

Springer Series in the Data Sciences

Lijun Chang · Lu Qin

# Cohesive Subgraph Computation over Large Sparse Graphs

Algorithms, Data Structures,  
and Programming Techniques

 Springer

# Springer Series in the Data Sciences

Series Editors:

David Banks, Duke University, Durham

Jianqing Fan, Princeton University, Princeton

Michael Jordan, University of California, Berkeley

Ravi Kannan, Microsoft Research Labs, Bangalore

Yurii Nesterov, Universite Catholique de Louvain, Louvain-la-Neuve

Christopher Ré, Stanford University, Stanford

Ryan Tibshirani, Carnegie Mellon University, Pittsburgh

Larry Wasserman, Carnegie Mellon University, Pittsburgh

Springer Series in the Data Sciences focuses primarily on monographs and graduate level textbooks. The target audience includes students and researchers working in and across the fields of mathematics, theoretical computer science, and statistics.

Data Analysis and Interpretation is a broad field encompassing some of the fastest-growing subjects in interdisciplinary statistics, mathematics and computer science. It encompasses a process of inspecting, cleaning, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision making. Data analysis has multiple facets and approaches, including diverse techniques under a variety of names, in different business, science, and social science domains. Springer Series in the Data Sciences addresses the needs of a broad spectrum of scientists and students who are utilizing quantitative methods in their daily research.

The series is broad but structured, including topics within all core areas of the data sciences. The breadth of the series reflects the variation of scholarly projects currently underway in the field of machine learning.

More information about this series at <http://www.springer.com/series/13852>

Lijun Chang • Lu Qin

# Cohesive Subgraph Computation over Large Sparse Graphs

Algorithms, Data Structures,  
and Programming Techniques

 Springer

Lijun Chang  
School of Computer Science  
The University of Sydney  
Sydney, NSW, Australia

Lu Qin  
Centre for Artificial Intelligence  
University of Technology Sydney  
Sydney, NSW, Australia

ISSN 2365-5674 ISSN 2365-5682 (electronic)  
Springer Series in the Data Sciences  
ISBN 978-3-030-03598-3 ISBN 978-3-030-03599-0 (eBook)  
<https://doi.org/10.1007/978-3-030-03599-0>

Library of Congress Control Number: 2018962869

Mathematics Subject Classification: 05C85, 05C82, 91D30

© Springer Nature Switzerland AG 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

*To my wife, Xi  
my parents, Qiyuan and Yumei  
Lijun Chang*

*To my wife, Michelle  
my parents, Hanmin and Yaping  
Lu Qin*

# Preface

Graph model has been widely used to represent the relationships among entities in a wide spectrum of applications such as social networks, communication networks, collaboration networks, information networks, and biological networks. As a result, we are nowadays facing a tremendous amount of large real-world graphs. For example, SNAP [49] and Network Repository [71] are two representative graph repositories hosting thousands of real graphs. An availability of rich graph data not only brings great opportunities for realizing big values of data to serve key applications but also brings great challenges in computation.

The main purpose of this book is to survey the recent technical developments on efficiently processing *large sparse graphs*, in view of the fact that real graphs are usually sparse graphs. Algorithms designed for large sparse graphs should be analyzed with respect to the number of edges in a graph, and ideally should run in linear or near-linear time to the number of edges. In this book, we illustrate the general techniques and principles, toward efficiently processing large sparse graphs with millions of vertices and billions of edges, through the problems of *cohesive subgraph computation*. Although real graphs are sparsely connected from a global point of view, they usually contain subgraphs that are locally densely connected [11]. Computing cohesive/dense subgraphs can either be the main goal of a graph analysis task or act as a preprocessing step aiming to reduce/trim the graph by removing sparse/unimportant parts such that more complex and time-consuming analysis can be conducted. In the literature, the cohesiveness of a subgraph is usually measured by *the minimum degree*, *the average degree*, or *their higher-order variants*, or *edge connectivity*. Cohesive subgraph computation based on different cohesiveness measures extracts cohesive subgraphs with different properties and also requires different levels of computational efforts.

