset. Given a new dataset, we first define what the high-dimensional space is. For metagenomic sequence data, it is the set of enumerable protein sequences up to some maximum length, while for small-molecule data, it is the set of connected chemical graphs up to some maximum size, and for protein structure data (using the FragBag model), it is the set of "bag-of-words" frequency vectors of length 400.

Given the high-dimensional space, we determine how database entries map onto points (for example, in the case of MICA, they are greedily broken into subsequences with a minimum length). Next, clustering can be implemented; a simple greedy clustering may suffice (as for esFragBag) but clustering of sequence data may be dramatically accelerated by using BLAST-style seed-and-extend matching (as used in MICA). Finally, coarse and fine search can be implemented; in many cases, existing tools may be used "out of the box," as with esFragBag and MICA. With MICA, we note that coarse search by default uses DIAMOND, while fine search provides a choice of DIAMOND or BLASTX. With Ammolite, we used the SMSD library, but incorporated it into our own search tool.

DISCUSSION

We have introduced an entropy-scaling framework for accelerating approximate search, allowing search on large omics datasets to scale, even as those datasets grow exponentially. The primary advance of this framework is that it bounds both time and space as functions of the dataset entropy (albeit using two different notions of entropy: metric entropy bounds time, while information-theoretic entropy bounds space). We proved that runtime scales linearly with the entropy of the database, but we also show (see "Theory" in Supplemental Experimental Procedures) that under certain additional constraints, this entropy-scaling framework permits a compressed representation on disk. This compression is particularly applicable in the case of metagenomic analysis, where the collection of read data presents a major problem for storage and transfer. Although we did not optimize for on-disk compression in any of our applications, choosing instead to focus on search speed, implementing this compression is feasible using existing software tools and libraries such as Blocked GZip (BZGF); each cluster would be compressed separately on disk.

Furthermore, we have justified and demonstrated the effectiveness of this framework in three distinct areas of computational molecular biology, providing the following open-source software: Ammolite for small-molecule structure search, MICA for metagenomic analysis, and esFragBag for protein structure search. All of our software is available under the GNU Public License, and not only can the tools we are releasing be readily plugged into existing pipelines, but the code and underlying methods can also be easily incorporated into the original software that we are accelerating.

The reason for the speedup is the combination of low fractal dimension and low metric entropy. Low fractal dimension ensures that runtime is dominated by metric entropy. The size of the coarse database provides an estimate of metric entropy. Furthermore, we can directly measure the local fractal dimension of the database by sampling points from the database and looking at the scaling behavior of the number of points contained in spheres of increasing radii centered on those sampled points. We have shown that for three domains within biological data science, metric entropy, and fractal dimension are both low.

Figure 3. Scaling Behavior of esFragBag

(A and B) EsFragBag benchmarking data with parameters varied until the acceleration advantage of esFragBag disappears. As search radius increases, the fraction of the database returned by the coarse search increases, ultimately returning the whole database. Unsurprisingly, when returning the whole database in the coarse search results, there are no benefits to using entropy-scaling frameworks. (A) Cosine distance gives on the whole better acceleration but results in >99.8% sensitivity, whereas (B) Euclidean distance as a metric is guaranteed by the triangle inequality to get 100% sensitivity.

See also Figure S3.

Table 6. Average Sensitivity of esFragBag Compared to FragBag when Using Cosine Distance for the Trials described in Figure 3A

	Query protein			
Cluster Radii	PDB: 4RHV	PDB: 1AKE	PDB: 1BMF	PDB: 1RBP
0.10	1	0.999840	0.998490	0.999950
0.20	1	0.999918	0.999001	0.999978
0.30	1	0.999926	0.999649	1
0.40	1	0.999974	0.999796	1
0.50	1	0.999984	0.999934	1

This table averages the sensitivities for each choice of search radii {0, 0.01, …, 0.49}. NB, no analogous table is given for Euclidean distance as the triangle inequality ensures perfect recall.

As discussed in the theoretical results, although the data live locally on a low-dimensional subspace, the data are truly high-dimensional globally. At small scales, biological data often live on a low-dimensional polytope (Hart et al., 2015). However, omics data are, by nature, comprehensive and include not just one but many such polytopes. Although each polytope can be individually projected onto a subspace using techniques such as PCA, the same projection cannot be used for all the polytopes at once because they live on different low-dimensional subspaces. Furthermore, as is the case with genomes, the low-dimensional polytopes are also often connected (e.g., through evolutionary history). Thus, collections of local projections become unwieldy. By using our clustering approach, we are able to take advantage of the existence of these low-dimensional polytopes for accelerated search without having to explicitly characterize each one.

