Junfeng Wu

DEVELOPMENT OF AN EFFECTIVE DATA STRUCTURE, SUPPORTING INCREMENTAL CHANGES IN PARALLEL TREE PROCESSING

ABSTRACT

Dynamic tree problem is an important problem in graph theory, it has been studied and applied widely. Dynamic tree data structures are used to support many applications, like edge insertions and deletions for a forest. Dynamic tree algorithms solve the problems with incremental changes in the data while parallel tree contraction provides a technique for the parallel solution of it. Many dynamic algorithms are based on parallel tree contraction, and the data structure used for supporting dynamic trees are ST-Trees, Topology Trees and RC-Trees. RC-Trees works faster than previous data structure when answering dynamic queries, and it takes O (log n) operation times.

Dynamic tree is still not solved perfectly and still under discussing. With technical development, to solve dynamic queries are more important than before. RC-Trees supports the dynamic queries in dynamic tree, but from the nowadays' point of view, it is not very efficient.

In this thesis, we develop data structure for solving the dynamic problems based on the RC-Trees data structure. Making the algorithm more efficient in parallel processing. Then complete the implementation of the developed data structure in parallel processing. We solve the research problem in this thesis that What algorithm can be developed for solving solve dynamic tree contraction problems in parallel based on RC-Trees data structure and How to implement this algorithm.

Keywords: data structure, dynamic problem, parallel tree contraction

FIGURES

TABLE OF CONTENTS

1 Introduction

Dynamic tree problem is an important problem in graph theory, it has been studied and applied widely. By applying dynamic tree, we can optimize the common algorithm, like solving the maximum-flow problem. And with dynamic tree, we can get very good time complexity. Dynamic tree data structures are used to support many applications, it was first considered by Sleator D D et al. (1983) and presented supporting edge insertions and deletions for a forest. A dynamic tree structure solving the dynamic tree problem by answering queries in a forest of trees when the trees are changing.

Dynamic tree algorithms solve the problems with incremental changes in the data while parallel tree contraction provides a technique for the parallel solution of it. Miller G. L. et al. (1985) and Miller G. L. et al. (1989) first introduced parallel tree contraction and further discussed in Miller G. L. et al. (1991) and Reif J. et al. (1994). Parallel tree contraction is a wide used method for parallel tree processing.

Many dynamic algorithms are based on parallel tree contraction, and the data structure used for supporting dynamic trees are ST-Trees, Topology Trees and RC-Trees, Acar U. et al. (2003). RC-Trees, also known as Rake & Compress Tree, was invited by Acar U. et al. (2003) to solve dynamic tree problems, they dynamized the parallel tree contraction algorithm of Miller G. L. et al. (1985) in their research Acar U. et al. (2004). In this thesis, we develop data structure for solving the dynamic problems based on the RC-Trees data structure. RC-Trees works faster than previous data structure when answering dynamic queries, and it takes O (log n) operation times.

Dynamic tree is still not solved perfectly and still under discussing. With technical development, to solve dynamic queries are more important than before. RC-Trees supports the dynamic queries in dynamic tree, but from the nowadays' point of view, it is not very efficient. Multicore computing is widely used in algorithm field, make the algorithm in more efficient parallel process can reduce the computing time. But there is not a completed implementation of dynamic tree in parallel processing, so in this thesis, a data structure will be developed for more efficient dynamic operation for time saving.

The applications of parallel tree contraction are described in the thesis. One problem that parallel tree contraction can solve is change propagation. Acar U et al. (2005). Change propagation automatically adjusting the output of an algorithm to changes in the input. Another problem that tree contraction solves is expression evaluation, performing tree contraction in parallel would make the computation more efficiently.