The book can be used either as an extended survey for people who are interested in cohesive subgraph computation or as a reference book for a postgraduate course on the related topics, or as a guideline book for writing effective C/C++ programs to efficiently process real graphs with billions of edges. In this book, we

will introduce algorithms, in the form of pseudocode, analyze their time and space complexities, and also discuss their implementations. C/C++ codes for all the data structures and some of the presented algorithms are available at the author's GitHub website.<sup>1</sup>

**Organization.** The book is organized as follows.

In Chapter 1, we present the preliminaries of large sparse graph processing, including characteristics of real-world graphs and the representation of large sparse graphs in main memory. In this chapter, we also briefly introduce the problems of cohesive subgraph computation over large sparse graphs and their applications.

In Chapter 2, we illustrate three data structures (specifically, linked list-based linear heap, array-based linear heap, and lazy-update linear heap) that are useful for algorithm design in the remaining chapters of the book.

In Chapter 3, we focus on minimum degree-based graph decomposition (aka *core decomposition*); that is, compute the maximal subgraphs with minimum degree at least  $k$  (called  $k$ -core), for all different  $k$  values. We present an algorithm to conduct core decomposition in  $\mathcal{O}(m)$  time, where  $m$  is the number of edges in a graph, and also discuss  $h$ -index-based local algorithms that have higher time complexities but can be naturally parallelized.

In Chapter 4, we study the problem of computing the subgraph with the maximum average degree (aka, *densest subgraph*). We present a 2-approximation algorithm that has  $\mathcal{O}(m)$  time complexity, a  $2(1 + \epsilon)$ -approximation streaming algorithm, and also an exact algorithm based on minimum cut.

In Chapter 5, we investigate higher-order variants of the problems studied in Chapters 3 and 4. As the building blocks of higher-order analysis of graphs are  $k$ -cliques, we first present triangle enumeration algorithms that run in  $\mathcal{O}(\alpha(G) \times m)$  time and  $k$ -clique enumeration algorithms that run in  $\mathcal{O}(k \times (\alpha(G))^{k-2} \times m)$  time, where  $\alpha(G)$  is the arboricity of a graph  $G$  and satisfies  $\alpha(G) \leq \sqrt{m}$  [20]. Then, we discuss how to extend the algorithms presented in Chapters 3 and 4 for higher-order core decomposition (specifically, truss decomposition and nucleus decomposition) and higher-order densest subgraph computation (specifically,  $k$ -clique densest subgraph computation), respectively.

In Chapter 6, we discuss edge connectivity-based graph decomposition. Firstly, given an integer  $k$ , we study the problem of computing all maximal  $k$ -edge connected subgraphs in a given input graph. We present a graph partition-based approach to conduct this in  $\mathcal{O}(h \times l \times m)$  time, where  $h$  and  $l$  are usually bounded by small constants for real-world graphs. Then, we present a divide-and-conquer approach, which invokes the graph partition-based approach as a building block, for computing the maximal  $k$ -edge connected subgraphs for all different  $k$  values in  $\mathcal{O}((\log \alpha(G)) \times h \times l \times m)$  time.

---

<sup>1</sup> [https://github.com/LijunChang/Cohesive\\_subgraph\\_book](https://github.com/LijunChang/Cohesive_subgraph_book).

**Acknowledgments.** This book is partially supported by Australian Research Council Discovery Early Career Researcher Award (DE150100563).

