5-2004

Indexing Genomic Databases

Gina Cooper

Wright State University - Main Campus

Michael L. Raymer

Wright State University - Main Campus, michael.raymer@wright.edu

Travis E. Doom

Wright State University - Main Campus, travis.doom@wright.edu

Dan E. Krane

Wright State University - Main Campus, dan.krane@wright.edu

Natsuhiko Futamura

Wright State University - Main Campus

Follow this and additional works at: <https://corescholar.libraries.wright.edu/knoesis>

Part of the [Bioinformatics Commons](#), [Communication Technology and New Media Commons](#), [Databases and Information Systems Commons](#), [OS and Networks Commons](#), and the [Science and Technology Studies Commons](#)

Repository Citation

Cooper, G., Raymer, M. L., Doom, T. E., Krane, D. E., & Futamura, N. (2004). Indexing Genomic Databases. *Proceedings of the Fourth IEEE Symposium on Bioinformatics and Bioengineering*, 587-591.
<https://corescholar.libraries.wright.edu/knoesis/921>

This Conference Proceeding is brought to you for free and open access by the The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) at CORE Scholar. It has been accepted for inclusion in Kno.e.sis Publications by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.

Indexing Genomic Databases

G. Cooper, M. Raymer, T. Doom, D. Krane and N. Futamura

Wright State University, 3640 Colonel Glenn Hwy, Dayton, OH 45434

[cooper.48, michael.raymer, travis.doom, dan.krane, natsuhiko.futamura]@wright.edu

Abstract

Current biological sequence comparison tools utilize full database searches to find approximate matches between a database and a query. A new approach to sequence comparisons can be performed by indexing the database using a novel indexing scheme. An indexed scheme can immediately eliminate highly mismatched sequences thereby improving performance and accuracy. iBlast is proposed as an indexed version of BLAST. In its initial implementation, iBlast uses a sequence-based index to catalog genomic databases in an NCR Teradata RDBMS. Several types of indexes and querying methods are explored to determine the most efficient solution utilizing the parallel nature of the Teradata system. Significant speedups were obtained and are explained in further detail in this paper. Future indexing methods based on prokaryotic and eukaryotic genome structures are also proposed.

1. Introduction

Indexing provides a method to decrease query evaluation costs by eliminating much of the data in the database and generating a much smaller answer set than exhaustive searching. Genomic databases continually increase in size making the use of exhaustive searches cumbersome and costly [4]. Molecular biologists perform several thousand queries per day and require speed, efficiency and accuracy in their results. As the number of sequences available in public databases grows, large scale sequence comparisons, including genome-to-genome comparisons, may provide new insights into the relationships among organisms and their genomes. There is an obvious need to improve query speed in genomic databases. This paper explores a new method of indexing to improve on the current search tools.

2. Index

Current search techniques increase query speed substantially, but at the cost of accuracy [1, 5]. This paper suggests an indexing scheme that will maintain the accuracy of current alignment tools, while performing

alignments much faster. Current search tools assume that all sequences in the database are a priori equally likely to be related to a query. Eliminating some of the data immediately promises to greatly improve query speed.

Due to the extent of the data involved in genomic databases, two passes will be needed. The first pass will eliminate much of the data based on the indexing, and the second pass will perform a Smith Waterman [8] alignment on the query results to determine the optimal answer. Due to the nature of databases, at this time only exact matches with no gaps are returned. Even so, because the alignment will only be needed on a portion of the data, query time will decrease.

A sequence based index was initially created. This index is comprised of a 16-mer word from the genomic database and a pointer to the location in the flat file where that word occurs. In addition, to reduce space and improve performance, the 16-mer word was converted into an integer. The conversion mechanism uses the following scheme: A=00, C=01, G=10, T=11. A 32-bit binary number is generated to represent each 16-mer, which is in turn converted into an integer. For example, the 4-mer word: AGCA is located at position 1144 in the database. This word is encoded as 00100100, which is equal to the decimal integer 36. Thus the record in the database will appear as (36, 1144). A unique primary index was created using the word and location. This initial implementation was coined "iBlast" for indexed BLAST [1]. iBlast was developed on an NCR Teradata relational database management system (RDBMS) [5]. The Teradata utilizes parallel processing to achieve fast and accurate answers to queries. Results of iBlast are described in detail in the implementation section of this paper.

