

Designing Efficient Parallel Algorithms for Graph Problems

Weifa Liang

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

September 1997

© Weifa Liang

Typeset in Computer Modern by $\text{T}_{\text{E}}\text{X}$ and $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$.

Some of the results in Chapter 2 are joint work with Richard Brent and Hong Shen. These results have also appeared in [114, 121]. Chapter 3 is joint work with Brendan McKay and Hong Shen. The results of this chapter have also appeared in [120]. Chapter 4 is joint work with Brendan McKay. The content of this chapter has also appeared in [116]. Chapter 5 is joint work with George Havas and Brendan McKay. The results of this chapter have appeared in [115, 119]. Some of the results in Chapter 6 are the joint work with Xiaojun Shen [122]. These results are included in Sections 6.2, 6.3, and 6.5. Chapter 8 is joint work with Richard Brent. The contents of this chapter have also appeared in [113]. Some of Chapter 9 was carried out with Xiaojun Shen and Qing Hu. They are included in Sections 9.3 and 9.4. The results of this chapter have also appeared in [112, 123]. Chapter 10 is joint work with Brendan McKay, the contents of this chapter have also appeared in [118].

Except where otherwise indicated, this thesis is my own original work.

Weifa Liang
19 September 1997

Acknowledgments

I am truly indebted to my supervisors, Brendan McKay and Richard Brent, for their support and encouragement over the past three years. As a supervisor, Brendan made himself frequently available day and night to help me through research crises, clarify my ideas and explanations, and provide advice on research. He not only gave lots of his time and comments which influenced my writing but also showed me how to strive for a good presentation. His approach to algorithms, with the emphasis on identifying the combinatorial structure in the problem, the technique used and the search for simple and elegant solutions, has left a deep impact on me. I am sincerely thankful to him for being so generous with his time and for being so immensely patient with me during the course of my Ph.D. Richard was always available when I needed him. I enjoyed talking with him which always left me with some new ideas. I have learnt much from him about exploring a new research area. In addition to his technical expertise in many diverse areas, I have been particularly impressed by his scientific rigor and integrity, and I will strive to emulate these qualities in my own research. Both of my supervisors have had a great influence on my development as a researcher, their knowledge and intuition have been invaluable, and I am grateful to have been their student.

I would like to express my gratitude to Xiaojun Shen in The University of Missouri at Kansas City. It was a rare pleasure working with him when I was a visitor there. I have been benefited greatly from his insight, enthusiasm, and encouragement. I would also like to thank E. V. Krishnamurthy for his many suggestions and helpful comments in the preparation of this thesis.

I am grateful to my co-authors Richard Brent, George Havas, Qing Hu, Brendan McKay, Hong Shen, Xiaojun Shen, and Bingbing Zhou for their contributions to our collaborative work. Each has taught me a great deal about research and about writing about research.

Many other people have helped me in many different ways during my years in the Department of Computer Science at The Australian National University. They have helped to make my life in Canberra much easier and enjoyable. In particular, I would like to thank my colleagues Steve Blackburn who helped me with many of my \LaTeX questions over years, Bill Keating for his patient reading of my papers and providing his comments, Hongxue Wang for showing me lots of funny software. Many thanks are also due to our department head Brian Molinari for his generous financial sponsorship which has allowed me to attend several important international conferences, our system consultants Huge and Williamson for their help on my software problems, and our secretary Kathy for her administrative help.

Most of all, I especially want to thank my family. My parents for instilling in me the values I hold so dear and for giving me the confidence and drive to achieve my

goals. My wife and daughter for their love and patience that took me through many times of doubt and worry. Without their support and encouragement, this thesis could not possibly have been finished. I dedicate this thesis to them.

Abstract

Graph algorithms are concerned with the algorithmic aspects of solving graph problems. The problems are motivated from and have application to diverse areas of computer science, engineering and other disciplines. Problems arising from these areas of application are good candidates for parallelization since they often have both intense computational needs and stringent response time requirements. Motivated by these concerns, this thesis investigates parallel algorithms for these kinds of graph problems that have at least one of the following properties: the problems involve some type of dynamic updates; the sparsification technique is applicable; or the problems are closely related to communications network issues. The models of parallel computation used in our studies are the Parallel Random Access Machine (PRAM) model and the practical interconnection network models such as meshes and hypercubes.

