EDGE IMPACT IN GRAPHS AND SOCIAL NETWORK MATRIX COMPLETION

By

Dennis Ross

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2016

ABSTRACT

EDGE IMPACT IN GRAPHS AND SOCIAL NETWORK MATRIX COMPLETION

By

Dennis Ross

Every graph G can be associated with many well-known invariant properties along with their corresponding values. Here, a framework is proposed to measure the change in any particular invariant upon addition of a new edge e in the resulting graph $G + e$. In graphs, the $\mathcal{P}$-*impact* of an edge e is the 'magnitude' of the difference between the values of the invariant $\mathcal{P}$ in graphs $G + e$ from G.

An edge is said to be of maximum $\mathcal{P}$-impact if it can be optimally added to G in order to achieve a desired result on either the invariant's value or graph's structure. New edges are added from the graph complement in simple graphs or between any pair of vertices otherwise. In this work several invariants are explored including: number of spanning trees, sum of distances between vertices, and vertex connectivity. A brief commentary on several other famous invariants is also provided.

A number of questions about the $\mathcal{P}$-impact of an edge on the structure of graphs are presented. Also included is an attempt to efficiently determine an optimal set of edges based on our invariant. Several restrictions and conjectures to determining this optimal set are discussed. A proof towards optimal edge addition for distance-impact in trees is given.

A natural application to measuring the impact of edge addition to a graph is that of link prediction problems. These applications are considered and an efficient algorithm for link prediction even with cold-start vertices using a subspace sharing method that decouples matrix completion and side information transduction is presented. This method is extended to predict ratings in user-item recommender systems where both may be cold-start. Mathematical guarantees and experimental results on real world publication and social networks are provided.

Végén egy útra.

ACKNOWLEDGMENTS

Before anything else can be said, I must offer my most heartfelt thank you to my advisor Dr. Abdol-Hossein Esfahanian. I have wanted to work with you from the day I finished your algorithmic graph theory course. I cannot thank you enough for taking a chance on me as your advisee, and it has truly been a privilege to earn my degree under your mentorship. I have achieved so many professional goals thanks to your unending support and leadership. I will miss the many cold Michigan afternoons consisting of drawing too many graphs on your white board, but I look forward to continuing our collaborations and friendship in the future.

I have been largely supported by working with USAID as part of the DSI. This would not have been possible without the tireless work of Dr. Pouyan Nejadhashemi. Thank you for taking the stress of maintaining funding off of my shoulders- not to mention exposing me to many interesting biosystems and global aid data problems.

I further want to extend thanks to the rest of my committee: Drs. Pang-Ning Tan and Guoliang Xing. Your guidance and insights made the process of completing a dissertation much more achievable.

This work would not be possible without the wonderful help of several colleagues. To Dr. Ronald Nussbaum, I greatly enjoyed the graph theory work we completed together. Also, a massive thank you to both Dr. Rana Forsati and Iman Barjasteh for sharing your interest and expertise in matrix completion problems (and enduring lots of silly questions). Finally, thank you to Ashley Depottey for making the DSI lots more fun than it otherwise would have been.

To my parents, Beth and Gary, thank you for encouraging intellectual curiosity throughout my entire life. That along with your love and support were instrumental in finishing this very challenging endeavor. Along with my sister, Valerie, I could always count on my family when I needed it.

To my friends, you all have stood by me both personally and professionally throughout my journey through graduate school. I would not have gotten through it without you all, nor would my time at Michigan State been as much fun. Thank you Ben Hardin, Sara Vredevoogd, Dr. Josh Vredevoogd, Dr. Andrew Ratkiewicz, Erich Owens, Dr. Luke Williams, Liz Wilson, Dr. Cheryl Jaeger Balm, Dr. Thomas Jaeger, and Dr. Tian Hao.

To my wife-to-be, Dr. Jennifer Cornacchione, you have the distinction of being just one place above Farkas in this section. You are also likely the only person to read every single word in this document, and your copy-editing was incredibly helpful. Thank you for being such a wonderful partner, sharing so many experiences with me, and of course for generously giving me all of the best chairs in New Leaf. Let's do some great things now that I am done too! I suppose I can also thank Gus and Oliver here- albeit begrudgingly...

Finally, I have to thank my most loyal companion, Farkas. Nagyon köszönöm. You were the only one with me every day, for every high and low, through both my masters and doctoral studies. Your fuzzy face always kept me going. Szeretlek kicsem!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Graphs provide a simple and powerful mathematical construct in which both abstract and real world models can leverage their structure to make compelling observations. The structure of a graph is then measured by examining its invariant properties. We are interested in maximizing or minimizing certain invariants, specifically distance-based measures, as a graph evolves under the addition of edges. Further applications will take a graph to be a social network. Using these graphs both link prediction and recommender system problems may be explored.