Sydney, NSW, Australia  
Sydney, NSW, Australia  
September 2018

Lijun Chang  
Lu Qin



# Contents

<b>1</b>	<b>Introduction</b> .....	1
1.1	Background .....	1
1.1.1	Graph Terminologies .....	1
1.1.2	Real Graph Datasets .....	3
1.1.3	Representation of Large Sparse Graphs .....	4
1.1.4	Complexity Analysis .....	6
1.2	Cohesive Subgraphs .....	6
1.2.1	Cohesive Subgraph Computation .....	7
1.2.2	Applications .....	8
<b>2</b>	<b>Linear Heap Data Structures</b> .....	9
2.1	Linked List-Based Linear Heap .....	9
2.1.1	Interface of a Linked List-Based Linear Heap .....	10
2.1.2	Time Complexity of <code>ListLinearHeap</code> .....	13
2.2	Array-Based Linear Heap .....	14
2.2.1	Interface of an Array-Based Linear Heap .....	15
2.2.2	Time Complexity of <code>ArrayLinearHeap</code> .....	18
2.3	Lazy-Update Linear Heap .....	18
<b>3</b>	<b>Minimum Degree-Based Core Decomposition</b> .....	21
3.1	Preliminaries .....	21
3.1.1	Degeneracy and Arboricity of a Graph .....	22
3.2	Linear-Time Core Decomposition .....	23
3.2.1	The Peeling Algorithm .....	23
3.2.2	Compute $k$ -Core .....	26
3.2.3	Construct Core Hierarchy .....	27
3.3	Core Decomposition in Other Environments .....	32
3.3.1	$h$ -index-Based Core Decomposition .....	32
3.3.2	Parallel/Distributed Core Decomposition .....	36
3.3.3	I/O-Efficient Core Decomposition .....	37
3.4	Further Readings .....	39

<b>4</b>	<b>Average Degree-Based Densest Subgraph Computation</b>	41
4.1	Preliminaries	41
4.1.1	Properties of Densest Subgraph	42
4.2	Approximation Algorithms	43
4.2.1	A 2-Approximation Algorithm	43
4.2.2	A Streaming $2(1 + \varepsilon)$ -Approximation Algorithm	45
4.3	An Exact Algorithm	47
4.3.1	Density Testing	47
4.3.2	The Densest-Exact Algorithm	50
4.3.3	Pruning for Densest Subgraph Computation	52
4.4	Further Readings	52
<b>5</b>	<b>Higher-Order Structure-Based Graph Decomposition</b>	55
5.1	$k$ -Clique Enumeration	55
5.1.1	Triangle Enumeration Algorithms	55
5.1.2	$k$ -Clique Enumeration Algorithms	62
5.2	Higher-Order Core Decomposition	64
5.2.1	Truss Decomposition	64
5.2.2	Nucleus Decomposition	68
5.3	Higher-Order Densest Subgraph Computation	71
5.3.1	Approximation Algorithms	71
5.3.2	Exact Algorithms	73
5.4	Further Readings	75
<b>6</b>	<b>Edge Connectivity-Based Graph Decomposition</b>	77
6.1	Preliminaries	77
6.2	Deterministic $k$ -Edge Connected Components Computation	79
6.2.1	A Graph Partition-Based Framework	79
6.2.2	Connectivity-Aware Two-Way Partition	80
6.2.3	Connectivity-Aware Multiway Partition	85
6.2.4	The KECC Algorithm	88
6.3	Randomized $k$ -Edge Connected Components Computation	90
6.4	Edge Connectivity-Based Decomposition	92
6.4.1	A Bottom-Up Approach	93
6.4.2	A Top-Down Approach	95
6.4.3	A Divide-and-Conquer Approach	95
6.5	Further Readings	98
	<b>References</b>	99
	<b>Index</b>	105

# Chapter 1

## Introduction



With the rapid development of information technology such as social media, on-line communities, and mobile communications, huge volumes of digital data are accumulated with data entities involving complex relationships. These data are usually modelled as *graphs* in view of the simple yet strong expressive power of graph model; that is, entities are represented by vertices and relationships are represented by edges. Managing and extracting knowledge and insights from large graphs are highly demanded by many key applications [93], including public health, science, engineering, business, environment, and more. An availability of rich graph data not only brings great opportunities for realizing big values of data to serve key applications but also brings great challenges in computation. This book surveys recent technical developments on efficiently processing large sparse graphs, where real graphs are usually sparse graphs.