Consider a communications network which can be represented by a graph $G = (V, E)$, where V is a set of *sites* (processors), and E is a set of *links* which are used to connect the sites (processors). In some cases, we also assign weights and/or directions to the edges in E . Associated with this network, there are many problems such as (i) whether the network is k -edge (k -vertex) connected with fixed k ; (ii) whether there are k -edge (k -vertex) disjoint paths between u and v for a pair of given vertices u and v after the network is dynamically updated by adding and/or deleting an edge etc; (iii) whether the sites in the network can communicate with each other when some sites and links fail; (iv) identifying the first k edges in the network whose deletion will result in the maximum increase in the routing cost in the resulting network for fixed k ; (v) how to augment the network at optimal cost with a given feasible set of weighted edges such that the augmented network is k -edge (k -vertex) connected; (vi) how to route messages through the network efficiently. In this thesis we answer the problems mentioned above by presenting efficient parallel algorithms to solve them. As far as we know, most of the proposed algorithms are the first ones in the parallel setting.

Even though most of the problems concerned in this thesis are related to communications networks, we also study the classic edge-coloring problem. The outstanding difficulty to solve this problem in parallel is that we do not yet know whether or not it is in NC. In this thesis we present an improved parallel algorithm for the problem which needs $O(\Delta^{4.5} \log^3 \Delta \log n + \Delta^4 \log^4 n)$ time using $O(n^2 \Delta + n \Delta^3)$ processors, where n is the number of vertices and Δ is the maximum vertex degree. Compared with a previously known result on the same model, we improved by an $O(\Delta^{1.5})$ factor in time. The non-trivial part is to reduce this problem to the edge-coloring update problem. We also generalize this problem to the approximate edge-coloring problem by giving a faster parallel algorithm for the latter case.

Throughout the design and analysis of parallel graph algorithms, we also find a

technique called the sparsification technique is very powerful in the design of efficient sequential and parallel algorithms on dense undirected graphs. We believe that this technique may be useful in its own right for guiding the design of efficient sequential and parallel algorithms for problems in other areas as well as in graph theory.

Contents

Acknowledgments	v
Abstract	vii
List of Figures	xiii
Abbreviations	xv
1 Introduction	1
1.1 The Parallel Computational Models	2
1.2 Notations and Terminology	4
1.3 The Problems and Related Literature Review	7
1.3.1 Testing for k -connectivity for fixed k	8
1.3.2 The optimal k -connectivity augmentation problem	8
1.3.3 Fully dynamic maintenance of k -connectivity	9
1.3.4 The partially dynamic maintenance of the solution of all pairs shortest paths	11
1.3.5 The k MVE MST problem and related problems	12
1.3.6 The approximate t -spanner problem	14
1.3.7 The edge-coloring problem and related problems	15
1.3.8 The maximal interlocking set problem	16
1.4 Overview of the Thesis	16
2 NC Algorithms for the Fully Dynamic Maintenance of k-Connectivity with $k = 2, 3$	21
2.1 Introduction	21
2.2 Preliminaries	24
2.2.1 Basic concepts	24
2.2.2 The sparse k -edge (k -vertex) certificate	24
2.2.3 Eppstein <i>et al</i> 's sparsification technique	25
2.3 The Maintenance of 2ECCs on Connected Graphs	26
2.3.1 The data structures	26
2.3.2 An algorithm for queries	29
2.3.3 Inserting and deleting edges	33
2.4 The Maintenance of 2- and 3-ECCs on General Graphs	34
2.4.1 2ECCs on disconnected graphs	34
2.4.2 An improved algorithm for 2ECCs	35
2.4.3 An algorithm for 3ECCs	35

