ICIC Express Letters
Volume 12, Number 7, July 2018
ICIC International ©2018 ISSN 1881-803X
pp. 723–729

METHODS AND EVALUATIONS OF DECISION TREE ALGORITHMS ON GPUS: AN OVERVIEW

Nesreen Adnan Hamad, Fatima Mousa Quiam
and Khalid Mohammad Jaber

Faculty of Science and Information Technology
Al-Zaytoonah University of Jordan
P.O. Box 130, Amman 11733, Jordan
{ nesreen.hamad; f.quiam; k.jaber }@zuj.edu.jo

Received November 2017; accepted February 2018

Abstract. *Biological data is currently being exponentially increasing and causing an essential issue when meaningful information is required to be extracted from massive DNA-protein databases. Various methods related to bioinformatics computations have been used to extract, search, integrate, and analyze biological data in efficient ways. In particular, the decision tree approach is one of the common approaches applied in processing biological data. However, when dealing with massive datasets, the time needed to build the decision tree will increase. Therefore, parallel computing is used to accelerate the construction of the decision tree. This paper provides an overview and background of the state of the art of Graphics Processing Unit (GPU) parallel computing approaches and other parallel computing approaches that are being used to build a decision tree.*
Keywords: Decision tree, Graphics Processing Unit (GPU), Bioinformatics, Parallel computing, DNA-protein sequence databases, Data mining

1. **Introduction.** Biological experiments produce massive amounts of data from time to time. As a result, the need to process this data by using computers has risen due to their capabilities in dealing with such data compared to traditional methods. Therefore, the field of bioinformatics, which is an interdisciplinary field that combines biology and information technology, was established. One of the main objectives of this field is to develop new software tools that can manage, integrate, and interpret information that is derived from the biological data comprising sequence, microarray, genomic, proteomic, metabolic, structural, the entire organism, image, phylogenetic, and cellular levels [1].

The biological data that are generated from massive biological experiments consists of many types. The most basic type of the data is the primary sequence data. The primary sequence data consists of three types, which are: the protein primary sequences, the Ribonucleic Acid (RNA), and the Deoxyribonucleic Acids (DNA). Public biological sequence databases are classified into three main groups, which comprise the databases of DNA, proteins, and the specialized database, which are the outputs of processing and organizing the DNA and the protein sequence data [2].

There are three major public databases for the DNA repository which share the same format and are however maintained and located differently. The most common DNA databases comprise the DNA Data Bank of Japan (DDBJ), the European Molecular Biology Laboratory (EMBL), and the human genome initiative (GenBank) [3]. All these databases share a common characteristic, which is being maintained and kept up-to-date on a regular basis. There are other databases such as the Protein Information Resource (PIR), the Protein Data Bank (PDB), and the Swiss-Prot.