A hierarchical clustering approach, rather than our flat clustering, has the potential to produce further gains (Loh et al., 2012). We have taken the first steps in exploring this idea here; the molecule size clustering in Ammolite can be thought of as an initial version of a multi-level or hierarchical clustering.

Entropy-scaling search is related to succinct, compressed, and opportunistic data structures, such as the compressed suffix array, the FM-index, and the sarray (Grossi and Vitter, 2005; Ferragina and Manzini, 2000; Conway and Bromage, 2011). However, these solve the problem of theoretically fast and scalable pattern matching (Box 1), whereas we solve, theoretically and practically, the much more general similarity search problem. An entropy-scaling search tree is also related to a metric ball tree (Uhlmann, 1991), although with different time complexity. Querying a metric ball tree requires $O(\log n)$ time, assuming the relatively uniform distribution of data points in a metric space. This distribution differs from the non-uniform distribution under which entropy-scaling search behaves well. In future work, we will investigate further acceleration of coarse search by applying a metric ball tree to the cluster representatives themselves; this approach may reduce the coarse search time to $O(\log k)$. This step, too, can be thought of as an additional level of clustering.

Other metric search trees can also be found in the database literature (Zezula et al., 2006), although, to our knowledge, they have not been explicitly applied to biological data science. The closest analog to entropy-scaling search trees is the M-tree (Ciaccia et al., 1997, 1998), which resembles a multi-level variation of our entropy-scaling search trees. However, the M-tree time-complexity analysis (Ciaccia et al., 1998) does not have a nice closed form and is more explicitly dependent on the exact distribution of points in the database. By using and combining the concepts of metric entropy and fractal dimension for our analysis, we are able to give an easier to understand and more intuitive, if somewhat looser, bound on entropy-scaling search complexity.

Entropy-scaling frameworks have the advantage of becoming proportionately faster and space efficient with the size of the available data. Although the component pieces (e.g., the clustering method chosen) of the framework can be either standard (as in esFragBag) or novel (as in Ammolite), the key point is that these pieces are used in a larger framework to exploit the underlying complex structure of biological systems, enabling massive acceleration by scaling with entropy. We have demonstrated this scaling behavior for common problems drawn from metagenomics, cheminformatics, and protein structure search, but the general strategy can be applied directly or with simple domain knowledge to a vast array of other problems faced in data science. We anticipate that entropy-scaling frameworks should be applicable beyond the life sciences, wherever physical or empirical laws have constrained data to a subspace of low entropy and fractal dimension.

EXPERIMENTAL PROCEDURES

Ammolite Small Molecule Search

Ammolite's clustering approach relies on structural similarity. We augmented the entropy-scaling data structure by using a clustering scheme based on molecular structural motifs instead of a distance function. Each molecule is "simplified" by removing nodes and edges that do not participate in simple cycles. Clusters are formed of molecules that are isomorphic after this simplified step. Each cluster can then be represented by a single molecular structure, along with pointers to "difference sets" between that structure and each of the full molecules in the cluster it represents. For both coarse search and fine search, we use the Tanimoto distance metric, defined as:

$$d(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{|G_1| + |G_2| - |mcs(G_1, G_2)|},$$

where *mcs* refers to the maximum common subgraph of two chemical graphs. The coarse search is performed in compressed space by searching the coarse database with the goal of identifying possible hits. The query molecule is simplified in exactly the same manner as the molecular database during clustering, and this transformed query graph is matched against the coarse database. To preserve sensitivity, this coarse search is performed with a permissive similarity score. Any possible hits—molecular graphs from the coarse database whose MCS to the transformed query molecule was within the similarity score threshold—are then reconstructed by following pointers to the removed atom and bond information and recreating the original molecules. Since the Tanimoto distance is used, we can bound the size of candidate molecules based on the size of the query molecule and the desired Tanimoto cutoff. Thus, a second level of clustering, at query time, based on molecule size, allows further gains in runtime performance. Finally, the fine search is performed against these decompressed possible hits that are within the appropriate size range based on the Tanimoto distance cutoff.