To implement the algorithm into application, we use a parallel library called PASL for parallel processing. PASL stands for Parallel Algorithm Scheduling Library, developed by Umut A. Acar et al. (2015). The library is based on C++ languange, so our implemetation will be done in C++ program. PASL generalizes existing parallelism constructs such as fork-join, providing the function for dynamic computing. PASL allows to solve the dynamic computational problems in parallel, effectively reducing the computing time and allows the performant parallel programs at a relatively high level. In addition, it is easy to evaluate the time cost of a query with PASL.

The following research questions are considered in the thesis:

1. What algorithm can be developed for solving solve dynamic tree contraction problems in parallel?
    a) How RC-Trees data structure can be improved?
    b) How to implement this algorithm?
2. What applications can be exampled with this algorithm?
3. How to evaluate the result of the research?

The rest of the thesis is organized as follows. Section 2 outlines the research methods used in this thesis, which helps make the research structure and deep enough. Next, Section 3 reviews the previous related work, which support the research in this thesis. Then Section 4 describes the concept of RC-Trees in details. The contribution of the thesis in Section 5 and 6. With the studies of previous research and the RC-Trees data structure, in Section 5, we make the improvement of the data structure to make better operation on dynamic queries. In Section 6, we implement the data structure into codes and evaluate the result. Finally, the results of the thesis, conclusion and discussions in this research are considered in Section 7.

2 Research methods

To research the problems in this thesis, two research methods are used, Literature review and Design science. The first research method, the literature review, follows the introduction of Creswell J. (2007). We use literature review to review the previous related work and find out the answers for the research problem. Literature review makes a survey of related references about dynamic tree and parallel tree contraction etc.. In the literature review, we list some basic theories which introduce the algorithm that will be used in the thesis, including dynamic tree problem, parallel tree contraction, RC-Trees data structure. And compare other data structure briefly, like Topology Trees, to evaluate the advantage of the RC-Trees. Then evaluate the RC-Trees which part of it should be improved.

The second research method is design science. The research follows a research framework introduced by von Alan R. H. et al. (2004) and Peffers, K. et al. (2007). The thesis focuses on the motivation of the improvement and implementation of the dynamic tree data structure. Then the research goal is set, along with the literature review, helps to improve the data structure and algorithm that will be used into the parallel process. Finally make the evaluation with the result.

The design science framework is used in the research, doing the development and evaluation in the research to satisfy the business needs and applicable knowledge. The application in the appropriate environment and additions to the knowledge base are made in the framework. In this thesis, we focus more on foundations, like theories, methods, and methodologies, like data analysis techniques and measures.

In this thesis, research problem is not much business relevant, but design science is usable in the thesis. First, we need to create an artifact of the whole research. Then the next part, in the development part, we analyze the RC-Trees data structure and improve it with new data structure to work more efficient and supports the modern parallel processing. This part solves the improvement and implementation of the developed data structure. Then the evaluation part, because we use PASL for parallel processing, and it supports the time

evaluation, thus makes the time efficiency evaluation more easily. Some applications will be used to evaluate the implemetation of the data structure.
The development and the evaluation effect with each other, we make the development of data structure and then evaluate it, the result of evaluation modifies and improves the development. Finally, the conclusion and discussion will show the research contributions.

3 Literature review

For developing an effective data structure, which supports incremental changes in parallel tree processing. We need to describe two definitions which used in the research. Dynamic tree problem is an important problem in graph theory, it has been studied and applied widely. Dynamic tree data structures are used to support many applications, supporting edge insertions and deletions for a forest. Dynamic tree algorithms solve the problems with incremental changes in the data while parallel tree contraction provides a technique for the parallel solution of it. Parallel tree contraction is a wide used method for parallel tree processing. In this literature review, we describe the parallel tree contraction and dynamic tree.

