Compressive genomics

Po-Ru Loh, Michael Baym & Bonnie Berger

Algorithms that compute directly on compressed genomic data allow analyses to keep pace with data generation.

In the past two decades, genomic sequencing capabilities have increased exponentially[1–3], outstripping advances in computing power[4–8]. Extracting new insights from the data sets currently being generated will require not only faster computers, but also smarter algorithms. However, most genomes currently sequenced are highly similar to ones already collected[9]; thus, the amount of new sequence information is growing much more slowly.

Here we show that this redundancy can be exploited by compressing sequence data in such a way as to allow direct computation on the compressed data using methods we term 'compressive' algorithms. This approach reduces the task of computing on many similar genomes to only slightly more than that of operating on just one. Moreover, its relative advantage over existing algorithms will grow with the accumulation of genomic data. We demonstrate this approach by implementing compressive versions of both the Basic Local Alignment Search Tool (BLAST)[10] and the BLAST-Like Alignment Tool (BLAT)[11], and we emphasize how compressive genomics will enable biologists to keep pace with current data.

A changing environment

Successive generations of sequencing technologies have increased the availability of genomic data exponentially. In the decade since the publication of the first draft of the human genome (a 10-year, $400-million effort[1,2]), technologies[3] have been developed that can be used to sequence a human genome in 1 week for less than $10,000, and the 1000 Genomes Project is well on its way to building a library of over 2,500 human genomes[8].

These leaps in sequencing technology promise to enable corresponding advances in biology and medicine, but this will require more efficient ways to store, access and analyze large genomic data sets. Indeed, the scientific community is becoming aware of the fundamental challenges in analyzing such data[4–7]. Difficulties with large data sets arise in settings in which one analyzes genomic sequence libraries, including finding sequences similar to a given query (e.g., from environmental or medical samples) or finding signatures of selection in large sets of closely related genomes.

Currently, the total amount of available genomic data is increasing approximately tenfold every year, a rate much faster than Moore's Law for computational processing power (**Fig. 1**). Any computational analysis, such as sequence search, that runs on the full genomic library—or even a constant fraction thereof—scales at least linearly in time with respect to the size of the library and therefore effectively grows exponentially slower every year. If we wish to use the full power of these large genomic data sets, then we must develop new algorithms that scale sublinearly with data size (that is, those that reduce the effective size of the data set or do not operate on redundant data).

Sublinear analysis and compressed data

To achieve sublinear analysis, we must take advantage of redundancy inherent in the data. Intuitively, given two highly similar genomes, any analysis based on sequence similarity that is performed on one should have already done much of the work toward the same analysis on the other. We note that although efficient algorithms, such as BLAST[10], have been developed for individual genomes, large genomic libraries have additional structure: they are highly redundant. For example, as human genomes differ on average by only 0.1% (ref. 2), 1,000 human genomes contain less than twice the unique information of one genome. Thus, although individual genomes are not very compressible[12,13], collections of related genomes are extremely compressible[14–17].

This redundancy among genomes can be translated into computational acceleration by storing genomes in a compressed format that respects the structure of similarities and differences important for analysis. Specifically, these differences are the nucleotide substitutions, insertions, deletions and rearrangements introduced by evolution. Once such a compressed library has been created, it can be analyzed in an amount of time proportional to its compressed size, rather than having to reconstruct the full data set every time one wishes to query it.

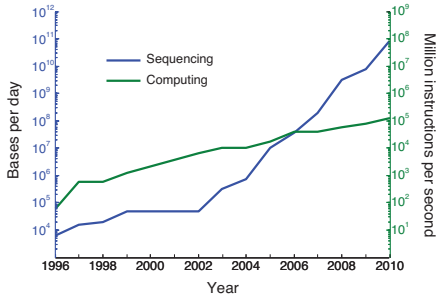

Figure 1 Sequencing capabilities versus computational power from 1996–2010. Sequencing capabilities are doubling approximately every four months, whereas processing capabilities are doubling approximately every eighteen. (Data adapted with permission from Kahn[4].)