DOI: 10.24507/icicel.12.07.723

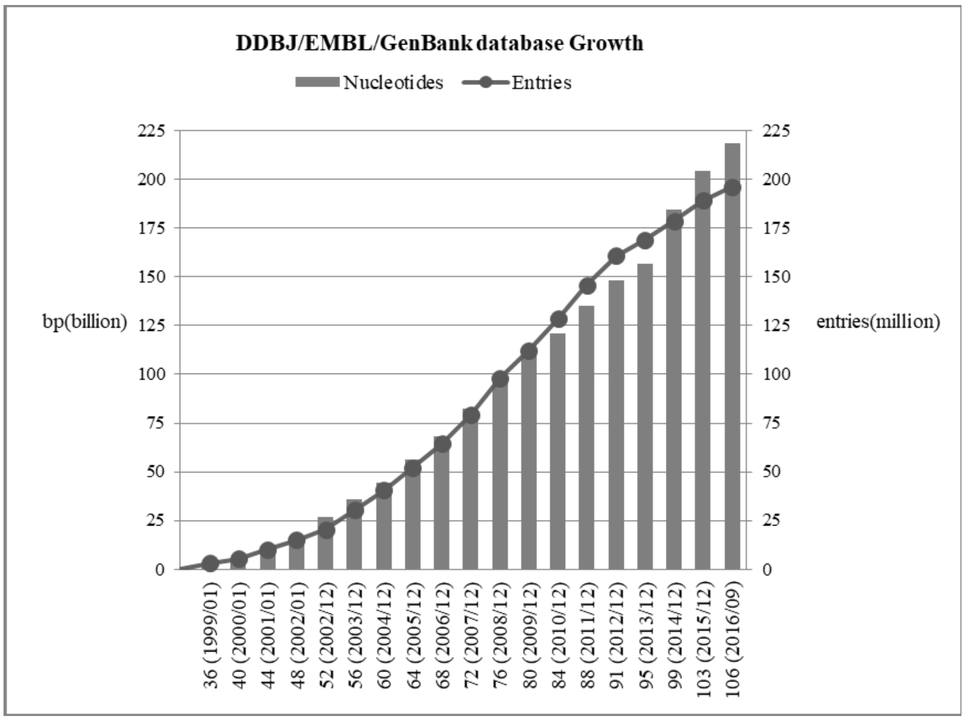

Figure 1. The exponential growth of the DDBJ, EMBL and GenBank from 1999 to 2016 (Adapted from [5])

The amount of the biological data is increasing exponentially. An example of this growth can be shown in Figure 1, which illustrates the growth of DDBJ, EMBL and GenBank from 1999 to 2016. Therefore, the need for improving the algorithms that can sort, search, and analyze this data became a must where this can be addressed via data mining. Data mining can be defined as a method for uncovering useful information that is concealed in massive databases. Over the past few years, data mining has been attracting researchers to apply it in many different disciplines such as in marketing databases, medicine, bioinformatics, and engineering [4].

One of the common techniques in data mining is the decision tree technique. A decision tree builds a model as a tree-structure in order to highlight the effect of a decision. It receives a dataset as an input, and breaks it down into smaller subsets, while an associated decision tree is incrementally developed at the same time. It is used to simplify complex problems, and to assess the cost-effectiveness of a research. Additionally, a decision tree is used in diverse areas as an indexing approach such as in information retrieval [6], video indexing [7], graph database [8,9], and index large DNA-protein sequence datasets [2]. The decision tree also has the ability to deal with large datasets where several previous efforts attempted at using it in large data such as Supervised Learning In Quest (SLIQ) [10], Scalable Parallelisable INduction of decision Tree (SPRINT) [11], and Classification for Large or OUt-of-core Datasets (CLOUDS) [12].

When the decision tree deals with a huge dataset, the time that is needed to build the tree is large. Therefore, parallel processing is used in order to assist in speeding up the constructive process of the decision tree. Parallel processing is a computational type in which many processors work simultaneously to process huge amounts of data. It is applicable in many disciplines such as data visualization, computational biology, engineering and weather forecasting [13].

Parallel processing can be classified into implicit and explicit parallelism. If the type of the task part can be decided to run in parallel by the programming language, then it is called implicit parallelism. If the programmer can, however, allow specific parts to run in parallel, then it is called explicit parallelism. When dealing with a massive amount of data, then two mechanisms are likely to be applied, which comprise both of the function decomposition and the data decomposition or one of them. Both data parallelisms are combined by the hybrid parallelism with the function decomposition, and the vertical or horizontal distribution [14].

One of the trends in parallel computing is the General-Purpose programming using a Graphics Processing Unit (GPGPU), which is a technique that is used for programming the chips of the Graphics Processing Unit (GPU) based on the use of the Application Programming Interface (API) functions (e.g., the Compute Unified Device Architecture (CUDA), the Direct3D, and the Open Graphics Library (OpenGL)) [15] in order to gain the required results within a short time. The Graphics Processing Units (GPUs) are considered to be streaming multiprocessors that are highly being threaded based on the data throughput and based on extremely high computations [16]. Many applications studied in the literature have been using the CUDA in many different fields such as in biology, chemistry, data mining, astronomy, and physics [17].

Due to the significant role that parallel decision tree can play in manipulating huge datasets, moreover, in order to recognize the new approaches for parallel decision tree which as a result can guide the future of research in this field, this paper highlights the way the researchers are applying the parallel decision tree algorithm on GPU for manipulating biological datasets.

This paper is organized as follows. Section 2 presents some of the related techniques on GPU used in decision trees. On the other hand, other parallel techniques are presented in Section 3. Finally, the drawn conclusions of this paper are presented in Section 4.

