CentroidBLAST: Accelerating Sequence Search via Clustering

Wu-chun Feng, Konstantinos Krommydas, Liqing Zhang
Department of Computer Science
Virginia Tech
{wfeng, kokrommy, lqzhang}@vt.edu

Abstract

BLAST, short for Basic Local Alignment Search Tool, searches for regions of local similarity between a query sequence and a large database of DNA or amino-acid sequences. It serves as a fundamental tool to many discovery processes in bioinformatics and computational biology, including inferring functional and evolutionary relationships between sequences, identifying members of gene families, and phylogenetic profiling. Consequently, researchers have spent many decades making local alignment search (such as BLAST) more efficient, both with respect to speed and accuracy.

In this paper, we present our approach for more efficient sequence search, which we dub CentroidBLAST. CentroidBLAST first works on a *representative* fraction of the original database, where each representative serves as a "centroid" of similar sequences. A centroid's cluster of sequences is then searched only if its representative sequence is a similar match to the query sequence. This approach delivers as much as a 6.85-fold speed-up over NCBI BLAST. In addition, we analyze different aspects of CentroidBLAST, including execution time, biological significance of resulting alignments, selection of e-value cut-off, and effect of database compression.

1 Introduction

Sequence search is a fundamental process in bioinformatics and computational biology. Informally, it can be defined as finding similarities between a *query sequence* (i.e., an unknown string of DNA bases or amino acids) and a *subject sequence* (i.e., a sequence in the database that may have *known* origin and functionality). Identifying similarities between DNA or protein sequences serves as a *proxy* for detecting similarities in function and structure between sequences, thus providing clues on the evolutionary history and origin of *unknown* sequences, for example. Other uses include phylogenetic profiling and identifying members of gene families.

Our work addresses the need for fast and accurate sequence search using commodity off-the-shelf hardware. In contrast to other alternatives, our approach uses the widely-trusted NCBI-BLAST and CD-HIT at its core. More importantly, we explore the search space for sequence search, with respect to parameters such as e-value cut-offs and clustering similarity threshold.

Our approach, and more generally, approaches based on compressive genomics, have the potential to reduce the need for supercomputers, at least for small- or medium-sized research facilities. They also help to make *in-house* analysis of sequence data a feasible alternative. Though we do not employ graphics processing units (GPUs) or other accelerators (such as Intel Xeon Phi) in our prototype implementation, the overall concept can be applied in such environments.

The rest of the paper is organized as follows. Section 2 presents our related work while Section 3 presents our representative-based approach and discusses the implementation of the core functionalities of such an approach in the context of our CentroidBLAST prototype. In Section 4, we present the results and discuss the efficiency of our approach with respect to various parameters. Section 5 concludes the paper, and Section 6 highlights opportunities for future work.

2 Related Work

Many algorithms have been proposed in order to facilitate efficient sequence search. Early sequencing technologies (i.e., Sanger sequencing) produced data at a significantly slower pace than what we are able to achieve today with the advent of *next-generation sequencing* (NGS) [20]. However, even before the advent of NGS, biological data analysis was slow. Specifically, *exact* alignment algorithms based on dynamic programming, such as Smith-Waterman [22] or Needleman-Wunsch [21], eventually became obsolete in favor of *heuristic-based* ones, such as BLAST [4], which perform much faster than either Smith-Waterman or Needleman-Wunsch. However, this faster speed using heuristics came at the expense of *reduced sensitivity* (i.e., increased possibility for less accurate results).

By the mid-2000s, with biological sequence databases growing exponentially in size and processor clock frequencies stalling out, researchers added *parallel computing* to the mix in order to accelerate the heuristic-based sequence-search algorithms, e.g., mpiBLAST [5, 13, 15]. These efforts also resulted in optimizing the associated parallel algorithms for the underlying hardware architectures [6, 7, 8, 9, 11, 16, 17, 23]. In addition, the emergence of accelerator-based computing systems gave rise to massively parallel GPU implementations of BLAST [24, 25]. Such parallel implementations require (1) the management and scheduling of tens, hundreds, and even thousands of CPU cores in order to deliver high performance in the case of mpiBLAST; (2) the mapping and optimization of parallel algorithms onto specialized hardware, such as CUDA-enabled GPUs, for the GPU implementations; or (3) the packaging of parallel algorithms to map onto a cloud infrastructure [19].