In this chapter, we firstly present in Section 1.1 the background information including graph terminologies, some example real graphs that serve the purpose of illustrating properties of real graphs as well as the purpose of empirically evaluating algorithms, and space-effective representation of large sparse graphs in main memory. Then, in Section 1.2 we briefly introduce the problem of cohesive subgraph computation and also discuss its applications.

## 1.1 Background

### 1.1.1 Graph Terminologies

In this book, we focus on *unweighted and undirected graphs* and consider only the interconnection structure (i.e., edges) among vertices of a graph, while ignoring possible attributes of vertices and edges. That is, we consider the simplest form of a graph that consists of a set of vertices and a set of edges.

We denote a graph by  $g$  or  $G$ . For a graph  $g$ , we let  $V(g)$  and  $E(g)$  denote the set of vertices and the set of edges of  $g$ , respectively, and we also represent  $g$  by  $(V(g), E(g))$ . We denote the edge between  $u$  and  $v$  by  $(u, v)$ , the set of neighbors of a vertex  $u$  in  $g$  by:

$$N_g(u) = \{v \in V(g) \mid (u, v) \in E(g)\},$$

and the degree of  $u$  in  $g$  by:

$$d_g(u) = |N_g(u)|.$$

We denote the minimum vertex degree, the average vertex degree, and the maximum vertex degree of  $g$  by  $d_{\min}(g)$ ,  $d_{\text{avg}}(g)$ , and  $d_{\max}(g)$ , respectively. Given a subset  $V_s$  of vertices of  $g$  (i.e.,  $V_s \subseteq V(g)$ ), we use  $g[V_s]$  to denote the subgraph of  $g$  induced by  $V_s$ ; that is:

$$g[V_s] = (V_s, \{(u, v) \in E(g) \mid u \in V_s, v \in V_s\}).$$

Given a subset of edges of  $g$ ,  $E_s \subseteq E(g)$ , we use  $g[E_s]$  to denote the subgraph of  $g$  induced by  $E_s$ ; that is:

$$g[E_s] = \left( \bigcup_{(u,v) \in E_s} \{u, v\}, E_s \right).$$

$g[V_s]$  is referred to as a vertex-induced subgraph of  $g$ , while  $g[E_s]$  is referred to as an edge-induced subgraph of  $g$ .

Across the book, we use the notation  $G$  either in definitions or to specifically denote the input graph that we are going to process, while using  $g$  to denote a general (sub)graph. For the input graph  $G$ , we abbreviate  $V(G)$  and  $E(G)$  as  $V$  and  $E$ , respectively; that is,  $G = (V, E)$ . We also omit the subscript  $G$  in other notations, e.g.,  $d(u)$  and  $N(u)$ . We denote the number of vertices and the number of undirected edges in  $G$  by  $n$  and  $m$ , respectively, which will be used for analyzing the time and space complexity of algorithms when taking  $G$  as the input graph. Without loss of generality, we assume that  $G$  is connected; that is, there is a path between every pair of vertices. We also assume that  $m \geq n$  for presentation simplicity; note that, for a connected graph  $G$ , it satisfies that  $m \geq n - 1$ .

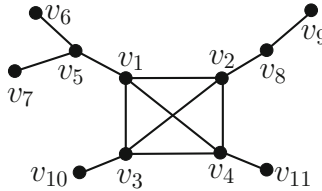


Fig. 1.1: An example unweighted undirected graph

*Example 1.1.* Figure 1.1 shows an example graph  $G$  consisting of 11 vertices and 13 undirected edges; that is,  $n = 11$  and  $m = 13$ . The set of neighbors of  $v_1$  is  $N(v_1) = \{v_2, v_3, v_4, v_5\}$ , and the degree of  $v_1$  is  $d(v_1) = |N(v_1)| = 4$ . The vertex-induced subgraph  $G[\{v_1, v_2, v_3, v_4\}]$  is a clique consisting of 4 vertices and 6 undirected edges.