2. Decision Tree and Related Techniques on GPU.

This section explores the use of the GPU for decision tree algorithms. The advantages, disadvantages, data decomposition, and function decomposition are all highlighted in this section. Additionally, Table 1 summarizes the researches that are discussed in this section.

Table 1. A summary of some researches for the parallel decision tree on GPUs

Author(s)	Year	Parallelism	Disk-Resident/ Database-Resident	Dataset	Training Data/ Testing Set
Chiu et al. [21]	2011	Data	Disk-Resident	N/A	N/A
Sharp [22]	2008	Data	Disk-Resident	Labeled object recognition database	100 training example
Grahn et al. [23]	2011	Data	Disk-Resident	EULA dataset	10-fold cross-validation
Nasridinov et al. [24]	2013	Data	Disk-Resident	N/A	N/A
Pilkington and Zen [25]	2010	Function	Disk-Resident	N/A	N/A

In parallel processing, many attempts have been made for forming a scaling process for the decision tree through massive amounts of datasets. Jaber et al. discuss the parallel decision tree used in genomic data, large datasets, and other fields [2,18-20].

Parallelization has become a widely known mechanism for speeding up the decision tree tasks that manage the massive amounts of data. Nonetheless, there exist many mechanisms which are likely to parallelize the decision trees. These comprise either data and function decompositions or one of them. Data decomposition includes the vertical and the horizontal distributions. In the horizontal distribution, the training data is divided evenly into many parts. This data is then assigned to each processor. The distinct data is only assigned to each processor and is stored in the memory that belongs to each processor. In order to explore the best split node, a communication occurs among the entire processors.

In the vertical distribution, the training data is also divided evenly into many parts where this data is then assigned to each processor. The whole data is only assigned to each processor and is stored in the memory that belongs to each processor. In this type of distribution, each processor stores in its memory only the whole data that is related to it and the classes' values. Therefore, each processor must evaluate its attributes when the construction is being processed. Both data parallelisms are merged by the hybrid parallelism with either the vertical distribution or the horizontal distribution, and the function decomposition.

Chiu et al. [21] present a parallel sorting tree in order to reduce the execution time during the calculation of the split criteria via the CUDA Data Parallel Primitives Library (CUDPP) sorting algorithm. The CUDPP library is provided by the CUDA for sorting and building data structures for the tree. However, the entire data is required by this method to be loaded into the main memory at all times prior to applying the induction over the sorting process. Therefore, the largest induced dataset is bounded by the size of the memory. Additionally, the obtained results, datasets, and performance analysis of the researchers' work are not presented to highlight the effectiveness of their methodology.

Sharp [22] implemented a decision tree for forests data in the GPU via the High-Level Shader Language (HLSL) and the Microsoft Direct3D Software Development Kit (SDK). The researcher's strategy for parallelizing the decision tree on the GPU is to perform a mapping process for the data structure that includes the decision forest and the 2D texture array. The move across the forest is done in a parallel way in each point included in the input data based on the efficient use of a non-branching pixel shader. The training data responses to a set of candidate features are calculated, where the responses are disseminated through to a suitable histogram based on the use of a vertex shader. Thus, the entire histograms would be used simultaneously with many different tree learning algorithms. Nonetheless, the largest dataset that can be processed has a higher limit in this method since it makes use of the data structure that is scaled with the size of the dataset. This data structure must exist at all times in the main memory. However, the methods of dividing the dataset for training data and testing data are not mentioned such as the bagging and boosting algorithm.

Grahn et al. [23] attempt to parallelize both building and classification based on the CUDA GPU Random Forest (CUDARF) learning algorithm. The researchers conducted comparisons between their proposed method and the state-of-the-art random forests algorithms (FastRF and LibRF) by using the Weka in terms of the accuracy, and by using the 10-fold cross-validation test. However, Nasridinov et al. [24] claim that this method is not appropriate for a massive amount of datasets if the size of the GPU memory is smaller than the size of these datasets.