A complementary approach to parallelizing the search algorithms is based on the idea of *clustered sequence representation* [10] or *compressive genomics* [12, 18]. Instead of parallelizing or accelerating the computation itself, this approach aims to *reduce* the amount of data on which the computation needs to be performed by taking advantage of the inherent *redundancy* of DNA and protein databases, as explained below.

DNA and protein databases are inherently redundant. When performing a typical BLAST search, one needs to search a query sequence against the *entire* database of interest. Sequences in the database that bear similarity (over a certain threshold) either trigger a successful alignment (*hit*) or not. In either case, searching against a redundant database incurs redundant computation. To address this inefficiency, our CentroidBLAST approach starts by *clustering* similar sequences together and selecting *one* sequence (i.e., centroid) to represent each cluster. If an initial BLAST search triggers a hit between a query and a centroid, there is a conceptually high probability that it would trigger hits for other members of the cluster, too. If it does *not*, the inverse implication that no other member of that cluster would trigger a hit obviates the need for performing a BLAST search between the query and the rest of that cluster's sequences. This idea allows for potentially signficiant computational savings but depends on the database and query combination as well as the underlying clustering algorithm's efficiency. In the worst case, if a query triggers hits against all the centroids, then all the original database's sequences will be searched.

3 Approach

Here we describe our compressive genomics approach to sequence search, herein referred to as CentroidBLAST. A high-level schematic representation of the CentroidBLAST algorithm, including the necessary *preprocessing stage* and the actual CentroidBLAST *search stage*, is depicted in Figure 1.

3.1 Preprocessing Stage

The preprocessing stage ensures that the *original* sequence database (protein or DNA) is represented by a carefully selected set of representative sequences (i.e., centroids). The collection of these centroids constitutes what we call the *centroid database*. The basis of compressive genomics (or representative-based sequence search) relies on the fact that an unprocessed database, i.e., the original database as formed from the output of the sequencing process or a concatenation of databases of closely related species, are bound to exhibit certain amounts of redundancy. The problem of efficient clustering is a entire area of research in and of itself, and compressive genomics will largely rely on the output of such research.

Our CentroidBLAST preprocessing stage handles the creation of the centroid database by employing an open-source clustering algorithm called CD-HIT [14]. CD-HIT follows a greedy and incremental clustering approach, where the longest input sequence is selected as the first cluster representative and then subsequent sequences (sorted from longest to shortest) are compared to the current representatives and classified as either redundant or representatives of new clusters. It supports both protein and DNA clustering and provides a sufficient amount of parameterization. One of its main advantages is multithreading support, i.e., utilizing multiple compute threads on a multi-core CPU. Although the clustering step is the main part of a *once-per-database* procedure, it is very time-consuming and fast execution is a desirable feature.

After running CD-HIT, similar sequences (where similarity is defined by a user-defined *similarity threshold* parameter) are identified and grouped, and a sequence per cluster is used as this cluster's *representative*.[1] The *centroid database* is stored in FASTA format, and the clustering information is contained in a separate file that undergoes further processing, as discussed below.

The centroid database effectively represents the original sequence database, and its size is much smaller than the original database. The effectiveness of this representation and the size of the centroid database depends on the setting of the similarity threshold as well as on the efficiency of the clustering algorithm. To better understand these aspects, we examine the compression efficiency of CD-HIT in detail for different

[1] The terms *centroid* and *representative sequence* will be used interchangeably throughout the rest of this paper.