3.1 Parallel Tree Contraction

Trees are fundamental to many computational problems, such as data sorting, nearest neighbour search. Trees are important in parallel problems, Miller G. L. et al. (1985) and Miller G. L. et al. (1989) first introduced parallel tree contraction and further discussed in Miller G. L. et al. (1991) and Reif J. et al. (1994). Parallel tree contraction is a framework that supporting efficient parallel algorithms on trees. Morihata A. et al. (2008). An efficient parallel tree contraction algorithm provides the possibility to gather values in the tree efficiently and without conflict, as a result, leading to several efficient parallel algorithms. Morihata A. et al. (2011).

Parallel tree contraction is a bottom-up approach when handling trees. It allows to implement facilitating new algorithms easier with fewer processors and less time and makes complicated problem simpler with using parallel algorithms. Miller G. L. et al. (1985). Using parallel tree contraction, only takes O (log N) parallel steps for an underlying tree structure regardless of its shape, it speedup the operations. Morihata A. et al. (2014).

In Parallel tree contraction, there are two abstract operations on trees, Miller G. L. et al. (1985) called them Rake and Compress. The task for tree contrac-

tion is to reduce it to a single node. First, define a rooted tree T with n nodes and root r. To perform the contraction, rake operations and compress operations are done in any order and in parallel. The rake operation removes all the leaves. the compress operation contracts all maximal chains of tree in one step. In other word, for instance, in binary tree, the compress operation removes the node which only have one child and its parent has it as the only child. The extra operation, contract, is the simultaneous application of rake and compress to the entire tree. According to Miller G. L. et al. (1985), it only takes O (log n) time to reduce a rooted tree to a single node using O (n / log n) processors.

Morihata A. et al. (2008) introduced a contraction algorithm called Shunt based on Miller G. L. et al. (1985) parallel tree contraction. In this algorithm, it consists basic rake and compress operation, and a new operation – Shunt. A Shunt operation removes the leaf and its parent and connects the sibling of the leaf to its grandparent. Shunt leads to a simple and efficient parallel tree contraction on binary trees.

3.2 Dynamic Tree

When they are balanced, it is easy to compute various properties of trees in parallel. When they are not balanced, however, such computations can be difficult. The parallel tree contraction is well developed and the shunt algorithm is efficient, but for solving the dynamic tree problem, we need to construct a dynamic tree data structure. Dynamic tree data structures was first considered by Sleator D D et al. (1983). Link and cut are the two operations, link combines two trees into one by adding an edge, cut divides one tree into two by deleting an edge. Using the link and cut tree can answer various path queries.

There are many data structures for dynamic tree problem. Reif J.et al. (1994) described a data structure for dynamic parallel tree contraction, the tree PT, it adds or deletes leaves of the original tree with incrementally processing parallel requests. If there are U numbers of nodes in the original tree, the dynamic parallel tree contraction takes O (log (|U| log n)) time and using O (|U|log n / log(|U|log n)) processors.

From the basic dynamic tree data structure, there are many new and different data structures are invented. ST-trees, ET-trees, and in addition, along with the tree contraction technique, topology trees, top trees and RC-Trees are developed. Here we take ST-trees and RC-Trees as the research example. ST-Trees are the classic dynamic tree data structure created by Sleator D D et al. (1983), supporting path operations. It is also known as the link and cut trees. It was the first to make the operation of dynamic tree into logarithmic time. Topology trees data sturcture is the first contraction-based data structure. But in its data structure, all the vertices in the forest must have constant bounded degree. Tarjan R E. et al. (2007). Acar U. et al. (2003) invented RC-Trees, which requires the bounded degree of underlying tre too, but no require to be a constant. RC-tree is formed from an original tree, it is a support tree. RC-Trees have the operations inherit from the link-cut trees, which are rake, compress

and contraction. The Rake node was the sub tree with a certain node as the root in the original tree, Compress node was the sub tree with a certain path as the root in the original tree. Rake and Compress operation can be done in parallel in a certain procedure. RC-Trees use generic interface to separate value updates. The extra implementations are needed to realize the RC-Trees which can make queries faster. In the research Acar U. et al. (2005). RC-Trees data structure is improved with change propagation, which automatically adjusts the output to changes in the input.