MICA Metagenomic Search

CaBLASTX's clustering approach relies on sequence similarity. We augmented the entropy-scaling data structure by using different distance functions for clustering and search. For clustering, we relied on sequence identity, while for search, we used the E-value measure that is standard for BLAST. All benchmarks were performed with an E-value of 10^{-7}. For coarse search, MICA uses the DIAMOND argument --top 60 in order to return all queries

with a score within 60% of the top hit. When MICA was tested using BLASTX for the coarse search, it used an E-value of 1,000. This seemingly surprisingly large coarse E-value is used because E-values are poorly behaved for short sequences; in sensitivity analysis, coarse E-values of 1 and 10 exhibited recall below 10%, and an E-value of 100 exhibited recall below 60%. Furthermore, during clustering (compression), we apply a preprocessing step that identified subsequences to be treated as distinct points in the database. We applied a reversible alphabet reduction to the protein sequences, which projected them into a subspace (Supplemental Experimental Procedures).

When applied to high-coverage, next-generation sequencing queries, caBLASTX can also perform clustering on the reads (Supplemental Experimental Procedures). In this instance, coarse search is performed by matching each representative query with a set of representative database entries. Fine search then matches the original queries within each cluster with the candidate database entries resulting from the coarse search.

esFragBag Protein Structure Search

In FragBag, the bag of fragments is essentially a term frequency vector representing the number of occurrences of each structural motif within the protein. FragBag turns out to be amenable to acceleration using an entropy-scaling data structure because much of the computation is spent in doing a similarity search on that frequency vector.

For the cluster generation, we trivially used a naïve randomized greedy two-pass approach. First, all proteins in the PDB were randomly ordered. Then in the final pass, proteins were selected as cluster centers if, and only if, they were not within a user-specified Euclidean distance r_c from an existing center (i.e., the first protein is always selected, and the second if further away than r_c from the first, etc.). Recall that this generation of cluster centers is the same as the one used to generate covering spheres in Figure 2; the covering spheres were overlapping, but we assigned every protein uniquely to a single cluster by assigning it to the nearest cluster center in the second pass.

Similarity search here was performed exactly as described earlier in the section "Entropy-Scaling Similarity Search," with no modification. For a given search query q and search radius r, a coarse search was used to find all cluster centers within distance $r + r_c$ of q. Then, all corresponding clusters were unioned into a set F. Finally, a fine search was performed over the set F to find all proteins within distance r of q.

SUPPLEMENTAL INFORMATION

Supplemental Information includes Supplemental Experimental Procedures, three figures, and three data files and can be found with this article online at http://dx.doi.org/10.1016/j.cels.2015.08.004.

AUTHOR CONTRIBUTIONS

Y.W.Y., N.M.D., and B.B. conceived the project. Y.W.Y., N.M.D., and B.B. developed the theoretical analyses. N.M.D. and D.C.D. implemented and benchmarked MICA and caBLASTX. D.C.D. implemented and benchmarked Ammolite, with help from N.M.D. and Y.W.Y. Y.W.Y. implemented and benchmarked esFragBag, with help from N.M.D. B.B. guided all research and provided critical advice on the study. Y.W.Y., N.M.D., and B.B. wrote the manuscript.

ACKNOWLEDGMENTS

Y.W.Y. is supported by a Hertz Foundation fellowship. N.M.D., D.C.D., and B.B. are supported by NIH grant GM108348. We thank Andrew Gallant for his implementation of FragBag. We thank Joseph V. Barrile for graphic design, Simon Ye for providing a bug fix to MICA, and Jian Peng for suggesting high-throughput drug screening as an application.

Received: March 18, 2015
Revised: June 21, 2015
Accepted: August 5, 2015
Published: August 26, 2015

REFERENCES

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. (1990). Basic local alignment search tool. J. Mol. Biol. *215*, 403–410.

Berger, B., Peng, J., and Singh, M. (2013). Computational solutions for omics data. Nat. Rev. Genet. *14*, 333–346.

Bolton, E.E., Wang, Y., Thiessen, P.A., and Bryant, S.H. (2008). Pubchem: integrated platform of small molecules and biological activities. Annu. Rep. Comput. Chem. *4*, 217–241.

Bredel, M., and Jacoby, E. (2004). Chemogenomics: an emerging strategy for rapid target and drug discovery. Nat. Rev. Genet. *5*, 262–275.

Buchfink, B., Xie, C., and Huson, D.H. (2015). Fast and sensitive protein alignment using DIAMOND. Nat. Methods *12*, 59–60.

