PROBLEM SOLVING IN ALGORITHMS

A RESEARCH APPROACH

Sanpawat Kantabutra
Faculty of Engineering, CMU

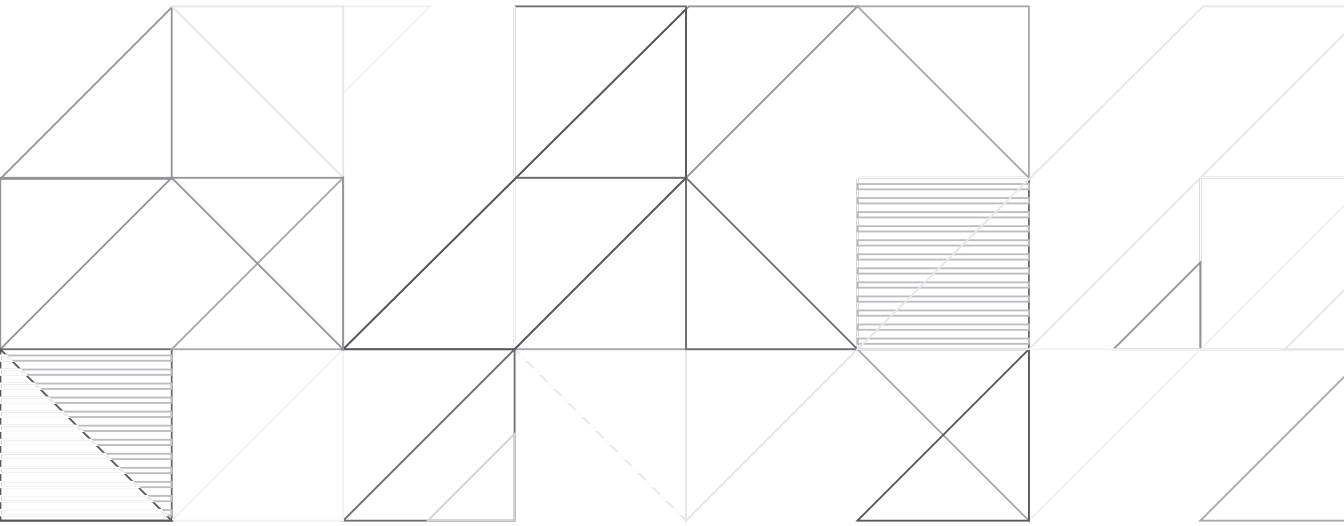
CURRICULUM VITAE

Sanpawat Kantabutra, PhD

Sanpawat Kantabutra is currently Associate Professor in the Theory of Computation Group in the Faculty of Engineering in Chiang Mai University, Thailand. He is also a Thailand Research Fund Career Award researcher and a researcher in the Research Center for Quantum Technology in Chiang Mai University. He earned his PhD in theoretical computer science from Tufts University in Boston in the United States and has regularly published his research findings in some of the world's top journals in theoretical computer science. His current research interests are in approximation and randomized algorithms and quantum computation.

PROBLEM SOLVING IN ALGORITHMS

A RESEARCH APPROACH



Sanpawat Kantabutra

Faculty of Engineering, CMU

Problem Solving in Algorithms

A Research Approach

Editor: Kanchana Kanchanasut
ISBN (e-Book) : 978-616-398-549-1
Author: Sanpawat Kantabutra
Published: Chiang Mai University Press
Office of Research Administration Center
Tel: +66 (0) 5394 3603-4
Fax: +66 (0) 5394 3600
https://cmupress.cmu.ac.th
E-mail: cmupress.th@gmail.com

Printed: March 2021
Price: 315 Baht

Cover design: Somsib Craft

Contact: Chiang Mai University Press
Tel: +66 (0) 53 94 3605 Fax: +66 (0) 53 94 3600
https://cmupress.cmu.ac.th, E-mail: cmupress.th@gmail.com

Appreciation

This book aims to develop critical thinking skills through problem solving. Each chapter is based on published research by the author and demonstrates a detailed thinking process for a specific problem that may arise in a diverse selection of research areas such as communication, biology, data mining, computer architecture and general computer science. The author begins each chapter by describing a single problem of interest and motivations for this study before diving into formal treatments of the topic and ending with comments, suggestions and open problems for further research in the conclusion. The book is modular with self-contained chapters suitable as a reference book for researchers or as a textbook for graduate students in algorithms.