3. Implementation

The proposed indexing scheme was implemented on an NCR Teradata WorldMark 4800 machine. This system has two nodes, where each node consists of 4 Intel Pentium 3 Xeon processors, 1 GB shared memory, and 72 GB disk space. The nodes are interconnected by a dual BYNET interconnection network supporting 800 Mbps of data bandwidth for each node. In addition, the nodes are connected to an external disk storage subsystem configured as a level-5 RAID (Redundant Array of

Inexpensive Disks) with 240 GB disk space [3]. The Teradata machine utilizes a complete relational database management system and parallel architecture. The Teradata utilizes Parsing Engine Processors (PEP) and Access Module Processors (AMP) to perform indexing and retrieval tasks. The AMPs store and retrieve distributed data in parallel and manage all data storage. Ideally, data should be divided evenly among the AMPs to allow for efficient retrieval of data. When a query is submitted, only those AMPs which contain the result data participate in the processing of the query. The AMPs return the data to the Message Processing Layer (BYNET) to merge the data for the client to view the result.

The Teradata machine utilizes both a primary index and a secondary index. Choice of the primary index is directly related to the performance of the Teradata machine. A hashing function is used for the data value in the primary index, and the resulting hash value is used to map that data to a specific AMP [5]. When the primary index is unique, row distribution is even, which allows for quick access. If the primary index is not unique, then the duplicate values are hashed to the same AMP, which will work harder than the other AMPs during a query. The secondary index allows direct access to rows in the data without requiring the primary index. The Teradata creates a subtable in which the primary index of the subtable is the value of the secondary index. The data in the subtable row is the hashed value of the primary index of the base table.

The implementation of the indexing scheme was carried out in two phases. The first phase consists of implementing a current exhaustive search tool, such as BLAST, on the NCR Teradata machine. This implementation is the iBlast project. The second phase encompasses the implementation of additional biological indices.

Three genomic databases were indexed and loaded onto the Teradata RDBMS. These include: *ecoli* (*E. coli* genomic nucleotide sequences), *yeast* (*Saccharomyces cerevisiae*) genomic nucleotide sequences), and *drosoph* (*Drosophila* genome provided by Celera and Berkeley *Drosophila* Genome Project (BDGP)). In addition, other test genomic databases of sizes 250kB, 500kB, 1000kB, 2000kB, and 4000kB were loaded. Several query tables were loaded as well ranging in size from 200 records to 30000 records. These tables were subsequences of *ecoli*.

A unique primary index was created for all three genomic databases consisting of a 16-mer nucleotide word converted to an integer ("num") and the location of that word in the flat file ("location"). All query tables used the same index as well.

Initial trials were performed to determine the most efficient method to query the database using SQL. Three methods were employed:

(1) using individual select statements of the form:

```
select * from iecoli
where num=395273;
select * from iecoli
where num=689032;
```

(2) using one select statement and the OR disjunctive operator:

```
select * from iecoli
where num=395273
OR num=689032;
```

(3) and joining the database and query table:

```
select * from iecoli, query200
where iecoli.num=query200.num;
```

The results were favorable for joining the tables as can be seen from figure 1.

Figure 1. Comparison of Select, OR, and Join Query Evaluation Techniques

Figure 2. Size of Database in Teradata RDBMS as Compared to Original Database Size

In addition to time constraints, space constraints must also be considered. The original size of a genomic database was plotted against the size of the database in the Teradata RDBMS (see figure 2). As can be seen from the plot, a linear relationship exists between original database size and Teradata RDBMS size. The Teradata RDBMS

size is approximately 22 times larger than the database due to the overhead of the index.