Budowski-Tal, I., Nov, Y., and Kolodny, R. (2010). FragBag, an accurate representation of protein structure, retrieves structural neighbors from the entire PDB quickly and accurately. Proc. Natl. Acad. Sci. USA *107*, 3481–3486.

Bunke, H., and Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. Pattern Recognit. Lett. *19*, 255–259.

Cao, Y., Jiang, T., and Girke, T. (2008). A maximum common substructure-based algorithm for searching and predicting drug-like compounds. Bioinformatics *24*, i366–i374.

Ciaccia, P., Patella, M., and Zezula, P. *(*1997). M-tree: an efficient access method for similarity search in metric spaces. M. Jarke, M.J. Carey, K.R. Dittrich, F.H. Lochovsky, P. Loucopoulos, and M.A. Jeusfeld, eds. Proceedings of the 23rd International Conference on Very Large Data Bases (Morgan Kaufmann), pp. 426–435.

Ciaccia, P., Patella, M., and Zezula, P. (1998). A cost model for similarity queries in metric spaces. In Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (Association for Computing Machinery), pp. 59–68.

Conway, T.C., and Bromage, A.J. (2011). Succinct data structures for assembling large genomes. Bioinformatics *27*, 479–486.

Daniels, N.M., Gallant, A., Peng, J., Cowen, L.J., Baym, M., and Berger, B. (2013). Compressive genomics for protein databases. Bioinformatics *29*, i283–i290.

David, L.A., Materna, A.C., Friedman, J., Campos-Baptista, M.I., Blackburn, M.C., Perrotta, A., Erdman, S.E., and Alm, E.J. (2014). Host lifestyle affects human microbiota on daily timescales. Genome Biol. *15*, R89.

Falconer, K. (1990). Fractal Geometry: Mathematical Foundations and Applications (John Wiley & Sons).

Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., and El Abbadi, A. (2000). Vector approximation based indexing for non-uniform high dimensional data sets. Proceedings of the Ninth International Conference on Information and Knowledge Management (Association for Computing Machinery), pp. 202–209.

Ferragina, P., and Manzini, G. (2000, November 12–14). Opportunistic data structures with applications. Proceedings of the 41st Annual Symposium on Foundations of Computer Science (IEEE), pp. 390–398.

Frank, D.N., and Pace, N.R. (2008). Gastrointestinal microbiology enters the metagenomics era. Curr. Opin. Gastroenterol. *24*, 4–10.

Gire, S.K., Goba, A., Andersen, K.G., Sealfon, R.S., Park, D.J., Kanneh, L., Jalloh, S., Momoh, M., Fullah, M., Dudas, G., et al. (2014). Genomic surveillance elucidates Ebola virus origin and transmission during the 2014 outbreak. Science *345*, 1369–1372.

Grossi, R., and Vitter, J.S. (2005). Compressed suffix arrays and suffix trees with applications to text indexing and string matching. SIAM J. Comput. *35*, 378–407.

Hart, Y., Sheftel, H., Hausser, J., Szekely, P., Ben-Moshe, N.B., Korem, Y., Tendler, A., Mayo, A.E., and Alon, U. (2015). Inferring biological tasks using Pareto analysis of high-dimensional data. Nat. Methods *12*, 233–235.

Hegyi, H., and Gerstein, M. (1999). The relationship between protein structure and function: a comprehensive survey with application to the yeast genome. J. Mol. Biol. *288*, 147–164.

Huson, D.H., Mitra, S., Ruscheweyh, H.-J., Weber, N., and Schuster, S.C. (2011). Integrative analysis of environmental sequences using MEGAN4. Genome Res. *21*, 1552–1560.

Indyk, P., and Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In Proceedings of the 30th Annual ACM Symposium on Theory of Computing (Association for Computing Machinery), pp. 604–613.

Kahn, S.D. (2011). On the future of genomic data. Science *331*, 728–729.

Kanehisa, M., and Goto, S. (2000). KEGG: Kyoto encyclopedia of genes and genomes. Nucleic Acids Res. *28*, 27–30.

Kent, W.J. (2002). BLAT—the BLAST-like alignment tool. Genome Res. *12*, 656–664.

Langille, M.G., Zaneveld, J., Caporaso, J.G., McDonald, D., Knights, D., Reyes, J.A., Clemente, J.C., Burkepile, D.E., Vega Thurber, R.L., Knight, R., et al. (2013). Predictive functional profiling of microbial communities using 16S rRNA marker gene sequences. Nat. Biotechnol. *31*, 814–821.