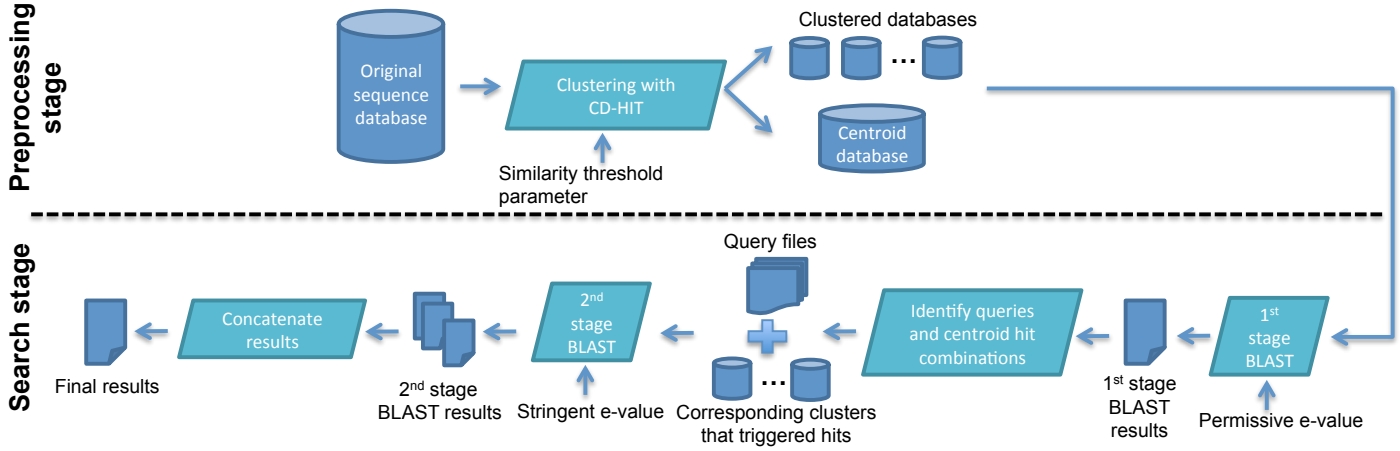

Figure 1: CentroidBLAST pipeline: preprocessing and CentroidBLAST search stages.

similarity thresholds and explore the trade-offs between compression efficiency and clustering execution time in Section 4.2.

Along with the centroid database, the original database information needs to be retained in a slightly different form. Specifically, in place of the original FASTA database, we store separate small databases that correspond to the sequence clusters, as generated by CD-HIT. These *cluster databases* as well as the centroid database must undergo formatting by NCBI-BLAST, using the corresponding NCBI-BLAST's *formatdb* tool. These databases are then used during the coarse-grained and fine-grained BLAST stages, respectively, of CentroidBLAST (as explained in more detail in Section 3.2).

The preprocessing stage only needs to be run *once per database*. Among the preprocessing stage's constituent procedures, CD-HIT clustering is by far the most time-consuming, i.e., *orders of magnitude* longer than the rest of the preprocessing.

3.2 Search Stage

Once preprocessing completes, CentroidBLAST is used in place of a regular BLAST search. CentroidBLAST conceptually consists of *two* BLAST searches: (1) a *coarse-grained* search stage (1st-stage BLAST), in which the BLAST search is conducted against the centroid database and with a more *permissive* e-value cut-off parameter and (2) a *fine-grained* search stage (2nd-stage BLAST), where each query that participated in a hit during the first stage is BLASTed against those cluster databases whose centroid triggered the hit with the query under consideration. For this second stage of CentroidBLAST search, a more *stringent* e-value cut-off is used.

As shown in Figure 1, our CentroidBLAST approach executes the following steps:

(1) Performs 1st-stage BLAST search with a more permissive e-value cut-off value and collects the hit results (i.e., combinations of queries and database sequences that align over a given e-value threshold). The centroid database and original query file (potentially consisting of multiple single queries) are used as inputs at this stage. The output is passed on the algorithmic pipeline in NCBI-BLAST's tabular format (*-m 8* parameter).

(2) Identifies the individual queries within a query file that triggered a hit against a certain centroid during the previous stage and splits them into multiple query files (one query per query file). The *sequence IDs* of all the centroid sequences that triggered a hit in the first stage are used as *pointers* to the corresponding clusters they represent. A separate 2nd-stage NCBI-BLAST search is subsequently performed between each of the above queries and the cluster databases previously identified. Those BLAST searches use a more stringent e-value cut-off than the 1st-stage BLAST.

(3) The second stage's BLAST results are concatenated in a single file and constitute the final CentroidBLAST output.

4 Results and Discussion

This section presents the results of our experiments with CentroidBLAST and a comparison against the results obtained from performing sequence alignment using NCBI-BLAST. Specifically, we focus on two

important aspects of sequence search: *execution time* and *accuracy*. Each of these two aspects depends on a number of factors (or parameters). In addition, these two aspects are expected to compete against one another. First, we perform a sensitivity analysis using such parameters in our experiments and present the results for each target metric in isolation. Then, we discuss the measured trade-offs between execution time and accuracy.

Prior to this, however, we begin by outlining our experimental set-up, followed by an analysis of the preprocessing stage and search stage for CentroidBLAST.

4.1 Experimental Setup

In this section we present our experimental setup, i.e., the software and hardware used to collect the results as well as the database and queries used.