This work shows the progression of a pure graph theoretical problem into matrix completion techniques in networks and recommender systems. As such, in this chapter the important graph theoretical definitions and structures required for the introduction of the $\mathcal{P}$-impact process are provided. Within this framework we further give the necessary motivation for codifying this new concept while providing an overview of the scope of this work. Additionally, definitions and symbology for matrix completion, link prediction, and recommender system problems will be necessary to examine the evolution of the impact problem.

1.1 Definitions and Terminology

The graph theoretic terms and symbols of Bollabás in *Modern Graph Theory*, unless otherwise noted, are used throughout this dissertation. To begin, a *graph* is an unordered pair, $G = (V, E)$, with the set of *edges*, E or E_G, composed of pairs of elements in the set of *vertices*, V or V_G. The elements in each edge $e_{x,y}$, where $x, y \in V$, are called the *endpoints* of the edge. In a *weighted graph* each edge is also assigned a real-valued label. In an *unweighted* graph all edges are assigned a value of one. The *order* of the graph is the cardinality of the vertex set and the *size* is the cardinality of the edge set. These are denoted $n = |V|$ and $m = |E|$ respectively. $G + x$ is understood as $(V \cup x, E)$ or $(V, E \cup x)$ contextually when x is a vertex set or edge set.

A *subgraph* $H = (V', E')$ of a graph $G = (V, E)$ is an ordered set where $V' \subseteq V$ and $E' \subseteq E$. An *induced subgraph*, $G[V']$, is the restriction of G to a vertex subset V' where all edges with both endpoints in V' are included. Graphs G and H are *isomorphic* if there exists bijective correspondence between their vertex sets that preserves adjacency under the bijective map $\varphi : G \to H$. A *graph invariant* is a property that remains unchanged up to isomorphism between graphs.

An edge is called *incident* to a vertex if it contains that vertex as at least one of its endpoints. Two vertices are called *adjacent* if they share an edge, and two edges are *adjacent* if they share at least one endpoint. The *neighborhood* of a vertex is the set of all adjacent vertices. The number of edges incident to a vertex is the *degree* of that vertex. The *minimum degree* of G, $\delta(G)$, is the smallest degree in the graph. Similarly, the *maximum degree* of G, $\Delta(G)$, is the largest degree present in the graph.

If the set E contains repeated elements, G is called a *multigraph* and the repeated edges are called *multiple edges*. A *loop* is an edge where both endpoints are the same. A graph is called *simple* if it contains no loops or multiple edges. All graphs will be considered to be simple unless explicitly stated.

A *path* is a linear sequence of vertices where adjacent vertices in the sequence are adjacent in the graph. The *shortest path* is the path between two vertices x and y with the lowest total edge-weight sum connecting them. The length of this path is called the *distance* denoted $d(x, y)$ or $d_G(x, y)$. The *diameter* of a graph is the length of the longest shortest path. The *index* of an edge is the number of shortest paths containing it. The *dsitance index*, or *distIndex*, of an edge is the sum of the shortest paths containing multiplied by their respective lengths. The two endpoints of the path created by the diameter are called *peripheral vertices*. A *cycle* is the same as a path except the underlying sequence is cyclic. The sum of the distance of all of the pairs of vertices is the *all-pairs shortest path distance* or *total distance*.

A graph is called *connected* if there exists a path between all pairs of vertices and *disconnected* otherwise. A *component* is a set of vertices that are connected and there are no paths to vertices outside of this set. Any vertex v whose removal creates more components in $G - v$ is called a *cut vertex*.

A graph of order $n \geqslant 2$ and size $\binom{n}{2}$ is called the *complete graph* K_n. The *trivial graph* K_1 is a single vertex with no edges. The *complement* of $G = (V, E)$ is a graph $\overline{G} = (V, \overline{E(G)})$. A *tree* is a connected graph containing exactly $n - 1$ edges. A *spanning tree* is a connected subgraph of G with $n-1$ edges and n vertices. A vertex is a *pendant* if it is degree one, and it is a *leaf* if it is degree one in a tree. A *bipartite graph* is a graph where the vertices can be partitioned into two sets U and V such that the endpoints of all edges lie in exactly one of the partitions. A *complete bipartite graph*, $K_{m,n}$, is a bipartite graph where every vertex in U is adjacent to every vertex in V. A special case of the complete bipartite graph is the *star*, $K_{1,n}$.