Nasridinov et al. [24] propose a ubiquitous approach for parallel computing in order to build the decision tree onto the GPU. In their approach, they applied the divide-and-conquer parallelism of the well-known decision tree learning, namely, the Iterative Dichotomiser 3 (ID3) algorithm, based on two levels. The first level represents the outer level where the tree is built node-by-node. The second level represents the inner level where data records are being sorted through one node. However, the researchers did not mention the dataset by which they use in their experiments. Furthermore, they did not compare their proposed method in terms of the accuracy attribute. Additionally, the authors did not clarify how they calculated the energy consumption.

Another work presented in the Interspeech 2010 Conference authored by Pilkington and Zen [25] discusses an implementation process that is performed for the decision tree-based context clustering over the GPUs in order to construct a speaker dependent of the Hidden Markov Model (HMM) that is based on a speech synthesis system. Experimental results showed that the new implementation running on GPUs was about 9.5 times faster than the conventional one running on CPUs.

3. Different Recent Parallel Techniques for Decision Tree.

This section discusses the decision tree using different recent parallel techniques other than the GPU technique.

The issue of reducing the time complexity in the decision tree elicitation of the training stage based on the use of the ParDTLT parallel threading algorithm is addressed by Franco-Arcega et al. [26]. The parallel method was controlled and performed in the training stage. Moreover, different datasets of different sizes were applied in the training and validation stages. These comprise the Knowledge Discovery in Databases (KDD) Donation, Weka Random, GalStar, Agrawal and PAMAP datasets. Additionally, the authors compared their proposed algorithm with the following sequential algorithms: C4.5, VFDT, YaDT and DTLT. The comparison included the classification quality and the time complexity. The results showed that the execution time of the training process of the ParDTLT algorithm was smaller than the ones obtained with the sequential algorithms. On the other hand, ParDTLT was very competitive in terms of the quality of classification.

A parallel decision tree construction algorithm is proposed by Boryczka et al. [27] in order to make use of the Ant Colony Optimization (ACO). The main idea of their algorithm is to speed up the construction of the tree by splitting the ants' population into smaller subpopulations by which the measurements are being performed in a parallel manner. Five different tests were selected to be performed by the researchers. These tests were derived from the UCI repository, namely: letter-recognition, connect-4, krkopt, poker-hand, and pendigits. The implementation of the proposed algorithm was performed by using the C++ programming language with the use of the Intel MPI Library. Nevertheless, the researchers did not give their strategies of the parallel model, for example, on whether it is a task/farming model or not.

A global decision tree system is implemented based on a parallel algorithm by Czajkowski et al. [28] in order to enhance the speed of the decision tree evolutionary induction. This system integrates the Message Passing Interface (MPI) paradigms and the shared memory (OpenMP) together for performing this enhancement. Four artificially produced datasets of many different attributes and characteristics were used by the researchers. All datasets consisted of 100,000 instances. The authors compared the results of their parallel algorithm with the sequential algorithm version. The parallel algorithm outperformed the sequential one by reducing the time needed for the tree induction from 9 hours to 42 minutes. However, the authors did not compare the accuracy of the results because the main aim of their work was to speed up the tree induction time.

Qu and Prasanna [29] propose the multi-threading technique based on the use of the Field Programmable Gate Array (FPGA) and the openMP in order to allow the conversion technique to perform a translation for the generic decision tree through many different compact hash tables. They used Internet traffic classification datasets for testing the decision tree. To evaluate the performance, the proposed design was prototyped on state-of-the-art FPGA and multicore General Purpose Processors (GPPs). Experimental results showed that, for a typical 92-leaf decision-tree, 533 Million Classifications Per Second (MCPS) throughput and 26 ns latency on FPGA, and 134 MCPS throughput and 239 ns latency on multi-core GPP were achieved. Additionally, $6\times$ and $2.7\times$ speedups respectively were obtained.

Ben-Gal and Trister [30] present an algorithm to construct parallel decision trees of consistently non-increasing Expected Number of Tests (ENT) in order to highlight the issue of gaining worse solutions when increasing the construction complexity. The authors mentioned that their algorithm can run on multiple processors where each processor builds a separate decision tree. The proposed algorithm depends on the subset parameter (s) which itself depends on the number of parallel processors. This allows increasing s without facing the growth in complexity when additional processors are available. As a result, the number of decision trees combinations can be decreased to $O(2^n)$.

