

# Lecture Notes in Computer Science

1089

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Advisory Board: W. Brauer D. Gries J. Stoer

G. Ramalingam

# Bounded Incremental Computation



Springer

**Series Editors**

Gerhard Goos, Karlsruhe University, Germany

Juris Hartmanis, Cornell University, NY, USA

Jan van Leeuwen, Utrecht University, The Netherlands

**Author**

G. Ramalingam

IBM T.J. Watson Research Center

P.O. Box 704, Yorktown Heights, NY 10598, USA

Cataloging-in-Publication data applied for

**Die Deutsche Bibliothek - CIP-Einheitsaufnahme**

**Ramalingam, G.:**

Bounded incremental computation / G. Ramalingam. - Berlin ; Heidelberg ; New York ; Barcelona ; Budapest ; Hong Kong ; London ; Milan ; Paris ; Santa Clara ; Singapore ; Tokyo : Springer, 1996

(Lecture notes in computer science ; 1089)

ISBN 3-540-61320-X

NE: GT

CR Subject Classification (1991): F.2, G.2, F.1, D.4.7, D.2.6, D.3.4

ISBN 3-540-61320-X Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1996

Printed in Germany

Typesetting: Camera-ready by author

SPIN 10513144 06/3142 - 5 4 3 2 1 0 Printed on acid-free paper

*to*  
*my mother, my father, and Raji*  
*with love*

என் தாய், என் தந்தை, ராஜி மூவருக்கும்

அன்புடன்

## Preface

Algorithms and programs transform their input into their output. *Incremental computation* concerns the re-computation of output after a change in the input. An incremental algorithm, consequently, transforms a *change in input* into a *change in output*. Incremental algorithms, also called dynamic or on-line algorithms, are becoming increasingly important given the popularity of interactive systems, which must respond efficiently to a user's actions, which are usually "modifications" of an "input document".

In the context of incremental computation, small changes in the input are likely to cause correspondingly small changes in the output. It is natural to attempt to identify the part of the previous output that is no longer "correct" and "update" it. Where it is not possible to identify the affected part of the output *exactly*, an incremental algorithm may attempt to identify a *conservative approximation* (that is, an over-estimation) of the affected part of the output. Given some part of the output that needs to be recomputed, an incremental algorithm would benefit by processing only the portion of the input it needs to process in order to generate that part of the output. The effectiveness or efficiency of an incremental algorithm depends on how accurate an approximation to the affected region it can identify and on the overhead it incurs in doing this.

This book presents results — upper bound results, lower bound results, and experimental results — for several incremental computation problems. What is common to all these results is that we seek to determine the efficiency of an algorithm by analyzing how accurate an approximation to the affected region it identifies and on the overhead it incurs in doing this. In particular, we try to analyze the complexity of incremental algorithms and problems in terms of a parameter  $\|\delta\|$  that measures the *size of the change in the input and output*. An incremental algorithm is said to be *bounded* if the time it takes to update the output depends only on the size of the *change in the input and output* (i.e.,  $\|\delta\|$ ), and not on the size of the *entire* current input. Otherwise, an incremental algorithm is said to be *unbounded*. A problem is said to be *bounded* (unbounded) if it has (does not have) a bounded incremental algorithm. The results established in this book illustrate a complexity hierarchy for incremental computation from this point of view. These results are summarized below.

We present efficient  $O(\|\delta\| \log \|\delta\|)$  incremental algorithms for several shortest-path problems — the single-sink shortest-path problem, the all-pairs shortest-path problem, and a generalization of the single-sink shortest-path problem due to Knuth — establishing that these problems are all polynomially bounded. These results show that it is possible in these problems to identify, without much of an overhead, exactly the part of the output that needs to be updated.

We present an  $O(2^{\|\delta\|})$  incremental algorithm for the circuit value annotation problem, which matches a previous  $\Omega(2^{\|\delta\|})$  lower bound for this problem. Consequently, this establishes that the circuit value annotation problem is an exponentially bounded problem. We also present experimental results that show that our algorithm,

in spite of a worst-case complexity of  $\Theta(2^{\|\delta\|})$ , works well in practice, often identifying a very good approximation to the affected output with very little overhead.

We present lower bounds showing that a number of problems, including graph reachability, dataflow analysis, and algebraic path problems, are unbounded with respect to a model of computation called the *sparsely-aliasing pointer machine* model.

We present an  $O(\|\delta\| \log n)$  incremental algorithm for the reachability problem in reducible flowgraphs, which identifies the affected output exactly, but with an  $O(\log n)$  factor overhead.

We present an algorithm for maintaining the dominator tree of a reducible flowgraph, which identifies a reasonable approximation to the affected output and updates it efficiently.

## Acknowledgements

This book is a slightly revised version of my Ph.D. thesis. I am greatly indebted to my thesis advisor Thomas Reps for his guidance, both personal and professional, both while I was a student and after my graduation, which, among other things, shaped the thesis and its form. I thank also

- Susan Horwitz and Marvin Solomon, for their patient reading of my thesis and their valuable suggestions which helped improve the thesis;
- Eric Bach and Richard Brualdi, the other members of my thesis committee;
- Prason Tiwari, for various useful discussions I had with him;
- Charles Fischer, for his excellent teaching;
- the members of the “PL” group, in particular, Paul Adams, Tom Ball, Sam Bates, Dave Binkley, Lorenz Huelsbergen, Phil Pfeiffer, Todd Proebsting, and Wu Yang;
- IBM, for partially supporting me through a graduate fellowship while I was a student, and for its support subsequently;
- my friends, too numerous to list here, who made my stay at Madison, Wisconsin, a memorable one;

I especially thank my family, without whose support this would not have been possible. I thank my parents, my wife, and my sister for their love and support over the years.