Definition 1.1. *The $\mathcal{P}$-impact of an edge e is the difference between the values of the invariant $\mathcal{P}$ in graphs $G + e$ from G. With $\mathcal{E} = E(\overline{G})$ for simple graphs and $\mathcal{E} = E(K_n)$ otherwise.*

When the invariant is discrete-valued the definition of difference is understood contextually. The set $\mathcal{E}$ may be restricted to any multiset of the edges of K_n and the result is the set of *permissible edges*. If more than one edge is added to the graph the $\mathcal{P}$-impact is measured with the addition of the entire set of edges. The method of adding edges to a graph G using their $\mathcal{P}$-impact is the *$\mathcal{P}$-impact-process*. For simple graphs the $\mathcal{P}$-impact-process terminates when, after the addition of e, we have a complete graph. For a mulitgraph, this process can be infinite or terminate after a prescribed number of edges is reached.

If the invariant $\mathcal{P}$ is taken to be the sum of the all-pairs distance in G, then the $\mathcal{P}$-impact is called the *distance-impact* of e on G. When $\mathcal{P}$ is the number of spanning trees of G, this is denoted as *sptree-impact* of e on G. The discrete version of the distance-impact of an edge considers the number of pairs of vertices whose distance is reduced with the addition of an edge. This problem is called the *all-pairs shortest path distance count-impact* or *count-impact*. All of these processes and values can also similarly be considered for edge deletion or edge evaluation in $G - e$ and $(G + e) - e$ respectively.

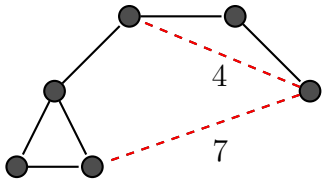

Figure 1.1 A graph where the dashed edges have their respective distance-impact noted.

Moving into the application space, the common descriptions for link-prediction and recommender system problems are described. Here the words *graph* and *network* will be used interchangeably with a preference for graph in the mathematical space and network in the application space. In general, the goal of these applications is to provide a list of edges to add to a graph or a set of items to recommend to a user. The notation describing the linear algebra structure is as follows. Lower case letters , such as u, are used to denote *scalars* and the bolded versions, such as $\mathbf{u}$, are instead *vectors*. Here $\mathbb{R}_+$ is used as the set of real non-negative numbers and $[n]$ to denote a set on integers $\{1, 2, \ldots, n\}$. For

matrices upper case letters, such as $\mathbf{M}$, are used. The *transpose* of vectors and a matrices are denoted by $\boldsymbol{m}^\top$ and $\mathbf{M}^\top$, respectively. The *scalar product*, or *dot product*, between two vectors $\boldsymbol{m}$ and $\boldsymbol{n}$ is denoted by $\boldsymbol{m}^\top\boldsymbol{n}$.

The *rank* of a matrix is the dimension of the vector space spanned by the columns of that matrix. The *Frobenius norm* of a matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$ is denoted by $\|\mathbf{M}\|_{\mathrm{F}}$ where $\|\mathbf{M}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}|\mathbf{M}_{ij}|^2}$. The *spectral norm* of $\mathbf{M}$ is $\|\mathbf{M}\|_2$ given by the square root of the maximum eigenvalue of $\mathbf{M}^*\mathbf{M}$. The *nuclear norm*, or *trace norm*, of a matrix is denoted by $\|\mathbf{M}\|_* = \mathrm{trace}\left(\sqrt{\mathbf{M}^\top\mathbf{M}}\right)$. Finally the *Moore-Penrose pseudo inverse* of matrix $\mathbf{M}$ is shown by $(\mathbf{M})^\dagger$.

For the applications, matrix completion problems are explored via collaborative filtering. For network completion there is a set of n users, $\mathcal{U} = \{u_1, \ldots, u_n\}$. As an extension, a set of m items, $\mathcal{I} = \{i_1, \ldots, i_m\}$ is added for recommender system applications. In the case with users and items each user, u_i, may (or may not) provide some form of feedback on subset of the item set. Examples of feedback can be varied, but in this work it will be restricted to an explicit real-valued rating.

Although the graph notation is consistent with the impact problems, some further definitions will be used to describe the graphical (network) structure of the applied problems. Given an order n graph $G = (V, E)$, the vertices are *distinguishable* as they represent distinct users. The *adjacency matrix*, $A \in \{0, 1\}^{n\times n}$, is such that every entry $e_{i,j} \in \{0, 1, ?\}$. An entry of "0" is a known missing edge, "1" is a known extant edge, and "?" is an edge that is unknown. There is further an induced subgraph of G called $O = (V, E')$ where $O \in \{0, 1, ?\}^{m\times m}$ with $1 \leqslant m \leqslant n$ and $E' \subset E$.