Comparing with the dynamic tree data structures and considering the parallel tree contraction, we use RC-Trees to support the incremental changes in parallel tree processing. The RC-trees need implemetation and the original RC-Trees data structure was introduced 10 years ago. To catch up the development of technologies, the data structure needs to be improved.

4 RC-Trees

An RC-Tree is a support tree obtained after the rake and compress operations of parallel tree comtraction by clustering a underlying tree. A support tree is a balanced tree mapped from a n aribitrary tree to be used to answering queries efficiently. Acar U. et al. (2003).

RC-Trees have the rake and compress operations, Rake operations delete all leaves in the tree and Compress operations delete an independent set of vertices on chains. Acar U. et al. (2005). The figure 1 shows the two operations as follows:

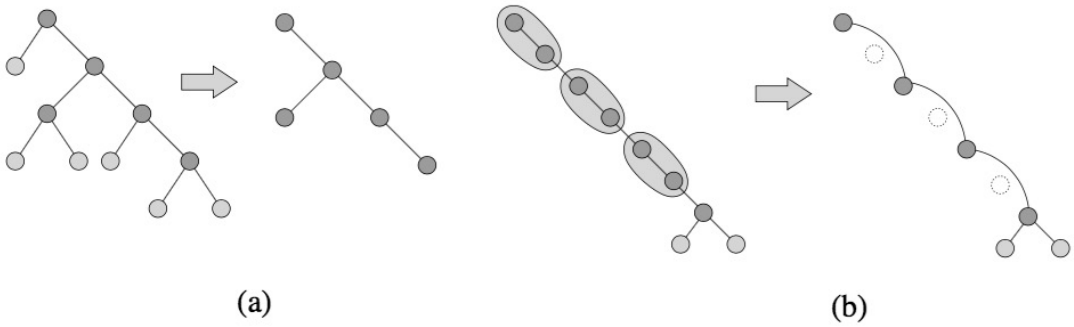

FIGURE 1 Rake(a) and Compress(b) operations.

Acar U. et al. (2003) has also indroduced an operation called finalize. Finalize operation, just as its name implies, it works when the tree is contracted to a single vertex, it stores this vertex. The main idea of RC-Trees: First, we need a unbalanced tree, called primitive tree. Then map it to a balanced tree, RC-Tree. Using the change propagation, introduced in Acar U. et al. (2005), to update RC-Tree dynamically.

The following figures shows the step of a tree contraction. We set a edge-weighted primitive tree, using Rake and Compress operation obtains a clustering. Finally, using finalize operation to store the last vertex, here the completed clustering. Then we get the RC-Tree, shows in figure 4.

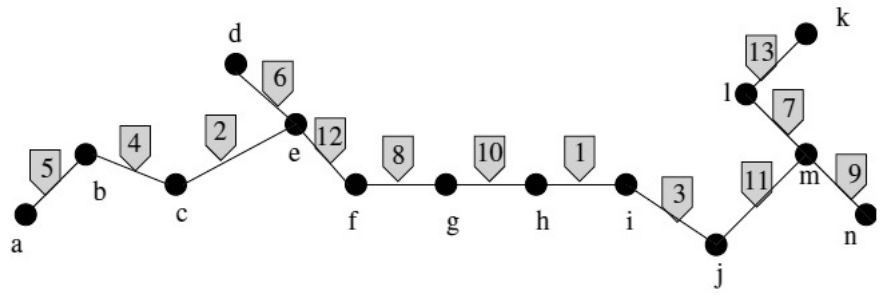

FIGURE 2 A primitive tree.