Linial, M., Linial, N., Tishby, N., and Yona, G. (1997). Global self-organization of all known protein sequences reveals inherent biological signatures. J. Mol. Biol. *268*, 539–556.

Loh, P.-R., Baym, M., and Berger, B. (2012). Compressive genomics. Nat. Biotechnol. *30*, 627–630.

Macfabe, D.F. (2012). Short-chain fatty acid fermentation products of the gut microbiome: implications in autism spectrum disorders. Microb. Ecol. Health Dis. *23*, 23.

Mackelprang, R., Waldrop, M.P., DeAngelis, K.M., David, M.M., Chavarria, K.L., Blazewicz, S.J., Rubin, E.M., and Jansson, J.K. (2011). Metagenomic analysis of a permafrost microbial community reveals a rapid response to thaw. Nature *480*, 368–371.

Marx, V. (2013). Biology: the big challenges of big data. Nature *498*, 255–260.

Menke, M., Berger, B., and Cowen, L. (2008). Matt: local flexibility aids protein multiple structure alignment. PLoS Comput. Biol. *4*, e10.

Nepomnyachiy, S., Ben-Tal, N., and Kolodny, R. (2014). Global view of the protein universe. Proc. Natl. Acad. Sci. USA *111*, 11691–11696.

Pruitt, K.D., Tatusova, T., and Maglott, D.R. (2005). NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. Nucleic Acids Res. *33*, D501–D504.

Rahman, S.A., Bashton, M., Holliday, G.L., Schrader, R., and Thornton, J.M. (2009). Small molecule subgraph detector (SMSD) toolkit. J. Cheminform. *1*, 1–13.

Sayers, E.W., Barrett, T., Benson, D.A., Bolton, E., Bryant, S.H., Canese, K., Chetvernin, V., Church, D.M., DiCuccio, M., Federhen, S., et al. (2011). Database resources of the national center for biotechnology information. Nucleic Acids Res. *39*, D38–D51.

Schaeffer, S.E. (2007). Graph clustering. Comput. Sci. Rev. *1*, 27–64.

Segata, N., Waldron, L., Ballarini, A., Narasimhan, V., Jousson, O., and Huttenhower, C. (2012). Metagenomic microbial community profiling using unique clade-specific marker genes. Nat. Methods *9*, 811–814.

Shindyalov, I.N., and Bourne, P.E. (1998). Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. Protein Eng. *11*, 739–747.

Subbiah, S., Laurents, D.V., and Levitt, M. (1993). Structural similarity of DNA-binding domains of bacteriophage repressors and the globin core. Curr. Biol. *3*, 141–148.

Tao, T. (2008). Product set estimates for non-commutative groups. Combinatorica *28*, 547–594.

Tyson, G.W., Chapman, J., Hugenholtz, P., Allen, E.E., Ram, R.J., Richardson, P.M., Solovyev, V.V., Rubin, E.M., Rokhsar, D.S., and Banfield, J.F. (2004). Community structure and metabolism through reconstruction of microbial genomes from the environment. Nature *428*, 37–43.

Uhlmann, J.K. (1991). Satisfying general proximity/similarity queries with metric trees. Inf. Process. Lett. *40*, 175–179.

Ukkonen, E. (1985). Algorithms for approximate string matching. Inf. Control *64*, 100–118.

Weber, R., Schek, H.-J., and Blott, S. (1998, August 24–27). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. Proceedings of the 24th International Conference on Very Large Data Bases. (Morgan Kaufmann), pp. 194–205.

Yona, G., Linial, N., and Linial, M. (1999). ProtoMap: automatic classification of protein sequences, a hierarchy of protein families, and local maps of the protein space. Proteins *37*, 360–378.

Yu, Y.W., Yorukoglu, D., Peng, J., and Berger, B. (2015). Quality score compression improves genotyping accuracy. Nat. Biotechnol. *33*, 240–243.

Zezula, P., Amato, G., Dohnal, V., and Batko, M. (2006). Similarity Search: The Metric Space Approach, *Volume 32* (Springer Science & Business Media).

Zhao, Y., Tang, H., and Ye, Y. (2012). RAPSearch2: a fast and memory-efficient protein similarity search tool for next-generation sequencing data. Bioinformatics *28*, 125–126.