Vukobratović and Struharik [31] propose a co-processor in order to apply it in the induction process for the hardware aided decision tree based on the use of the evolutionary approach called the Evolutionary Full Tree Induction co-Processor (EFTIP). This approach is used for speeding up the fitness evaluation task of the hardware since it is shown to be proven that this task represents a bottleneck of the execution time. The comparison of the HW/SW EFTI algorithm implementation with the pure software implementations showed that the proposed HW/SW architecture offered substantial speedups for all the tests performed on the selected UCI datasets. The authors mentioned that their approach could use multiple memories to read data out. Additionally, they used parallel computing in order to compute the sum of products in one of the equations they used. However, the authors did not clearly mention the parallel method they used for the computation process.

4. Conclusions.

In this paper, the origin sources of the high computational demands problem of building decision tree are presented. Additionally, the research efforts made for solving this problem were underlined, particularly, when using the parallel computing approaches. A review of many published state-of-the-art approaches was carried out in order to form the direction of the future research in the area. Its suggested future direction is to propose solutions, which can be offered based on this review, to the problem by using multiple methods of parallel computing approaches in order to reduce complexity, in both time and memory storage requirements. It can be concluded from this study that, in spite of some limitations in the existing parallel-based decision tree presented in this paper, the parallel processing methods are able to provide practical solutions. The need for further research in utilizing parallel computing methods for solving this problem is still much desired. Future efforts can possibly be directed to the implementation of parallel techniques such as the hybridization between distributed and shared memory models, the hybridization between GPU and CPU methods, or the hybridization between the data decomposition and the function decomposition.

REFERENCES

[1] D. R. Westhead, J. H. Parish and R. M. Twyman, *Instant Notes in Bioinformatics*, BIOS Scientific Publishers, Oxford, UK, 2002.

[2] K. M. Jaber, R. Abdullah and N. A. Rashid, Adapting decision tree-based method to index large DNA-protein sequence datasets, *The Research Bulletin of Jordan ACM*, vol.2, no.3, pp.57-73, 2011.

[3] T. Kulikova, R. Akhtar, P. Aldebert et al., EMBL nucleotide sequence database in 2006, *Nucleic Acids Research*, vol.35, 2007.

[4] S. Chakrabarti, M. Ester, U. Fayyad, J. Gehrke, J. Han, S. Morishita, G. Piatetsky-Shapiro and W. Wang, *Data Mining Curriculum: A Proposal (Version 1.0)*, http://www.kdd.org/curriculum/index.html, 2006.

[5] DNA Data Bank of Japan (DDBJ), *DDBJ Database Release History (rel.1 rel.106)*, http://www.ddbj.nig.ac.jp/breakdown_stats/dbgrowth-old-e.html, 2017.

[6] G. Quellec, M. Lamard, L. Bekri, G. Cazuguel, B. Cochener and C. Roux, Multimedia medical case retrieval using decision trees, *The 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Lyon, France, pp.4536-4539, 2007.

[7] K. Shearer, H. Bunke and S. Venkatesh, Video indexing and similarity retrieval by largest common subgraph detection using decision trees, *Pattern Recognition*, vol.34, no.5, pp.1075-1091, 2001.

[8] C. Irniger and H. Bunke, Graph database filtering using decision trees, *Proc. of the 17th International Conference on Pattern Recognition*, Cambridge, UK, pp.383-388, 2004.

[9] C. Irniger and H. Bunke, Decision trees for filtering large databases of graphs, *International Journal of Intelligent Systems Technologies and Applications*, vol.3, nos.3-4, pp.166-187, 2008.

[10] M. Mehta, R. Agrawal and J. Rissanen, SLIQ: A fast scalable classifier for data mining, *Proc. of the 5th International Conference on Extending Database Technology: Advances in Database Technology, Lecture Notes in Computer Science*, vol.1057, 1996.

[11] J. C. Shafer, R. Agrawal and M. Mehta, Sprint: A scalable parallel classifier for data mining, *Proc. of the 22nd International Conference on Very Large Data Bases (VLDB'96)*, Morgan Kaufmann, San Francisco, CA, USA, pp.544-555, 1996.