When a row or column is completely unobserved, the corresponding node is considered a *cold-start node*. There are no assumptions as to the distribution of observed nor unobserved entries in A. However, in O, the edges are sampled uniformally at random and do not contain any cold-start entries. This is enforced even when such a restriction dictates that $m \ll n$.

The objective of this work to fill out the missing entries of a matrix using the observed entries and possible other information. The process of filling in the missing entries is called *matrix completion*. Any external information about the nodes in a social network graph beyond the adjacency matrix is called *side information*. Side information can be widely varied, depending on the problem domain, and takes on many forms (e.g. human demographic information, URL click patterns, protein folds in yeast). The process of applying the side information to complete the adjacency matrix (or ratings matrix) is called *transduction*.

1.2 Motivation and Goals

Graph invariants describe the nature of graphs and are at the core of graph theoretical knowledge. Further, it is useful to examine the evolution of graphs over time via edge addition and deletion. These facts motivated us to create a process in which the changes in invariant values of graphs can be tracked or controlled via the $\mathcal{P}$-impact process. By adding edges to an empty graph and examining its evolution, definite patterns emerged. This led to attempts to generalize and categorize the entire process. According to an extensive literature search, there is no previous work resembling the $\mathcal{P}$-impact or $\mathcal{P}$-impact process.

The polynomial nature of the complexity of computation of many invariants is also leveraged as testing the $\mathcal{P}$-impact process can be fully computationally completed and compared against known benchmarks. This computational advantage is imperative because the determination of the optimal $\mathcal{P}$-impact process is generally not trivial– even on basic graph classes.

By creating a new definition and process there are several clear objectives to this work. Primarily, the goal is understand as much of the fundamental mathematical nature of the $\mathcal{P}$-impact process as possible. To this end, the goal is to determine necessary and sufficient conditions for efficient discovery of optimal $\mathcal{P}$-impact edges in special graph classes

before considering the general case. This work settles many cases in trees and provides a starting point for using graphical information to determine side information to be used in link prediction and recommender system problems.

There are many results of these types in social media networks. Initially, distance-impact was deployed as a way to bridge distinct groups of people for more effective messaging. Messages tend to stay close to their sources in social networks [4]. Policy makers or advertisers could pay to connect seemingly disparate users by a high impact edge so then could release coordinated messages. This effectively reduces the distance that a message would need to travel and creates an additional source for newly reduced-distance users to receive the message.

These applications, from a pure graph theoretical concept to an initial application-driven approach, led to an exploration of using side information to predict edges in partially observed networks. This exploration pushed this work into the realm of matrix completion problems in the realm of link prediction, recommender systems, and, finally, in item-based taxonomy problems.

Real world networks, social or otherwise, are often modeled by graphs. However, in real world networks it can be very difficult to gather the complete graph structure that models many applications (e.g. social networks, biological networks, internet website interactions). Such issues arise from the nature of these problems. Information may be protected as a matter privacy, corporate knowledge, or may not be queried due to the sheer size of the data. Further, actors in networks may try to deliberately obscure themselves, or may have few measurable interactions. Although not all of these problems may be alleviated, there are several mathematical techniques that have been successfully deployed to complete the missing information from such partially observed graphs. These techniques initially collect a partial sample of a network and then infer the networks structure. This, admittedly broad, technique is referred to as *network completion* or less frequently as *survey sampling*.

To improve on network completion and recommender system problems via matrix completion side information is collected and analyzed. The work herein is used to solve such problems in their most difficult case, namely when cold-start issues occur. Currently, this is the least understood area of matrix completion applications to graph completion and recommender systems.

1.3 Related Work

Although the concept of the edge impact process is new, graph invariants have been widely studied using a variety of mathematical techniques. Even more similarly, many results in extremal graph theory define a minimum addition of edges to achieve a desired invariant value. Simple graph classes are examined to gain insight and follow extremal work that was done using constructive and probabilistic tools.

To understand the $\mathcal{P}$-impact in general graphs, first we consider particular graph classes. Similar work was done by Ross et al. in the distance-preserving problem for regular graphs [77]. Work in random graphs generation to make general observations was given by Szemerédi's Regularity Lemma [43]. This allows claims of randomness when the $\mathcal{P}$-impact process is used on large graphs.