Several queries were tested against the three databases (ecoli, yeast, and drosoph), to determine approximate running times. Initial tests revealed apparently inconsistent behavior in smaller query sizes as can be seen from figure 3.

Figure 3. Query Times in Seconds Based on Database Size

To alleviate this problem, which was related to the distribution of the database data and the Teradata AMP scheduling algorithm, a secondary index was created on num (the 16-mer nucleotide word converted to an integer). This increased query speed greatly and removed the spikes and inconsistent behavior from smaller query sizes. This index was at most 100 times faster than queries performed without a secondary index (see figure 4). However, the addition of a secondary index doubled the size of the database on the RDBMS. Table 1 illustrates the size in MB of the three genomic databases.

Figure 4. Query Times in Seconds Based on Database Size Using a Secondary Index

Database	Before Secondary Index	After Secondary Index
Ecoli	102	260
Yeast	267	671
Drosoph	2700	6600

Table 1. Genomic Database Size Comparison in MB

By using the secondary index in addition to the unique primary index, the size of the database on the Teradata RDBMS is now approximately 60 times the size of the original database. Therefore in order to reduce space and maintain speed, a primary index was created on the num field and no unique index was employed. When a unique primary index is used, the data is more evenly distributed across the access module processors of the Teradata system. By removing the unique primary index, the possibility existed that an uneven distribution of data would occur resulting in load imbalance. However, due to the nature of genomic data, the skew among the processors was very low. Thus by using one primary index on num, the size of the database remains small while enabling rapid query processing. In addition, the query performance was improved by using a single nonunique primary index when compared to a unique primary index and a secondary index. These results are illustrated in figure 6.

Figure 6. Query Times in Seconds Based on Database Size Using a Nonunique Primary Index

Genome-to-genome comparisons can be done very quickly using the parallel nature of Teradata's RDBMS. Table 2 illustrates these times for the three databases. The most significant improvement over current sequential search tools is seen in large, genome sized queries.

In order to compare iBlast with a sequential search tool, standalone BLAST was used. The standalone version of BLAST was executed at the Ohio Supercomputer on the Sunfire 6800 server. This machine contains twenty-four 900 MHZ UltraSPARC III microprocessors chips with a memory size of 48 GB [6]. As query sizes increase, the time required to execute these queries grows linearly when standalone BLAST is used as the search tool. This is expected, and can be seen in Figure 8. While BLAST performs linearly as query size increases, iBlast is virtually constant for a variety of query sizes. Figure 9 shows the difference in query times when the entire ecoli genome was compared to the entire yeast genome. This plot portrays the improvement in query time of iBlast over BLAST. iBlast performs 68 times faster than standalone BLAST for the entire genome comparisons evaluated here.

Database	Query	Time
Ecoli	Yeast	8
Ecoli	Drosoph	61
Yeast	Drosoph	729

Table 2. Query Times in Seconds for Genome-to-Genome Comparisons

Figure 8. Comparison of Query Times Between BLAST and iBlast

Figure 9. Ecoli-to-Yeast Query Execution Times in iBlast and BLAST

A web based client was created to interface with the database from any location. Currently nucleotide databases with sequence-based indices have been loaded. Several options will be available on this web based client including:

- Genome-to-genome comparisons
- Amino acid databases
- Text file queries
- Query submission as input to the web page
- Variations in word size (16 or 12)
- Biologic indexes
- Sequence based indexes (iBlast)

4. Discussion and Future Work

Numerous areas of the iBlast indexing scheme can be optimized. Optimization algorithms can aid in determining the best method of indexing the genomic database. A Design Of Experiments (DOE) test will be constructed to test specific variables in the analysis. Three measures: accuracy, repeatability, and speed, will be used to determine the optimal solution. These measures will be tested for varying interval lengths such as 10000, 1000, 100, 25, 10, etc. Other variables include: number of intervals, input sequence size, and type of data. This indexing method will be compared to current tools such as BLAST and FASTA [7] as well as other indexed schemes.