2.5	The Maintenance of 2- and 3-VCCs on General Graphs	36
2.5.1	Finding a strong, sparse k -vertex certificate of G	36
2.5.2	The maintenance of 2VCCs	37
2.5.3	The maintenance of 3VCCs	38
2.6	Conclusions	38
3	NC Algorithms for the Partially Dynamic Maintenance of the Solution of All Pairs Shortest Paths	39
3.1	Introduction	39
3.2	An Algorithm for the All Pairs Shortest Paths Problem	41
3.3	Algorithms for Other Problems	43
3.3.1	Finding all pairs longest paths in a <i>DAG</i>	43
3.3.2	Topological sorting in a <i>DAG</i>	44
3.3.3	The dynamic transitive closure problem	45
3.4	Conclusions	46
4	NC Algorithms for Testing k-Connectivity of Directed and Undirected Graphs with Fixed k	47
4.1	Introduction	47
4.2	Directed graphs	48
4.3	Undirected graphs	51
4.4	NC reduction of edge connectivity to vertex connectivity	52
4.5	Conclusion	52
5	NC Approximation Algorithms for the Optimal k-Connectivity Augmentation Problem with Fixed k	53
5.1	Introduction	53
5.2	Preliminaries	56
5.3	2-Edge Connectivity Augmentation	57
5.4	Biconnectivity Augmentation	60
5.5	k -Edge Connectivity Augmentation	65
5.5.1	k -edge connectivity augmentation on a $(k - 1)$ -edge connected graph	65
5.5.2	k -edge connectivity augmentation on a general graph	72
5.6	Conclusions	73
6	Finding k Most Vital Edges in Minimum Spanning Trees	75
6.1	Introduction	75
6.2	Finding k Most Vital Edges	77
6.3	NC Algorithms for Finding the Single Most Vital Edge	80
6.4	Sequential and Parallel Algorithms for $k = 2, 3$	83
6.4.1	Finding the first two most vital edges	83
6.4.2	Finding the first three most vital edges	86
6.5	Sequential and Parallel Algorithms with Fixed $k \geq 4$	90
6.5.1	A sequential algorithm and its efficient implementation	90

6.5.2	A parallel implementation	94
6.6	Conclusions	96
7	Finding the Single Most Vital Edge on Mesh and Hypercube Processor Arrays	99
7.1	Introduction	99
7.2	The Computational Models	100
7.2.1	The 2-D mesh array	100
7.2.2	The hypercube array	101
7.3	Algorithms on the 2-D Mesh Array	101
7.3.1	Finding the single most vital edge <i>w.r.t.</i> the MST problem . . .	101
7.3.2	Finding the single most vital edge <i>w.r.t.</i> SPs	107
7.4	Algorithms on the Hypercube Array	109
7.5	Conclusions	113
8	Constructing the Spanners of Graphs in Parallel	115
8.1	Introduction	115
8.2	Preliminaries	116
8.3	Finding Approximate t -Spanners	117
8.3.1	Unweighted Graphs	117
8.3.2	Weighted Graphs	120
8.4	NC Algorithms for Finding Moderate $2t$ -Spanners	123
8.4.1	The algorithm for unweighted graphs	123
8.4.2	The algorithm for weighted graphs	126
8.5	Conclusions	127
9	Parallel Algorithms for Edge-Coloring	129
9.1	Introduction	129
9.2	Concepts and Notations	130
9.3	An Edge-Coloring Update Algorithm	131
9.4	A New Edge-Coloring Algorithm	133
9.4.1	A simple version of the Karloff-Shmoys' algorithm	133
9.4.2	An improved edge-coloring algorithm	136
9.5	An Approximate Edge-Coloring Algorithm	140
9.5.1	Graph decomposition	140
9.5.2	An algorithm for approximate edge-coloring	142
9.6	Conclusions	145
10	An NC Algorithm for Computing Maximal Interlocking Sets	147
10.1	Introduction	147
10.2	Preliminaries	148
10.3	The Algorithm	149
10.3.1	The algorithmic description	149
10.3.2	The correctness proof	151
10.4	Conclusion	153

11 Conclusions and Open Problems	155
Bibliography	157

List of Figures

2.1 An illustration of the proof of Lemma 2.12. 32

5.1 An example. 67

9.1 Inverting the $\alpha\beta$ -path of v_2 affects the free color of w_1 in a fan of v_1 , and
inverting the $\alpha\beta$ -path of v_3 affects the free color of $z(v_2)$ in a fan of v_2 . . 135

10.1 An illustration. 153

Abbreviations

ANLV	All nearest larger value
ANSV	All nearest smaller value
CC	Connected component
CREW	Concurrent read and exclusive write
CRCW	Concurrent read and concurrent write
DAG	Directed acyclic graph
EREW	Exclusive read and exclusive write
$LCA(x, y)$	The lowest common ancestor of x and y
MST	Minimum spanning tree
MSF	Minimum spanning forest
MWSC	Minimum weighted set cover
PRAM	Parallel random access machine
2ECC	2-Edge connected component
3ECC	3-Edge connected component
2VCC	2-Vertex connected component
3VCC	3-Vertex connected component (triconnected component)
k MVE	The k most vital edges
SP	Shortest path