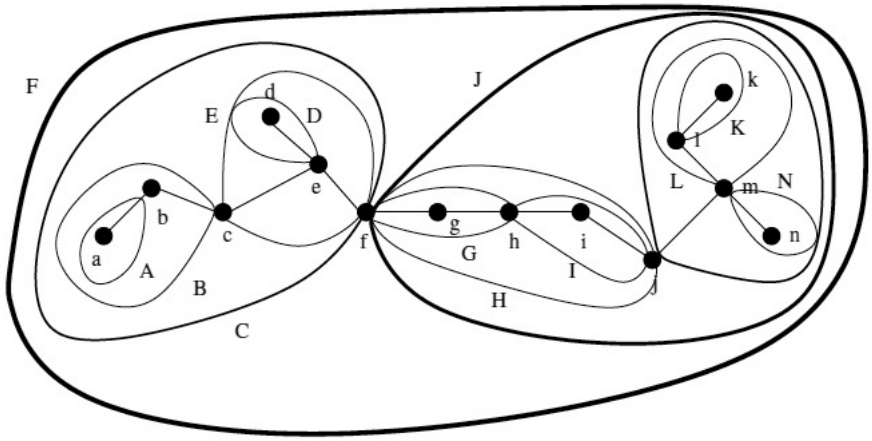

FIGURE 3 A completed clustering.

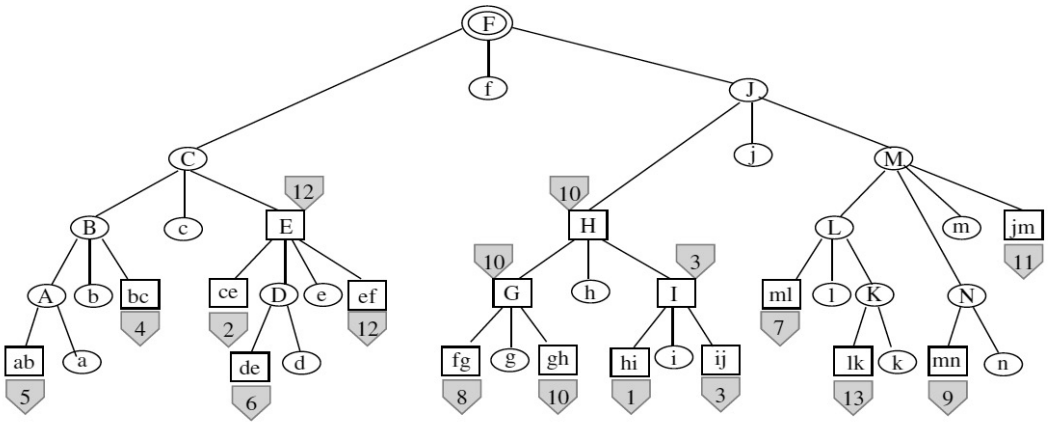

FIGURE 4 A RC-Tree.

One example query that RC-Trees can answer is a path query. A path query is to find the maximum weight path between two selected vertices. The

path query can be answered by walking up the path simultaneously with two vertices. Set up two values of each path, left and right, left is the length between the left boundary of the cluster and the vertex, so the same as the right one. When the two vertices meet, the maximum of the value will be the answer.

5 Improvement

In this thesis research, we develop data structure based on the previous data structure RC-Trees. The feature of the RC-Trees itself requires the extra implementation of the RC-Trees, thus the first improvement is made the implementation of RC-Trees completed.

Second, using the modern technology of programming to make the code usable for the modern applications.

Third, the program is coded to support the parallel process in each rake and compress operations and between rake and compress operations. In other words, in every operation round, each operation of rake should be done in parallel, so as the operations of compress. Then the rake and compress operations themselves should be done in parallel.

6 Implementation

With the improvement of the data structure. The implemeentation will include the implementation of a new algorithm and applied into C++ program.

First, the algorithm is coded by C++ language, after debugging the program logically. Some application of queries are made to test the algorithm whether it is right implemented or not.

Second, using the interface from the PASL, combine the codes and make the program working in parallel processing.

Third, evaluate the time efficiency in parallel processing comparing with the original algorithm without parallel processing.