4.1.1 Software

The results presented in this work are obtained by use of our CentroidBLAST prototype implementation. CentroidBLAST is designed as a bi-partite software, consisting of a *server-side* pipeline (used for preprocessing of the input databases) and a *client-side* pipeline (used for the actual sequence search process). Each of the pipelines utilizes custom Perl scripts performing the required operations. NCBI-BLAST [3] lies at the core of the client-side CentroidBLAST pipeline, performing the two required parameterized BLAST stages, while CD-HIT [1] is the application used in the preprocessing stage in the server-side CentroidBLAST pipeline that is responsible for finding similarities between the database's sequences and grouping them in clusters represented by a single sequence (*centroid*).

4.1.2 Database and Queries

For our proof of concept experiments we chose to use a reasonably big, yet not huge input database. The reason behind that lies in the detailed analysis we wanted to conduct which would be prohibitive time-wise if we were to use a much larger database (i.e., in the order of tens or hundreds of gigabytes). Moreover, the databases we used to form our bigger test database are *redundant* by their nature, i.e., contain information potentially including duplicates or near duplicates (as opposed to *non-redundant* databases). Specifically, we obtained three redundant protein databases from the European Bioinformatics Institute (part of the European Molecular Biology Laboratory- EMBL): the EPO database (429MB), JPO database (688MB), and the USPO database (581MB) [2]. The aforementioned databases contain proteins extracted from patent applications submitted to the European, Japan, and United States patent offices and consist of 459328, 416219 and 505583 sequences, respectively. These three databases were combined to a single database, which was the target input database in all our experiments.

We search these databases using six different sets of queries of random length, where each set consists of 100, 200, 300, 400, 500, 600 queries drawn randomly from the combined input database after being randomly mutated at a 20% rate. Due to lack of space for presenting the huge amount of data points (see Section 4), and for better visual clarity, in some experiments we only present results for the 300 queries file.

4.1.3 Hardware

The platform used to run all experiments is based on an Intel E5-2680 processor, which is a representative implementation of a high-performance, homogeneous multi-core system and is a processor with 16 cores in a package organized as two independent eight-core modules packaged together. Each core supports *two* hardware threads (for a total of 32 threads) and is clocked at 2.7 GHz.

4.2 Preprocessing Stage for CentroidBLAST

As mentioned in Section 3, the preprocessing stage is dominated by the clustering procedure (in terms of execution time). CD-HIT employs all available cores in a multi-core system and clustering run-time heavily depends on the *similarity threshold* parameter. We experiment with similarity threshold values ranging from 60-90% to evaluate the compression efficiency, as well as the execution time. The results are shown in Table 1. The first row corresponds to the original database, as described in Section 4.1, where each sequence effectively corresponds to a single cluster. One can observe the huge space savings even at the 90% similarity threshold, where the resulting centroid database consists of 17.98% of the original sequences. A similarity threshold of 80% increases the reduction at 11.94%. After that point we only observe diminishing returns, where a 70% and 60% similarity threshold yields a mere 2.07% and 1.78% further reduction, respectively. In terms of actual space savings the above reduction in number of clusters (i.e., sequences in the centroid database compared to sequences in the original database) corresponds to reductions starting at 80.67% and going up to 88.93%. Looking at the clustering execution time, we observe a steady increase going from 90% to 80% to 70%. At that point reducing

Table 1: Preprocessing stage: clustering and Perl scripts.

Similarity threshold	# Clusters	Cluster # savings (%)	Space savings (%)	Clustering time (minutes)	Auxiliary scripts time (seconds)
100%	6,910,715	N/A	N/A	N/A	N/A
90%	1,242,770	82.02	80.67	17	59
80%	825,362	88.06	84.66	22	61
70%	682,378	90.13	86.91	27	66
60%	559,689	91.91	88.93	147	72

the similarity threshold to 60% induces a huge increase of clustering execution time (by a factor of 5.44). The main reason lies in the internals of the CD-HIT clustering algorithm. Specifically, the *seed* used in comparing sequences for similarity and clustering needs to be reduced to a minimum of 4 letters for 60% similarity threshold, as opposed to 5 (for over 60%) to preserve comparable compression efficiency [14]. Intermediate auxiliary Perl scripts in the preprocessing CentroidBLAST stage only range between 59 and 72 seconds. In any case, since the preprocessing step only needs to be performed once for each target database, and for lack of space, we select for our experiments and present our findings for the centroid database and clustering output of running CD-HIT with the 60% similarity threshold.