The author emphasizes firm theoretical understanding of each problem with related complexities. Formal treatment of cyber security is an example of such an approach where the problem is properly defined with two computing problems from the economic perspectives of the attackers and defenders. The author first defines a cyber security model, discusses two computing solutions and analyses their complexities that turn out to be NP-complete. Approximation algorithms are then introduced.

The topics selected are relevant and important for the digital transformation age which is still lacking theoretical treatments. The author has made an attempt to fill this gap and stimulate interest.

Graduate students will not only learn from the materials covered, but also from the research approaches taken in this book.

Professor Kanchana Kanchanasut
School of Engineering and Technology
Asian Institute of Technology

Foreword

It is my great pleasure to write this foreword. In his latest book, *Problem Solving in Algorithms: A Research Approach*, Dr. Sanpawat Kantabutra provides an innovative and refreshing perspective on how to go about solving a problem in a computing discipline. To the extent possible, he walks the reader through the equivalent of the scientific method for computing. Although the work is largely abstract, Dr. Kantabutra does a good job in making it feel concrete and hands-on. His approach is a nuts-and-bolts one, where sometimes a hypothesis leads to a dead end. And, as Dr. Kantabutra points out, such dead ends are part of the process of doing successful research. Along the way to solving difficult problems, failures are encouraged and to be expected. The road to achieving the final answer is a long and winding one, but a very rewarding journey in the long run. This blue-collar approach to critical thinking and problem solving will be of great benefit to many students, researchers, and faculty members alike.

I applaud Dr. Kantabutra for including chapters on an extremely diverse set of topics. He includes material on networking, graph labeling, wireless communication, hierarchical clustering, cyber-security, partitions, graph theory, parallel computation, and clustering. In all cases, he makes a serious effort to provide the reader with the background necessary to comprehend these different domains. Along another dimension, he explores algorithms from a wide variety of angles—sequential, parallel (on several different models), and randomized. Each different topic, combined with a different algorithmic framework, provides him with a rich playing field; one where he can touch on many aspects problem-solving issues and critical-thinking skills. Dr. Kantabutra takes full advantage of all the various options to cover a wide range of techniques and thought processes. Each different model that he describes gives the reader the opportunity to experiment in a broad range of settings.

The book fulfills Dr. Kantabutra's goals of teaching the reader how to problem solve and go about conducting research in any computing domain. The techniques introduce the reader to new ways of thinking and methods for viewing problems. He helps the reader develop a problem-solving toolkit, as it were. When a reader gets stuck in an effort to solve a problem, this book contains techniques that can help guide research or help one to shift directions, say by trying a new process or methodology or asking another relevant question. Dr. Kantabutra takes the reader on a problem-solving journey, and one feels as though he is there accompanying you on that journey—providing encourage-

ment, guidance, strategies, wisdom, and support. The book can help boost one's confidence in going about research. I strongly recommend the book to advanced undergraduate students with a foundation in a computing discipline, graduate students, instructors, and professors who want to get involved in research. The book is well-written, friendly, easy-to-use, and highly innovative. Kudos to Dr. Kantabutra for taking on another ambitious project and seeing it through to an excellent conclusion.

