Can we treat networks and graphs more the same, please? Also this is a survey for similarity techniques

Marissa Graham

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mathematics

Brigham Young University

ABSTRACT

Can we treat networks and graphs more the same, please? Also this is a survey for similarity techniques

Marissa Graham
Department of Mathematics, BYU
Master of Science

This is going to be the abstract.

Keywords:

Contents

List of Tables

List of Figures

0.1 Preface

We're going to take a second to motivate the problem a bit and describe what we're trying to do here. Kind of like section 1.2 in the red book, which is great and something that everybody should do always.

This is kind of like the motivation from the SRC presentation. Traditional graph theory can feel very pure mathy and useless, but networks are so important and such a new field that there aren't as many overview type resources, and I don't think the stuff I've seen does a very good job really putting everything in context, especially across fields. It's also not always clear what prerequisites you need to understand all the different things, so I want to make that easier, too.

Computer science and math mostly call things graphs, because they mostly work with them on a theoretical level. Either small ones for graph theory or big ones for modeling large networks. Throughout, I need to make sure to use the convention that a graph is the mathematical object and a network is a graph which models a real-world system. The definitions are clear enough, but the connotations make things confusing. It's not really separate topics and we're going to make an effort to treat them in a properly integrated manner.

Chapter 1. Background

We need to introduce all of the math that the different methods I want to talk about are going to use. Just quickly explain things and light context, so you don't have to google so much like when I was reading the fifty years paper. I'm going to cross reference things by what chapters use them so you know why it's important. We could move this stuff into the actual chapters, but I think it will be nice to have it all in one place so you know what you're getting into before reading.

This is also a good place to get everybody on the same page, terminology wise. How do we feel about a glossary? At minimum, a notation page would be great so I can just use G, A, etc. pretty freely throughout. It might not be too necessary, though.

1.1 Properties that all graphs have and also definitions

A **graph**[1] is formally defined as a finite, nonempty set V of **nodes** or **vertices**, combined with a set $E \subset V \times V$ of **edges** representing relationships between pairs of nodes. Throughout this work, we denote the number of vertices in a graph by n and the number of edges by m where not otherwise specified.

We normally deal with with **simple graphs**, which are those that do not have more than one edge between any pair of nodes (that is, a **multiedge**), and do not have any edges from a node to itself (a **self-edge** or **self-loop**). We should definitely still make sure that our math applies to generic graphs though and think about this, but most of the time it's simple. I think.

We also care about whether a graph is **directed** or **undirected**. In an undirected graph, we have an edge *between* two nodes, whereas in a directed graph we have edges *from* one node *to* another node. For simplicity, throughout this work we use the notation $v_i \leftrightarrow v_j$ for an undirected edge between nodes v_i and v_j, and $v_i \rightarrow v_j$ for a directed edge. We say two nodes are **adjacent** or **neighbors**when they are connected by an edge, and a node and an edge are **incident** when the edge is connected to the node. In the case of a directed graph, we call node v_i a **predecessor** of node v_j if there is an edge $i \rightarrow j$. In this case, v_j is a **successor** of v_i.

A graph can also be **weighted**, meaning each edge is assigned some real value w_{ij} representing the "strength" of the connection between nodes v_i and v_j. Generally these are positive.

In an undirected graph, the **degree** of a node is the number of nodes adjacent to it, or

[1]The term *network* is sometimes used interchangeably with the term *graph*. While they both refer to the same mathematical object, we attempt to follow the heuristic throughout of using the term *graph* to refer to a purely mathematical object and *network* to refer to a real-world system.

the sum of the weights of the incident edges in a weighted graph. In a directed graph, we distinguish between **in-degree** and **out-degree**, which are the number or total weight of a node's incoming and outgoing edges, respectively.

When studying real-world networks, we also make a distinction between **deterministic** and **random** networks, as discussed in (cite fifty years paper). This distinction is the same as that between a variable and a random variable A deterministic network is one in which the nodes and edges are "fixed"; that is, there is no other distinct graph which represents the same object and the nodes and edges do not change over time. In order to use graph theory to study real-world phenomena, we generally must use statistical inference methods and stuff and just treat it like a random variable instead. Some methods account for this and some don't and we should pay attention to this.

1.2 Fancier properties and special graphs

Trees. If you're gonna talk about generalized trees, which will probably come up, put it in the main chapter and make sure to mention it's about just picking a root, you can do it to any graph

Adjacency matrix

Laplacian

Maximum common subgraph and minimum common supergrpah

Isomorphisms and alignments

Kernels

Discussion of metrics with respect to graphs

Tanimoto index and Hamming distance between I'm still salty that the one survey paper didn't include

1.3 Basic algorithms and methods

Path finding and walks

Basic node similarity

Basic community detection

1.4 Other math background and definitions

Whatever spectral stuff we need to explain methods, without being too condescending

Edit distances?

Dynamical systems

Cost function (context of edit distances and alignment)

Linear and quadratic programming

Weight matching

Kernels

Chapter 2. Introduction

We're going to start off by talking about why graphs are useful, and the history of the study of them (briefly) and then the history of comparing graphs as a whole (in more detail, obviously).

Basically the whole point of this whole paper is that I want to emphasize the commonality between network and graph theory, even though they're treated fairly separately. I want to hammer that point home here by looking into the Wikipedia network for both terms, and same thing for my citation network once it gets more comprehensive. Colloquially, my impression is that "graph" is usually used for small numbers of nodes, and "network" is used for anything real-world based, whether it's small or not. Really, the important properties that make them different are how big they are and whether they're deterministic. It'd be really nice to have more words, because "graph" is most associated with "small and deterministic" and “network” is most associated with “large and non-deterministic”, but the definitions bleed into each other, and where does that leave “small and non-determistic” (aka bio, I think? maybe chemistry?) in terms of what you call it?

We also want to get into the difference between graph isomorphism, graph matching, network alignment, and network similarity, both in terms of actual definitions and how they’re commonly used. I should have already talked about the definitions in the the background chapter, at least a little bit, but I think it makes more sense to put that here.

Now we’re going to put in a few pages of the history of how they’ve been studied, and kind of the main categories of approach. I’m not completely sure yet how that’s going to break down, though, but it’ll be semi-based around partitioning the citation network, because that’s cool and gives you a really awesome visual. I’d like to do this by getting the “year” property built into my citation network, that should work really well and we can get some great visuals. Also, if we can, I want to get CrossRef to give me the field of study as well, so I can have at least an approximation of that for the all of the outcomponent papers.

2.0.1 Approach. Do a good github readme, basically, explaining how I went about this and making sure I’ve got all the papers from everybody, how the database works, general methodology.

Also I’d like to have a nice visual kind of like the one in the giant graph matching survey, so my reference section is friendlier. Here seems like a decent place to put it.

Chapter 3. Graph Matching and Isomorphism

It’s entirely possible that we’ll break down the chapters some other way, because I don’t want to just have a bunch of summaries of all this stuff, I want to keep it connected throughout.

Chapter 4. Network Alignment

Chapter 5. Graphlet Method and Similar

Chapter 6. Etc.? I don't know how these are going to break down yet

You're not going to have the black or red books in the paper list but you still need to have them in your bibliography, don't forget.

Bibliography

[1] Nobody Jr. My article please actually change when i do this, 2006.

[2] Alison Pease and Simon Colton. Computational Creativity Theory: Inspirations behind the FACE and the IDEA models. In *ICCC*, pages 72–77, 2011.

Index