Po-Ru Loh, Michael Baym and Bonnie Berger are in the Department of Mathematics and Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. Michael Baym is also in the Department of Systems Biology, Harvard Medical School, Boston, Massachusetts, USA. P.-R.L. and M.B. contributed equally to this work.
e-mail: bab@mit.edu or baym@mit.edu

Many algorithms exist for the compression of genomic data sets purely to reduce the space required for storage and transmission[12–15,17,18]. Hsi-Yang Fritz *et al.*[18] provide a particularly instructive discussion of the concerns involved. However, existing techniques require decompression before computational analysis. Thus, although these algorithms enable efficient data storage, they do not mitigate the computational bottleneck: the original uncompressed data set must be reconstructed before it can be analyzed.

There have been efforts to accelerate exact search through indexing techniques[16,19,20]. Although algorithms—such as Maq[21], Burrows-Wheeler Aligner (BWA)[22] and Bowtie[23]—already can map short resequencing reads to a few genomes quite well, compressive techniques will be extremely useful in the case of matching reads of unknown origin to a large database (say, in a medical or forensic context). Search acceleration becomes harder when one wishes to perform an inexact search (e.g., BLAST[10] and BLAT[11]) because compression schemes in general do not allow efficient recovery of the similarity structure of the data set.

As proof of principle for the underlying idea of compressive genomics, we present model compressive algorithms that run BLAST and BLAT in time proportional to the size of the nonredundant data in a genomic library (**Box 1**, **Fig. 2**, **Supplementary Methods**, **Supplementary Figs. 1–5** and **Supplementary Software**). We chose BLAST for a primary demonstration because it is widely used and also the principal means by which many other algorithms query large genomic data sets; thus any improvement to BLAST will immediately improve various analyses on large genomic data sets. Furthermore, the compressive architecture for sequence search we introduce here is tied not only to BLAST but also to many algorithms (particularly those based on sequence similarity).

Challenges of compressive algorithms

There are trade-offs to this approach. As more divergent genomes are added to a database, the computational acceleration resulting from compression decreases, although this is to be expected, as these data are less mutually informative. Although our compressive BLAST algorithm achieves over 99% sensitivity without substantial slowdown (**Fig. 3** and **Supplementary Figs. 6,7**), improvements in sensitivity necessarily involve losses in speed.

There is also a trade-off between achieving optimal data compression and accuracy of analysis (**Supplementary Fig. 6a**). This trade-off is fundamental to the problem of compressive algorithms for biology: in genomic analysis, one is interested in the probability of similar sequences occurring by chance rather than because of common ancestry, whereas compression ratios depend only on the absolute sequence similarity. For example, two sequences of 50% identity for over 1,000 bases are a strong BLAST hit, but admit no useful compression because the overhead would outweigh the savings. Although these two measures of sequence similarity are closely related,

Box 1 Compressive genomics using BLAST and BLAT

We describe versions of the widely used BLAST and BLAT algorithms that illustrate the compressive genomics paradigm. BLAST and BLAT search a genomic database to identify sequences that are similar to a given sequence. Our compressive algorithms have two phases: (i) compressing the database and (ii) searching the compressed data (**Supplementary Fig. 1**). The compression phase can be realized by various schemes. We used an approach based on edit script compression. The search phase can be implemented using nearly any existing search algorithm. We show the modularity of our approach by implementing compressive BLAST and BLAT search algorithms that can operate on the same compressed database.

Database compression. To compress data, we store only the differences between similar sequence fragments, rather than the complete, highly redundant sequences themselves. We implement this approach by scanning the nucleotide database and identifying sequence fragments sufficiently similar to previously seen fragments. Once identified, each fragment is replaced with a link to the original sequence and a compact list of differences. By default, we consider only fragments 300 base pairs or longer with at least 85% identity to previous fragments (**Supplementary Methods**). The initial data-compression phase only needs to be done once, and the compressed database can be updated incrementally if new data are added. This approach substantially reduces the storage required for many genomes (**Fig. 2a**).

The exact output of compression is dependent on the order in which the uncompressed data are examined; however, changing the order in which genomes are added to the library does not substantially affect the database size, compression speed, search speed or search accuracy (data not shown). For example, using our compressive BLAST algorithm, accuracy to hits in the first genome added to the database was perfect, and the accuracy of all subsequent hits was <1% lower.