7 Conclusion and Discussion

From the result of literature review, the parallel tree contraction data structure is not enough for solving our research problem. To answering the dynamic queries, we need to build a dynamic tree data structure. Have compared the dynamic tree data structures, we choose RC-Tress to be the basic data structure used to support the incremental changes in parallel proseccing.

The dynamic parallel tree contraction data structure is developed and implemented into C++ programs.

REFERENCES

Sleator D D, Tarjan R E. (1983). A data structure for dynamic trees. Journal of Computer and System Sciences, 26(3), 362-391.

Miller G. L., & Reif J. H. (1985). Parallel tree contraction and its application (pp. 478-489). Defense Technical Information Center.

Miller G. L., & Reif J. H. (1989). Parallel Tree Contraction--Part I: Fundamentals.

Miller G. L., & Reif J. H. (1991). Parallel tree contraction part 2: further applications. SIAM Journal on Computing, 20(6), 1128-1147.

Reif J., Tate S. (1994). Dynamic Parallel Tree Contraction. Proceedings of SPAA. – 1994. – P. 114-121.

Acar U., Blelloch G., Vittes J. (2003). Separating Structure from Data in Dynamic Trees. – 2003.

Acar U., Blelloch G, Harper R., Vittes J., Woo S. (2004). Dynamizing Static Algorithms with Applications to Dynamic Trees and History Independence. Proceedings of SODA. – 2004. – P. 531-540.

Acar U., Blelloch G., Vittes J. (2005). An Experimental Analysis of Change Propagation in Dynamic Trees. Workshop on Algorithms Engineering and Experiments. – 2005.

Umut A. Acar, Arthur Charguéraud, and Mike Rainey. (2015). Parallel Computing in C++ with PASL. v1.1 2015-04 .

Creswell J. (2007) Review of the Literature, Chapter 2 of Research Design: Qualitative, Quantitative, and Mixed Method Approaches. Thousand Oaks: Sage Publications.

von Alan R. H., March S. T., Park J., & Ram S. (2004). Design science in information systems research. MIS quarterly, 28(1), 75-105.

Peffers K., Tuunanen T., Rothenberger M. A., & Chatterjee S. (2007). A design science research methodology for information systems research.Journal of management information systems, 24(3), 45-77.

Morihata A., & Matsuzaki K. (2008). A parallel tree contraction algorithm on non-binary trees (Vol. 336). Technical Report METR 2008-27, Department of Mathematical Informatics, University of Tokyo.

Morihata A., & Matsuzaki K. (2011). A practical tree contraction algorithm for parallel skeletons on trees of unbounded degree. Procedia Computer Science, 4, 7-16.

Morihata A., & Matsuzaki K. (2014). Parallel Tree Contraction with Fewer Types of Primitive Contraction Operations and Its Application to Trees of Unbounded Degree. IPSJ Online Transactions, 7(0), 148-156.

Reid-Miller, M., Miller, G. L., & Modugno, F. (1993). List ranking and parallel tree contraction. Synthesis of Parallel Algorithms, 115-194.

Shun J., Gu Y., Blelloch G. E., Fineman J. T., & Gibbons P. B. (2015, January). Sequential random permutation, list contraction and tree contraction are highly parallel. In Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (pp. 431-448). SIAM.

Tarjan R E, Werneck R F. Dynamic trees in practice. (2007). Experimental Algorithms. Springer Berlin Heidelberg, 80-93.

Umut A. Acar . (2015). Algorithm Design: Parallel and Sequential. April 2015

Sumer O., Acar U., Ihler A., Mettu R. (2011). Fast Parallel and Adaptive Updates for Dual-Decomposition Solvers. Proceedings of 25th AAAI Conference.

Acar U., Charguéraud A., Rainey M. (2014). Data Structures and Algorithms for Robust and Fast Parallel Graph Search. – 2014.