The expected outcome is that of maintaining the accuracy of tools such as BLAST or FASTA while doubling the speed of the search. This improvement in search time will be beneficial as long as accuracy is maintained. Current indexing schemes are less accurate yet much faster than BLAST or FASTA. This scheme can be as precise as current search tools, with a significant decrease in processing time.

Biological indices will also be explored as a more efficient indexing method. Since large regions of genomes (called isochors) have been observed to exhibit homogeneous G/C content (the percent of all nucleotides that are either G or C), an index based on G/C content may allow rapid elimination of highly dissimilar sequences [2]. The genome would be indexed using a fixed interval size based on the G/C percentage content. Due to substitutions, insertions, and deletions, the query sequence often may not match sequences in the database exactly. Therefore, a threshold value can be used. When searching the database for a sequence of a specified G/C content, sequences with somewhat higher and lower G/C content can be returned in the match as well. For the human genome, consisting of 3400 Mbp (mega base pairs), utilizing an interval of 100 base pairs yields 34 million values for the index. This indexing scheme will quickly eliminate sequences in the database with a G/C percentage that differs greatly from that of the query sequence. In addition, a Smith Waterman alignment will be performed as a second step to determine an answer to a query.

The above indexing scheme does not take into account the relationships between genes. To further improve the index and querying capabilities, related genes can be grouped by utilizing secondary relations. Genes may be grouped by function and structure. Thus, a user may query a genomic database given a sequence or a gene identifier. All genes returned will have similar function and structure across genomes.

Genomic data is updated often. A method for streaming the updates into a local RDBMS is necessary to ensure accurate and consistent data. This can be performed using data from the NCBI web site.

5. Conclusion

Genomic databases are increasing in size very rapidly. Molecular biologists need to be able to extract and use accurate information quickly. They often perform sequence alignments to determine an organism's evolutionary history, or to determine functional data. Several exhaustive and indexed tools exist to facilitate timely retrieval. Even so, the need to index and speed up queries grows daily. The indexing scheme described in this paper can maintain the accuracy of existing search tools, while significantly increasing query speed. As each new genome sequence project is initiated it is becoming more evident that the best option for searching genomic data lies in indexed systems.

6. Acknowledgements

The authors would like to thank Sheryl Rajbhandari, Doug Raiford, and Akash Jain for their contributions and feedback. This work was supported in part by the Ohio Biomedical Research and Technology Transfer (BRTT) program, through award BRTT02-0002, and by the Dayton Area Graduate Studies Institute (DAGSI) through their graduate fellows program.

7. References

- [1] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, "Basic Local Alignment Search Tool," *J. Molecular Biology*, vol. 215, pp. 403-410, 1990.
- [2] G. Bernardi. "The isochore organization of the human genome and its evolutionary history – a review," *Gene*, vol. 135, pp. 57-66, 1993.
- [3] S. Chung, "NCR WorldMark 4800". <http://www.cs.wright.edu/~schung>
- [4] H.E. Williams and J. Zobel, "Indexing and Retrieval for Genomic Databases" *IEEE Transactions on Knowledge and Data Engineering*, vol.14, no. 1, pp.63-78, 2002.
- [5] M. Larkins and T. Coffing, *Teradata SQL Unleash the Power*, Middletown, OH: Coffing Publishing, 2001.
- [6] Ohio Super Computer Center, "OSC SUN Sunfire 6800". <http://oscinfo.osc.edu/hardware/#coe>
- [7] W.R. Pearson and D.J. Lipman, "Improved Tools for Biological Sequence Comparison," *Proc. US Nat'l Academy of Sciences*, vol. 85, pp. 2444-2448, 1988.
- [8] T.F. Smith and M.S. Waterman. "Identification of common molecular subsequences." *J. Mol. Biol.* 147:195-197, 1981.