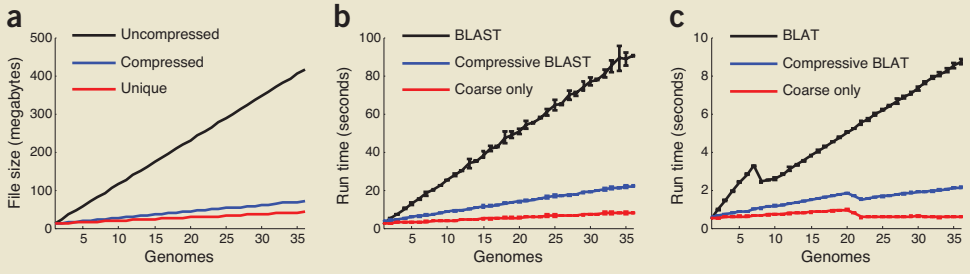

Figure 2 Results of compressive algorithms on up to 36 yeast genomes. (**a**) File sizes of the uncompressed, compressed with links and edits, and unique sequence data sets with default parameters. (**b**) Run times of BLAST, compressive BLAST and the coarse search step of compressive BLAST on the unique data ('coarse only'). Error bars, s.d. of five runs. Reported runtimes were on a set of 10,000 simulated queries. For queries that generate very few hits, the coarse search time provides a lower bound on search time. (**c**) Run times of BLAT, compressive BLAT and the coarse search step on the unique data ('coarse only') for 10,000 queries (implementation details in **Supplementary Methods**). Error bars, s.d. of five runs. BLAST and BLAT were both run with default parameters. The data shown represent differences between searches with 10,000 and 20,000 queries so as to remove the bias introduced by database construction time in BLAT. The anomalous decrease in run time with more data at 8 uncompressed genomes or 21 compressed genomes is a repeatable feature of BLAT with default parameters on these data.

Compressive BLAST. For the search phase, we implemented a two-step variant of BLAST. First, the algorithm uses standard BLAST to search the unique data (that is, data not replaced by links during compression) with a more permissive hit threshold (*E* value). Second, the algorithm traces

Box 1 Compressive genomics using BLAST and BLAT (continued)

links to determine potential hit regions in the full database and examines these potential hits with the original, stricter threshold. The initial 'coarse' search runs on the compressed data without decompression, yielding a run time proportional to the size of the compressed database. The second 'fine' alignment is done by locally decompressing only the potential hit regions until either a hit is determined or the region can be ruled out.

As the coarse search has a relaxed threshold, searches that hit repeat regions will result in many more coarse hits and thus burden the full computation. In practice, we mitigate this issue by masking repeat regions. For the results presented here, we used a coarse *E* value threshold of 10^{-20}, and always set the BLAST database size parameter to the size of the uncompressed database (**Supplementary Methods**).

To determine whether compression yields acceleration, we compressed 36 *Saccharomyces* sp. genomes[24] (**Fig. 2a**), four sets of bacterial genera and twelve *Drosophila* sp. fly genomes[25]. We simulated queries by sampling from the data set and adding mutations, producing a set of queries with many expected hits.

Compressive BLAST analysis of the yeast data set achieved a more than fourfold increase in speed with respect to a BLAST analysis. As expected, the advantage increased substantially with the number of genomes (**Fig. 2b**). We found a similar increase in speed for the microbial data sets (**Supplementary Figs. 2–4**). As our queries had many hits, the majority of the computation time (~73% for yeast) was spent on the fine search step, whereas for queries with few hits, the coarse step alone would be a more accurate predictor of run time. We expect that much faster fine search times can be achieved with an optimized fine search algorithm; our implementation simply runs BLAST a second time on potential hit regions.

For the fly species, although we achieved a large increase in search speed for the closely related *D. melanogaster*, *D. simulans* and *D. sechellia* genomes, the gains diminished as we included more distant cousins (**Supplementary Table 1**). In general, the run time of our compressive technique scales linearly with respect to the size of the nonredundant component of the library (which we expect to be a diminishing proportion), and linearly in the number of coarse hits.