May 1996

G. Ramalingam

# Table Of Contents

<b>Chapter 1. Introduction .....</b>	<b>1</b>
<b>Chapter 2. On Incremental Algorithms and Their Complexity .....</b>	<b>7</b>
2.1. Incremental Algorithms: Why and Where? .....	7
2.2. Why Incremental Graph Algorithms? .....	9
2.2.1. The Reachability Problem .....	9
2.2.2. The Shortest-Path Problem .....	10
2.2.3. The Circuit Value Annotation Problem .....	10
2.2.4. The Dominator Tree Problem .....	11
2.3. Evaluating Incremental Algorithms: Why and How? .....	11
2.3.1. Analytical Evaluation of Incremental Algorithms .....	12
2.3.1.1. Asymptotic Analysis Versus Micro-analysis .....	13
2.3.1.2. Alternatives to Worst-Case Analysis .....	13
2.3.1.3. Alternatives to Input Size as Complexity Parameters .....	16
2.3.1.4. On Classifying Incremental Computation Problems .....	20
2.3.2. Experimental Evaluation of Incremental Algorithms .....	22
<b>Chapter 3. Terminology and Notation .....</b>	<b>25</b>
<b>Chapter 4. Incremental Algorithms for Shortest-Path Problems .....</b>	<b>30</b>
4.1. The Dynamic Single-Sink Shortest-Path Problem .....	30
4.1.1. Deletion of an Edge .....	31
4.1.2. Insertion of an Edge .....	36
4.1.3. Handling Edges of Non-Positive Length .....	38
4.1.4. A Batch SSSP Algorithm .....	42
4.2. The Dynamic All-Pairs Shortest-Path Problem .....	43
4.2.1. Deletion of an Edge .....	45
4.2.2. Insertion of an Edge .....	48
4.2.3. Handling Edges of Non-Positive Length .....	51
4.3. Related Work .....	51
<b>Chapter 5. Generalizations of the Shortest-Path Problem .....</b>	<b>52</b>
5.1. An Overview of the Chapter .....	52
5.2. Maximum Fixed Point Problems .....	55

5.3. Path Problems in Graphs .....	56
5.3.1. Algebraic Path Problems .....	56
5.3.2. Dataflow Analysis Problems .....	61
5.3.3. The Relation between Dataflow Analysis and Algebraic Path Problems .....	62
5.4. Grammar Problems .....	63
5.4.1. The Idea Behind The Problem .....	63
5.4.2. The Problem Definition .....	66
5.4.3. The Grammar Problem as a Maximum Fixed Point Problem .....	69
5.4.4. The SSF Grammar Problem and the SWSF Fixed Point Problem .....	71
5.5. Related Work .....	77

<b>Chapter 6. An Incremental Algorithm for a Generalization of the     Shortest-Path Problem .....</b>	<b>78</b>
6.1. Introduction .....	78
6.2. The Idea Behind the Algorithm .....	79
6.3. The Algorithm .....	85
6.4. An Improved Algorithm .....	87
6.5. Extensions to the Algorithm .....	92
6.5.1. Answering Queries on Demand .....	92
6.5.2. Maintaining Minimum Cost Derivations .....	93
6.5.3. The All-Pairs Shortest-Path Problem .....	94
6.5.4. Handling Edges of Non-Positive Length .....	94
6.6. Related Work .....	95

<b>Chapter 7. Incremental Algorithms for the Circuit Value     Annotation Problem .....</b>	<b>101</b>
7.1. Introduction .....	101
7.2. The Change Propagation Strategy .....	103
7.3. The Iterative Evaluate-and-Expand Strategy .....	107
7.4. Breadth-First Expansion .....	110
7.4.1. Bounded-Outdegree Circuits .....	110
7.4.2. Handling Unbounded Outdegree .....	112
7.4.3. Monotonic Circuits: A Special Case .....	114



7.4.4. A Tradeoff .....	115
7.5. The Weighted Circuit Value Annotation Problem .....	119
7.6. The Empirical Boundedness of Incremental Algorithms .....	124
7.7. Some Remarks .....	128
<b>Chapter 8. Inherently Unbounded Incremental Computation</b>	
<b>Problems</b> .....	130
8.1. Introduction .....	130
8.2. The Model of Computation .....	130
8.2.1. Locally Persistent Algorithms .....	130
8.2.2. The Cost of Elementary Operations .....	131
8.2.3. Sparsely Aliasing Algorithms .....	133
8.3. The Unboundedness of Reachability .....	137
8.4. Other Unbounded Problems .....	142
8.4.1. Unbounded Path Problems .....	142
8.4.2. Unbounded Dataflow Analysis Problems .....	144
8.5. Some Remarks .....	146
<b>Chapter 9. Incremental Algorithms for Reducible Flowgraphs</b> .....	147
9.1. Introduction .....	147
9.2. Reachability, Domination, and Reducible Flowgraphs .....	147
9.3. The Dynamic Single-Source Reachability Problem .....	149
9.3.1. Insertion of an Edge .....	153
9.3.2. Deletion of an Edge .....	153
9.3.3. Handling Irreducible Flowgraphs .....	153
9.4. The Dynamic Dominator Tree Problem .....	156
9.4.1. Insertion of an Edge .....	158
9.4.2. Deletion of an Edge .....	164
9.4.3. Related Work .....	166
9.4.4. Some Remarks .....	167
<b>Chapter 10. Conclusions</b> .....	169
<b>Bibliography</b> .....	172