[12] K. Alsabti, S. Ranka and V. Singh, CLOUDS: A decision tree classifier for large datasets, *Proc. of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, CA, USA, pp.2-8, 1998.

[13] K. Jaber, *Fast Decision Tree-Based Method to Index Large DNA-Protein Sequence Databases Using Hybrid Distributed-Shared Memory Programming Model*, Ph.D. Thesis, Universiti Sains Malaysia, Malaysia, 2010.

[14] A. Grama, A. Gupta, G. Karypis and V. Kumar, *Introduction to Parallel Computing*, Person Education, UK, 2003.

[15] D. B. Kirk and W. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, Morgan Kaufmann, USA, 2010.

[16] D. Patterson and J. Hennessy, *Computer Organization and Design – The Hardware/Software Interface*, 5th Edition, Morgan Kaufmann, USA, 2013.

[17] NVIDIA Research, *GPU-Accelerated Applications*, http://www.nvidia.com/object/gpu-applications.html, 2017.

[18] K. Jaber, R. Abdullah and N. A. Rashid, A framework for decision tree-based method to index data from large protein sequence databases, *IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, Kuala Lumpur, Malaysia, pp.120-125, 2010.

[19] K. M. Jaber, R. Abdullah and N. A. Rashid, Adapting decision tree-based method to index large DNA-protein sequence datasets, *The Research Bulletin of Jordan ACM*, vol.2, no.3, pp.57-73, 2011.

[20] K. M. Jaber, R. Abdullah and N. A. Rashid, HDT-HS: A hybrid decision tree/harmony search algorithm for biological datasets, *International Conference on Computer & Information Science (ICCIS)*, Kuala Lumpur, Malaysia, pp.341-345, 2012.

[21] C.-C. Chiu, G.-H. Luo and S.-M. Yuan, A decision tree using CUDA GPUs, *Proc. of the 13th International Conference on Information Integration and Web-Based Applications and Services*, New York, NY, USA, pp.399-402, 2011.

[22] T. Sharp, Implementing decision trees and forests on a GPU, *Proc. of the 10th European Conference on Computer Vision, Lecture Notes in Computer Science*, vol.5305, pp.595-608, 2008.

[23] H. Grahn, N. Lavesson, M. H. Lapajne and D. Slat, CudaRF: A CUDA-based implementation of random forests, *The 9th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, Sharm El-Sheikh, Egypt, pp.95-101, 2011.

[24] A. Nasridinov, Y. Lee and Y.-H. Park, Decision tree construction on GPU: Ubiquitous parallel computing approach, *Computing*, vol.96, no.5, pp.1-11, 2013.

[25] N. Pilkington and H. Zen, An implementation of decision tree-based context clustering on graphics processing units, *Proc. of the 11th Annual Conference of the International Speech Communication Association*, Makuhari, Chiba, Japan, pp.833-836, 2010.

[26] A. Franco-Arcega, J. Suarez-Cansino and L. G. Flores-Flores, A parallel algorithm to induce decision trees for large datasets, *XXIV International Conference on Information, Communication and Automation Technologies (ICAT)*, Sarajevo, Bosnia and Herzegovina, pp.1-6, 2013.

[27] U. Boryczka, J. Kozak and R. Skinderowicz, Heterarchy in constructing decision trees – Parallel ACDT, *Transactions on Computational Collective Intelligence X, Lecture Notes in Computer Science*, vol.7776, pp.177-192, 2013.

[28] M. Czajkowski, K. Jurczuk and M. Kretowski, A parallel approach for evolutionary induced decision trees. MPI + OpenMP implementation, *International Conference on Artificial Intelligence and Soft Computing, Lecture Notes in Computer Science*, vol.9119, pp.340-349, 2015.

[29] Y. R. Qu and V. K. Prasanna, Compact hash tables for decision-trees, *Parallel Computing*, vol.54, pp.121-127, 2016.

[30] I. Ben-Gal and C. Trister, Parallel construction of decision trees with consistently non-increasing expected number of tests, *Applied Stochastic Models in Business and Industry*, vol.31, no.1, pp.64-78, 2015.

[31] B. Z. Vukobratović and R. J. R. Struharik, Co-processor for evolutionary full decision tree induction, *Microprocessors and Microsystems*, vol.45, pp.253-269, 2016.