Compressive BLAT. To implement a compressive version of the faster BLAT algorithm, we substituted BLAT for BLAST in the coarse search step and used BLAT's local alignment algorithm for the fine search to ensure comparable results. We tested compressive BLAT on the same data as above using BLAT's minIdentity parameter for coarse and fine search thresholds (minIdentity = 80 and 90, respectively).

Our compressive approach achieved acceleration over BLAT comparable to our results from accelerating BLAST (**Figs. 2c** and **Supplementary Fig. 5**). Although the coarse search step in BLAT theoretically takes a constant amount of time, in practice the running time of BLAT on a database of many genomes scales linearly with database size owing to the existence of many more 10-mer seed matches found during a search. Compression accelerates this step by allowing BLAT to rule out families of spurious hits only once. The hits produced by compressive BLAT analysis had an overall 96% accuracy and 97% specificity with respect to a BLAT analysis. The hits found by one algorithm and not the other were overwhelmingly of weak similarity. Thus, although it did not produce precisely the same hits as BLAT, compressive BLAT obtained coverage of true hits similar to the performance of BLAT.

the difference is at the root of these trade-offs. However, sacrificing some accuracy of distant matches helps to achieve a dramatic increase in speed from compression.

As computing moves toward distributed and multiprocessor architectures, one must consider the ability of new algorithms to run in parallel. Although we expect that the primary method of parallelizing compressive genomic search algorithms will be to run queries independently, truly massive data sets will require single queries to be executed in parallel as well. In the algorithms presented in **Box 1**, queries can be parallelized by dividing the compressed library and link table among computer processors, although the exact gains from doing so will depend on the topology of the link graph on the uncompressed database.

To the extent that researchers restrict their analyses to small data sets (e.g., what could be generated in a single laboratory as opposed to a large sequencing center), existing noncompressive custom pipelines may be sufficiently fast in the short term. However, if one wishes to extend an analysis to a much larger corpus of sequencing data (perhaps several terabytes of raw data), noncompressive approaches quickly become computationally impractical. This is where compressive algorithms are useful for smaller research groups in addition to large centers.

Conclusions

Compressive algorithms for genomics have the great advantage of becoming proportionately faster with the size of the available data. Although the compression schemes for BLAST and BLAT that we presented yield an increase in computational speed and, more importantly, in scaling, they are only a first step. Many enhancements of our proof-of-concept implementations are possible; for example, hierarchical compression structures, which respect the phylogeny underlying a set of sequences, may yield additional long-term

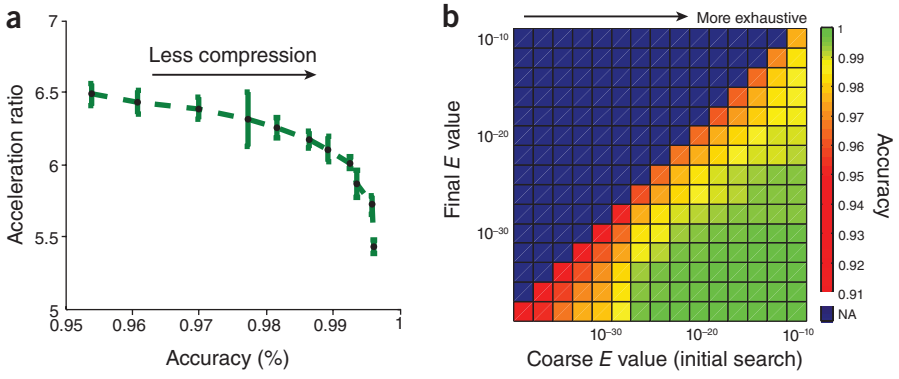

Figure 3 Trade-offs in compressive BLAST. (**a**) Speed versus accuracy as a function of the match identity threshold in database compression. From left to right, the points represent thresholds of 70–90%, with points every 2%. *E* value thresholds of 10^{-20} (coarse) and 10^{-30} (fine) were used. (**b**) Accuracy as a function of coarse and fine *E* value thresholds. Data presented are from runs on the combined microbial data set (yeast genomes and those of four bacterial genera) with search queries drawn randomly from the combined library and then mutated. NA, inapplicable parameter choices, as the coarse *E* value should always be larger than the fine one.

performance gains. Moreover, analyses of such compressive structures will lead to insights as well. As sequencing technologies continue to improve, the compressive genomic paradigm will become critical to fully realizing the potential of large-scale genomics.