4.3 Execution Time of CentroidBLAST: Search Stage

In this subsection we present our experiments that are related to the execution time of the sequence search through the CentroidBLAST pipeline. We experiment with an extensive range of coarse- and fine-grained stage e-value cut-offs, since these are the main execution parameters that affect total execution time. Specifically, we test e-values from 10^{-38} to 10^{-10} in many possible combinations. Allowed combinations require the first stage BLAST (or *coarse-grained*) e-value to be less stringent (i.e., larger) than the second stage's (or *fine-grained*). Figure 2 shows some of the values for the 120 possible combinations in the x-axis, while the y-axis depicts the relative execution time between the coarse- and fine-grained BLAST stages comprising CentroidBLAST. Multiple query files drawn randomly from the target database and randomly mutated at a certain level (as described in 4.1) exhibit similar behavior. In particular, while the coarse-grained BLAST stage is characterized by *minimal* variations in execution time (since the cut-off e-value only affects what final results are presented after the *same* amount of computation has been performed), the execution time of the fine-grained stage depends on the amount of *hits* resulting from the coarse-grained stage. The more permissive the coarse-grained e-value, the more hits, i.e., the more centroids are found. In turn, the clusters represented by these centroids formulate the databases to be searched in the fine-grained BLAST stage. Conceptually, the larger the databases to be searched the longer the fine-grained search, and the lower the ratio, as observed in Figure 2.

Figure 3 presents another aspect of execution time by comparing CentroidBLAST's speed-up over NCBI-BLAST. For visual clarity, we show results for the query file including 300 queries only, but our observations are quite similar for the rest of the test cases. In contrast to Figure 2, in Figure 3 we group the pair of e-values on the x-axis according to the coarse-grained stage e-value. This is to highlight that the decisive factor in attainable speed-ups lies in that first, coarse-grained BLAST search. The lowest speed-up observed here is 5.37x and the highest 6.85x, compared to the corresponding multi-threaded (i.e., parallel) NCBI-BLAST implementation that uses an e-value cut-off similar to the cut-off e-value of the fine-grained BLAST stage of CentroidBLAST.

4.4 Accuracy of CentroidBLAST Search Results

For any sequence search method accuracy of the results is arguably a very important aspect. In our case, we treat the original NCBI-BLAST results (obtained by using the same e-value as the fine-grained stage of CentroidBLAST) as the gold standard against which we calculate our method's accuracy (in terms of true positives, false positives, etc.) for each coarse-/fine-grained e-value combination. In all cases, only alignments with *identical* information (i.e., not only query and subject sequence, but start and end point, bit-score, e-value etc.) are considered true positives. Figure 4 shows the ROC curve for sequence search with CentroidBLAST using NCBI-BLAST as the gold standard. Different points represent different combinations of coarse- and fine-grained e-values for the two BLAST stages within CentroidBLAST. As such, one should not be confused by the fact that the ROC curve does not exhibit positive correlation, as it captures the TPR and FPR as a function of such combinations and encompasses the non-trivial interactions of the two BLAST stages within CentroidBLAST. As observed in the ROC curve all combinations achieve *true positive rate* (TPR) and *false positive rate* (FPR) more than 83% and less than 8%, respectively. More specifically, for about 2/3 of the combinations our method achieves TPR greater than 97% with FPR less than 2%. About 1/3 of the tested combinations yields TPR $>$ 99% with FPR $<$ 0.7%. Overall, CentroidBLAST exhibits satisfactory accuracy for most e-value combinations.