Dr. Raymond Greenlaw
July 20, 2020
Office of Naval Research Distinguished Chair in Cyber-Security, Retired
United States Naval Academy

Preface

This *Problem Solving in Algorithms* book is designed to illustrate how research in theoretical computer science is typically conducted. In particular, some commonly found proof techniques in the design and analysis of algorithms and some important mathematical ideas are discussed in length. Unsurprisingly, our approach in composing this book is research-based because we feel that it is the best way to practice critical thinking and problem-solving skills. Topics in each chapter are chosen such that they are diverse, interesting, and recent. Each chapter can also be used separately as a case study in class. There are ten chapters in this book, covering topics ranging from data clustering, to cyber security, to wireless communication, to graph theory, and to parallel computation. All materials in these chapters are selected from a pool of our recent research publications. In each chapter we first discuss how a theoretical model is defined and the reasons that it is defined in a particular way. We then show how a problem is defined with respect to the theoretical model. Possible proof techniques and the reasons that they might be or might not be used are surveyed. Additionally, an alternative is shown when one is available. In illustrating our points we discuss possible solutions. Of course, some of them might be wrong. But this is what usually happens in research or in problem solving. In fact, it is our intention to show several wrong solutions to a problem because to realize that something is wrong is an essential part of research and self discovery. At the end of each chapter we provide a problem set. A problem marked with an asterisk may be either open, difficult, or time-consuming. This book aims at graduate students, researchers, and computer professionals in computer science or related fields. We expect that readers should have a strong background in the design and analysis of algorithms and discrete mathematics at the undergraduate level. Readers should feel comfortable with basic mathematical proof techniques such as mathematical induction, proof by counter examples, and proof by contradiction. This book is not an introduction to complexity and algorithms. However, when there are certain things that we feel readers might not already have learnt in class, a review and/or a book reference are given.

Dr. Sanpawat Kantabutra
The Theory of Computation Group
July 20, 2020

Contents

List of Figures

List of Tables

Chapter 1

Network Embedding

"*When you get stuck, do it over.*"

In this chapter we introduce a concept of *network embedding*. Roughly speaking, network embedding is a way to use another network A when your physical network is not A. For instance, if your physical network is a ring network but your communication protocol is based on a line network, how are you going to make your communication protocol work without buying a new expensive line network or totally changing the communication protocol to suit a ring network? The first thing to observe is that a line network is a subnetwork of a ring network. Hence, if we could "designate" the edges in the ring network to match those of the line network somehow, our line-based communication protocol would work. However, if our physical network is a hypercube network and our communication protocol is based on a tree network, it is not clear whether a tree network is a subnetwork of a hypercube network. Indeed, if a tree network has many nodes, it is not a subnetwork of a hypercube network. In this case we could still "designate" the edges but with some slowdown factor in speed. For the rest of this chapter we consider a new theoretical network called *Completely Overlapping Network* (CON) and we want to embed a d dimensional hypercube network into a completely overlapping network. Additionally, taking advantage of faster and more powerful processors and a large gap between communication and computation speeds, we also show that the embedded d-dimensional hypercube can be scaled using latency hiding. Related mathematical properties are also shown for optimal scaling. This chapter elucidates the ideas and thinking processes in the original work by Kantabutra and Chawachat [23].

1.1 Introduction

Embedding problems are well known among graph theorists, mathematicians, as well as computer scientists. Typically, embedding problems are described in

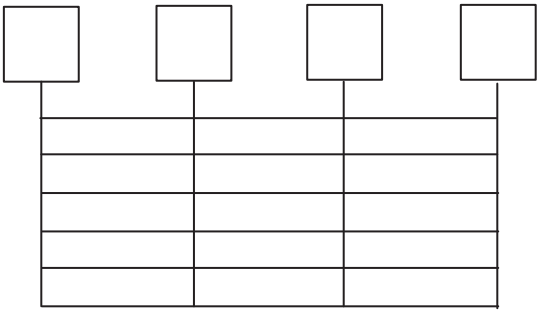
graph theory terms but in our context we describe in computer network terms. There is a lot of motivation behind embedding parallel architectures [1], one of which is the ability to allow one network to operate efficiently in another network with different architectures. Suppose we have two networks A and B. If our physical network is A and we want to run our communication protocol based on B, we "embed" network B into network A. Network A is called an *embedding* or *host* network and network B is called an *embedded* or *guest* network. In other words, we simulate network B in network A.