Software is available at http://cast.csail.mit.edu/.

Editor's note: This article has been peer-reviewed.

Note: Supplementary information is available at http://www.nature.com/doifinder/10.1038/nbt.2241.

ACKNOWLEDGMENTS

We thank J. Kelner, E. Demaine, G. Church, X.R. Bao, M. Schnall-Levin, Z. Albertyn, M. Lipson and E. Lieberman-Aiden for helpful discussions and comments, and L. Gaffney for assistance improving the figures. P.-R.L. acknowledges support from the National Defense Science and Engineering Graduate and US National Science Foundation Fellowships. M.B. acknowledges support from the Fannie and John Hertz Foundation and the National Science Foundation Mathematical Sciences Postdoctoral Research Fellowship.

COMPETING FINANCIAL INTERESTS

The authors declare no competing financial interests.

1. Lander, E.S. *et al. Nature* **409**, 860–921 (2001).
2. Venter, J.C. *et al. Science* **291**, 1304–1351 (2001).
3. Kircher, M. & Kelso, J. *Bioessays* **32**, 524–536 (2010).
4. Kahn, S.D. *Science* **331**, 728–729 (2011).
5. Gross, M. *Curr. Biol.* **21**, R204–R206 (2011).
6. Huttenhower, C. & Hofmann, O. *PLoS Comput. Biol.* **6**, e1000779 (2010).
7. Schatz, M., Langmead, B. & Salzberg, S. *Nat. Biotechnol.* **28**, 691–693 (2010).
8. 1000 Genomes Project data available on Amazon Cloud. NIH press release, 29 March 2012.
9. Stratton, M. *Nat. Biotechnol.* **26**, 65–66 (2008).
10. Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. *J. Mol. Biol.* **215**, 403–410 (1990).
11. Kent, W.J. *Genome Res.* **12**, 656–664 (2002).
12. Grumbach, S. & Tahi, F. *J. Inf. Process. Manag.* **30**, 875–886 (1994).
13. Chen, X., Li, M., Ma, B. & Tromp, J. *Bioinformatics* **18**, 1696–1698 (2002).
14. Christley, S., Lu, Y., Li, C. & Xie, X. *Bioinformatics* **25**, 274–275 (2009).
15. Brandon, M.C., Wallace, D.C. & Baldi, P. *Bioinformatics* **25**, 1731–1738 (2009).
16. Mäkinen, V., Navarro, G., Sirén, J. & Välimäki, N. in *Research in Computational Molecular Biology*, vol. 5541 of *Lecture Notes in Computer Science* (Batzoglou, S., ed.) 121–137 (Springer Berlin/Heidelberg, 2009).
17. Kozanitis, C., Saunders, C., Kruglyak, S., Bafna, V. & Varghese, G. in *Research in Computational Molecular Biology*, vol. 6044 of *Lecture Notes in Computer Science* (Berger, B., ed.) 310–324 (Springer Berlin/Heidelberg, 2010).
18. Hsi-Yang Fritz, M., Leinonen, R., Cochrane, G. & Birney, E. *Genome Res.* **21**, 734–740 (2011).
19. Mäkinen, V., Navarro, G., Sirén, J. & Välimäki, N. *J. Comput. Biol.* **17**, 281–308 (2010).
20. Deorowicz, S. & Grabowski, S. *Bioinformatics* **27**, 2979–2986 (2011).
21. Li, H., Ruan, J. & Durbin, R. *Genome Res.* **18**, 1851–1858 (2008).
22. Li, H. & Durbin, R. *Bioinformatics* **25**, 1754–1760 (2009).
23. Langmead, B., Trapnell, C., Pop, M. & Salzberg, S. *Genome Biol.* **10**, R25 (2009).
24. Carter, D.M. *Saccharomyces* genome resequencing project. Wellcome Trust Sanger Institute <http://www.sanger.ac.uk/Teams/Team118/sgrp/> (2005).
25. Tweedie, S. *et al. Nucleic Acids Res.* **37**, D555–D559 (2009).