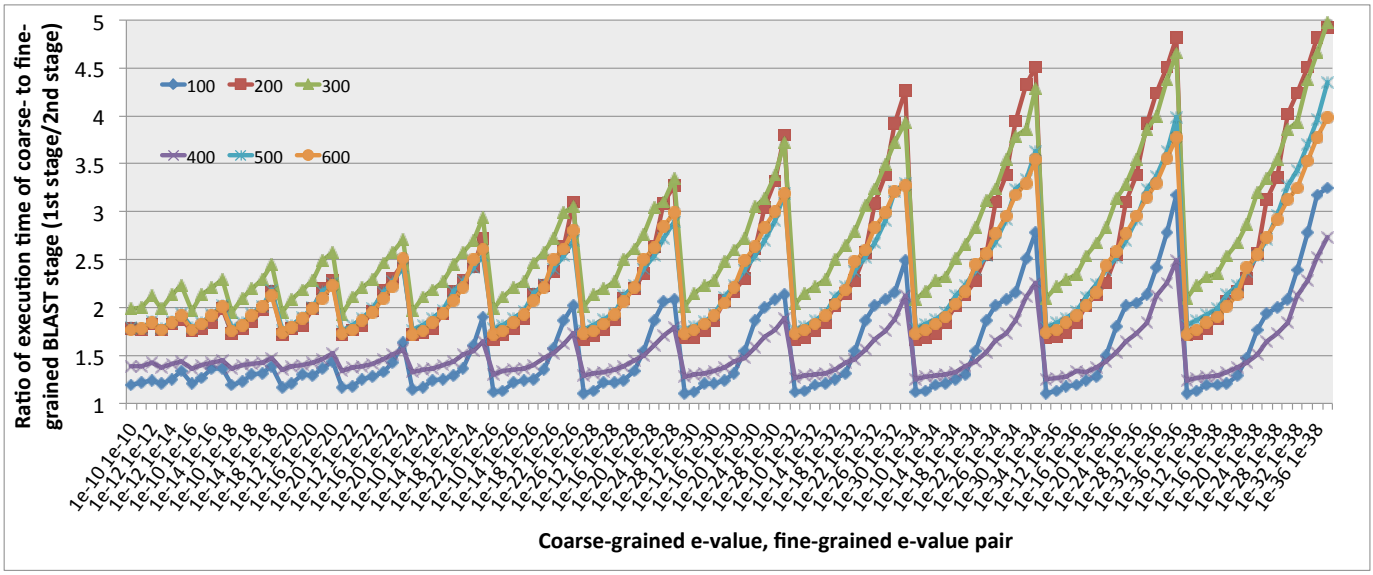

Figure 2: Ratio of execution time of coarse- to fine-grained BLAST stages within CentroidBLAST.

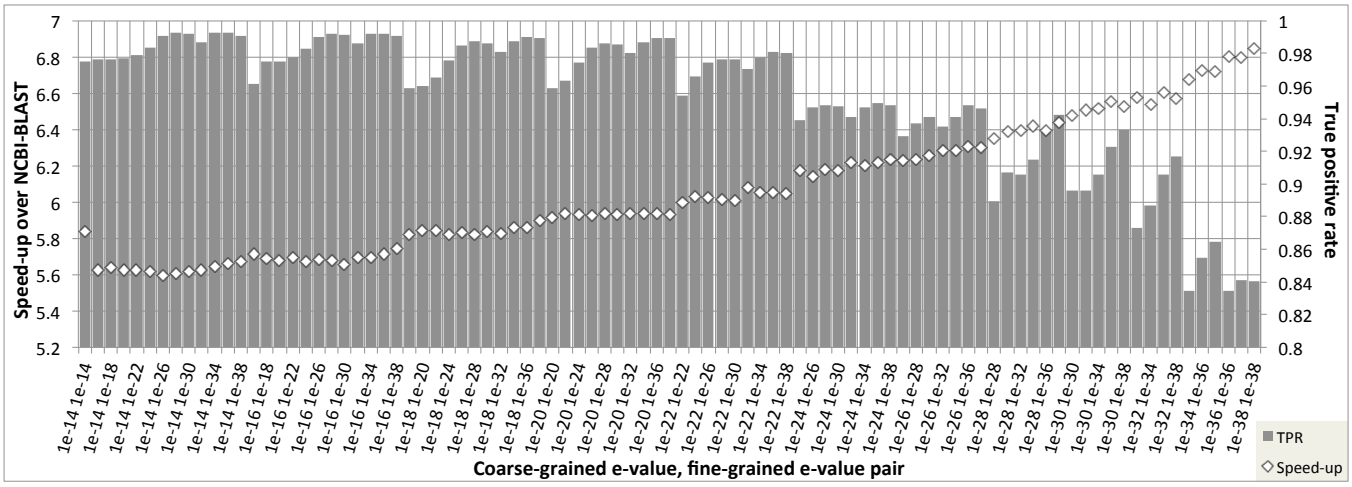

Figure 3: Speed-up over NCBI-BLAST for e-value cut-off similar to the fine-grained CentroidBLAST.

4.5 Execution Time vs. Accuracy Trade-off

In the previous sections, we discussed two important aspects: (1) the execution time of CentroidBLAST and (2) the accuracy of CentroidBLAST compared to NCBI-BLAST. We observed the importance of parameters like the similarity threshold of the clustering algorithm in the preprocessing stage's execution time and space savings and most importantly the effect of coarse- and fine-grained e-values in the execution time of the CentroidBLAST search. While our method has a proven ability to perform fast sequence alignment, as well as accurate alignment results (compared to NCBI-BLAST), we need to quantify the relationship (or *trade-off*) between execution speed and accuracy. Intuitively, one would expect those two targets to be competing and this trend is generally the case, as we observe in Figure 3. Specifically, to achieve the highest speed-up one has to sacrifice accuracy, in that many hits discovered by NCBI-BLAST are not identified as such by CentroidBLAST. Depending on the coarse- and fine-grained e-value pair selection, we can generally attain speed-ups approaching six-fold with a TPR $> 98\%$.