We consider a binary hypercube in this chapter. The binary hypercube is one of the most versatile and efficient interconnection networks yet discovered for parallel computation. One of the biggest reasons for the popularity of the hypercube is its ability to being both host and guest networks efficiently. In terms of being a host network, a lot of research has been carried out on embedding different topologies into the hypercube. Following are some examples. In [2] and [3] a hypercube is embedded with hierarchical networks. In [4]-[6] authors show how to embed a few kinds of grids into hypercubes. Many kinds of trees can also be embedded into hypercubes as in [7]-[10]. Others embed star networks into hypercubes [11], or shuffle networks into hypercubes [12]. In [13] authors show how to embed a hyper-pyramid into a hypercube. On the other hand, the hypercube can also be a *guest* network. Many parallel algorithms use hypercubes as the communication topology among their processes. Some authors, for instance, try to embed hypercubes into toruses and rings as in [14]-[15]. Others try to embed a hypercube into a mesh as in [16]. In a more mathematical flavor Cayley graphs are shown to be a host for hypercubes in [17]. In area of optical networks, an efficient embedding of a hypercube in a Wavelength Division Multiplexing network is shown in [18].

Motivated by faster Ethernet lines and more powerful processors of the past two decades, a group of computer scientists developed an experimental network called *overlapping network* [19]-[21]. Since connecting several computer nodes to a single Ethernet line limited the shared communication channel, they worked on the concept of using multiple Ethernet lines in some certain configurations. These configurations are in the general classification of overlapping connectivity networks. Overlapping connectivity networks have the characteristic that regions of connectivity are provided and the regions overlap so as to provide parallelism. To generalize their overlapping network for the purpose of our study, more line segments have been added to their overlapping network for a complete parallel connectivity; hence, the name completely overlapping network. A four-node completely overlapping network is shown in Figure 1.1.

At this point several questions should arise. What does an n-node completely overlapping network look like? How many line segments are there in terms of n? How do two nodes communicate in an n-node completely overlapping network? What is the communication protocol? How do we compute time and communication complexities? Because this *is* our research, we have to think about the answers to these questions and eventually give them a model and related definitions.

Figure 1.1: Four-node completely overlapping network



1.2 CON and its Related Definitions

Generally, we want our theoretical model and definitions to reflect reality but there is always a tradeoff to consider. If we design our model to truly reflect reality, we probably have to add every single detail into our model. Every bit of details increases the complexity of our model. The complexity might be so much that we get bogged down in details and are not able to see the whole picture. On the other hand, if our model is too simple, we might not be able to gain insights into anything. Hence, a balance has to be made when we create a model and its related definitions.

We begin with defining an n-node completely overlapping network. Traditionally, a network is described in a graph theory term. A graph has two components. A graph $G = (V, E)$ is a set V of vertices and a set E of edges. If we are going to use a graph G to represent CON, how do we relate V and E to CON? After some thoughts, we realize that two different kinds of vertices are needed because an n-node CON contains non-computer and computer vertices. If we are going to proceed in this direction, we would have $V = V_{\diamond} \cup V_{\triangle}$ and we would have to differentiate the two types of vertices in our communication protocol every time. It seems more complex and tedious than necessary. Can we use some other simpler representation that still suits our purpose? Yes, we can. Because a completely overlapping network is composed of several overlapped communication line segments that connect among several nodes or processors to provide parallelism as shown in Figure 1.1, we can define the network in terms of vertical and horizontal communication line segments. What else do we need to consider putting it in our definition of CON besides the line segments and nodes? None. Hence, we have everything at this point and we define a completely overlapping network using a 3-tuple containing nodes, horizontal line segments, and vertical line segments. The following is the formal definition of an n-node completely overlapping network.

Definition 1 (Completely Overlapping Network)**.** *A* completely overlapping network *is a 3-tuple* (N, H, V)*, where* N*,* H*, and* V *are all finite sets, and*

1. N *is the set of* n *nodes,*