5 Conclusion

In this work, we demonstrated the effectiveness of the compressive-genomics concept. We presented a prototype implementation of an automated centroid-

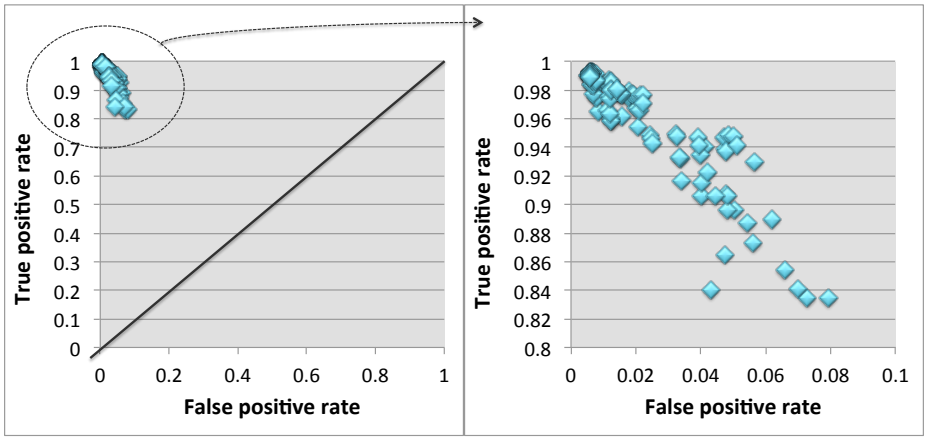

Figure 4: ROC curve for sequence search with CentroidBLAST with varying e-value combinations using NCBI-BLAST as the gold standard.

BLAST pipeline, by means of appropriate core and supplementary shell scripts, as well as their constituent Perl programs. The centroidBLAST framework we propose is built with a lego-like approach for being easily *adjustable* and with future *extensibility* in mind. While high-throughput next generation sequencing has resulted in huge amounts of sequencing data and typical BLAST searches have become notoriously time-consuming, our centroidBLAST approach delivered an up to 6.85x speed-up in our experimental set-up with minimal sacrifice in results' accuracy. Although our test protein database already demonstrates the effectiveness of centroidBLAST, much larger input databases (in the order of *hundreds* or *thousands* of GB of sequencing data) can potentially prove the centroidBLAST approach even more efficient. We presented a detailed analysis of the obtained results with respect to various parameters' selection, including clustering similarity threshold, different query sets, and most importantly cut-off e-value combinations for our two-stage BLAST approach at the core of the centroidBLAST pipeline.

6 Future Work

While our centroidBLAST prototype implementation serves as a reasonable proof of concept of our approach, there yet remains space for future work to further extend its utility and effectiveness. Experimentation with more and novel *clustering tools* is one of our primary concerns, as clustering efficiency in the centroidBLAST preprocessing step constitutes the basis for better centroidBLAST searches. Furthermore, adding support for *incremental clustering* is an important avenue of future research; by saving intermediate BLAST results for a query of interest, one can avoid repeating large amounts of computation, once a database is updated with new sequences. As new sequences are added to an existing database they are assigned to existing clusters. A new search (based on a saved BLAST result) needs to only take into account those new sequences that fall under the clusters whose centroid previously incurred a hit. Otherwise they can be disregarded altogether. Further research on *auto-tuning* the parameters mentioned earlier (clustering similarity threshold, etc.) for achieving the best results, either in terms of execution time or in terms of biological accuracy could be conducted, depending on the user's preference. To further enhance execution speed, with even less sacrifices in accuracy, one can employ parallel hardware, such as the many-core graphics processing unit (GPU) or other accelerators, such as Intel Xeon Phi. Finally, from a software engineering standpoint, while our script-based automated pipeline only supports the most important BLAST options, it can be trivially extended to support the full array of BLAST features/options and a *graphical user interface* (as opposed to a command line tool) would greatly enhance its usability.

Acknowledgments

This work was supported in part by the Institute for Critical Technology and Applied Science (ICTAS) and by NSF IIS-1247693.

References

[1] CD-HIT Website. `http://weizhong-lab.ucsd.edu/cd-hit/`.

[2] EMBL-EBI: Patent protein sequences. `http://www.ebi.ac.uk/patentdata/proteins`.

[3] NCBI-BLAST FTP server. `ftp://ftp.ncbi.nlm.nih.gov/blast/executables/release/LATEST/`.

[4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

[5] J. Archuleta, E. Tilevich, and W. Feng. A maintainable software architecture for fast and modular bioinformatics sequence search. In *IEEE Int'l Conf. on Software Maintenance*, 2007.

[6] P. Balaji, W. Feng, J. Archuleta, and H. Lin. ParaMEDIC: Parallel Metadata Environment for Distributed I/O and Computing. In *Supercomputing Storage Challenge Winner*, 2007.

[7] P. Balaji, W. Feng, J. Archuleta, H. Lin, R. Kettimuthu, R. Thakur, and X. Ma. Semantics-based distributed I/O for mpiBLAST. In *ACM SIGPLAN PPoPP*, 2008.

[8] P. Balaji, W. Feng, and H. Lin. Semantics-based distributed I/O with the ParaMEDIC framework. In *Proceedings of the ACM/IEEE Int'l Symposium on High Performance Distributed Computing*, Jun 23-27 2008.

[9] P. Balaji, W. Feng, H. Lin, J. Archuleta, S. Matsuoka, A. Warren, J. Setubal, E. Lusk, R. Thakur, I. Foster, D. Katz, S. Jha, K. Shinpaugh, S. Coghlan, and D. Reed. Distributed I/O with ParaMEDIC: Experiences with a worldwide supercomputer. In *Int'l Supercomputing Conf.*, 2008.

[10] M. Cameron, Y. Bernstein, and H. E. Williams. Clustered sequence representation for fast homology search. *Journal of Computational Biology : a Journal of Computational Molecular Cell Biology*, 14(5):594–614, June 2007.

[11] A. Ching, W. Feng, H. Lin, X. Ma, and A. Choudhary. Exploring I/O strategies for parallel sequence database search tools with S3aSim. In *Proceedings of the Int'l Symposium on High Performance Distributed Computing*, June 2006.

[12] N. M. Daniels, A. Gallant, J. Peng, L. J. Cowen, M. Baym, and B. Berger. Compressive genomics for protein databases. *Bioinformatics*, 29(13):i283–i290, 2013.

[13] A. E. Darling, L. Carey, and W. Feng. The design, implementation, and evaluation of mpiBLAST. In *ClusterWorld*, 2003.

[14] L. Fu, B. Niu, Z. Zhu, S. Wu, and W. Li. CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 2012.

[15] M. Gardner, W. Feng, J. Archuleta, H. Lin, and X. Ma. Parallel genomic sequence-searching on an ad-hoc grid: Experiences, lessons learned, and implications. In *SC*, 2006.

[16] H. Lin, P. Balaji, R. Poole, C. Sosa, X. Ma, and W. Feng. Massively parallel genomic sequence search on the Blue Gene/P architecture. In *SC*, 2008.

[17] H. Lin, X. Ma, W. Feng, and N. F. Samatova. Coordinating computation and I/O in massively parallel sequence search. *IEEE Transactions on Parallel and Distributed Systems*, 99, 2010.

[18] P. Loh, M. Baym, and B. Berger. Compressive genomics. *Nature Biotechnology*, 30(7):627–630, July 2012.

[19] A. Matsunaga, M. Tsugawa, and J. Fortes. CloudBLAST: Combining MapReduce and virtualization on distributed resources for bioinformatics applications. In *IEEE Fourth Int'l Conf. on eScience*, pages 222–229, Dec 2008.

[20] M. L. Metzker. Sequencing technologies - the next generation. *Nature Reviews. Genetics*, 11(1):31–46, January 2010.

[21] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March 1970.

[22] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, March 1981.

[23] O. Thorsen, K. Jiang, A. Peters, B. Smith, H. Lin, W. Feng, and C. Sosa. Parallel genomic sequence-search on a massively parallel system. In *ACM Int'l Conf. on Computing Frontiers*, May 2007.

[24] P. D. Vouzis and N. V. Sahinidis. Gpu-blast: Using graphics processors to accelerate protein sequence alignment. *Bioinformatics*, 2010.

[25] S. Xiao, H. Lin, and W. Feng. Accelerating protein sequence search in a heterogeneous computing system. In *2011 IEEE Int'l Parallel Distributed Processing Symposium*, May 2011.