Erdős published many results on extremal graph theory [22] and specifically graphs containing certain structures [21]. These extremal problems try to define the smallest number of edges or vertices that must be used for certain properties to be satisfied. The $\mathcal{P}$-impact process examines similar phenomena, yet restricts to the addition of edges to preserve or achieve invariant values. Strictly working with the number of edges, other extremal results also exist [15].

Some major invariants are understood through similar methods as the $\mathcal{P}$-impact process. Edge-connectivity can have a relationship with both distance and spanning tree applications. The problem itself is quite mature and many computational techniques for computing edge-connectivity exist [54]. Algorithms and improved bounds for edge-connectivity

were also found by Esfahanian and Hakimi [23]. This is in addition to other bounds, computational results [56, 25] and extremal results on the number of edges that can be added to enforce k-edge-connectivity [13].

Although this work focuses more on the role edges play in effecting distances, often the vertices of concern are highly central to the graph. Previous work has explored the discovery and properties of central vertices. Nieminen examined and classified the centrality of vertices in graphs [62]. Later work by Borgatti examined centrality's role in social networks [10]. These are examined in the practical applications as the impact of important actors in social media graphs are explored.

The maximum distance-impact edges try to reduce either total distance or the distances of the largest number of shortest-paths in a graph. This process is similar to other ideas that aim to create structurally important edges by measuring betweenness and centrality. Freeman gave a framework for measuring betweenness centrality, and Brandes gave a fast algorithm for computational finding it [26, 11]. This also leverages similar work exploring the centers and centroids of graphs [63].

The applications contained herein use the modeling power of graphs for physical and social networks. There has been much work in these fields that can guide these findings. Several authors have found methods of measuring impact of messages in social media [85, 37]. Others have explored the predictive powers of examing the social networks for predicting: user activity [86], influenza propagation [18], and box-office revenue [3]. Bakshy et al. also put forth a method for estimating the popularity of content by measuring distance that URLs traveled through a network [4].

Graph (network) completion problems exist in a number of different settings where incomplete graph information is present. The literature is deep with methods to solve such instances with applications in information retrieval, social network analysis, and computational biology [2, 66]. Some relevant examples involve large social networks such as Facebook and Twitter. Because these networks have users that number in the billions,

to infer the full network topology describing the relationships between users (edges in the network) is known to be difficult in many cases [66].

Extant learning algorithms for network completion problems commonly make structural assumptions on the nature of the underlying network. From here, there are many methods to efficiently reconstruct the actual network. The classical network completion implementations make an assumption that random entries of the adjacency matrix are missing. However, in network completion is accomplished by randomly subsampling the partially observed network. Specifically, in [41] it is assumed that the underlying network follows the Kronecker graph model. An expectation maximization (EM) framework was used to infer the unobserved pieces of the network.

A computational method that was used to find missing and spurious interactions within complex network is given in [32]. Here these interactions were found using stochastic block models which then captured the structural features of the network itself. The same method was applied on the interactions of proteins in a yeast network. Further, a sampling method that derives confidence intervals from sampled networks is given in [36].

A similar link prediction application that applies most similarly to the $\mathcal{P}$-impact process is given in [48]. This work predicts future edges that will be added to a network. However, because no links can be observed for unsampled nodes, these statistical models perform poorly in these cases. Such problems exist when extreme sparsity is present, and fare even worse when presented with the cold-start scenario.

Maximum margin matrix factorization [83] was a method developed for collaborative filtering. Several works show that, theoretically, this method may perfectly complete a matrix [90, 58, 72]. Some extensions of this work include collaborative filtering [83] and also allow the use of side information [1].

One principle method explored in the work is the use of transduction of side information to complete matrices and recent work approaches matrix completion using a similar

tack. Some work considers matrix completion with transduction [24, 65] and even can be expanded to the case where side information is of infinite dimension as described in [1].

To handle sparsity problems, several studies have given matrix factorization models that try to optimize their results by taking fewer samples from the original matrices. Menon et al. gave a logistic regression approach that added a logistic regression with a principled confidence-weighting scheme to its objective function [58]. A Bayesian approach was taken by Porteous et al. wherein regression is applied to the side information themselves [72]. One advantage to this Bayesian approach is that the mixture model for the prior of the users/items provides a different regularization for each of the latent classes. Park et al. examined recommenders as simple regression problems [67]. Here they used a combination of both user and item metadata to create the side information.

A relaxation of the network completion problem can be seen as link prediction on bipartite graphs with weights. These recommender sytems problems have also been widely studied with many similar techniques to those used in link-prediction. There has also been some work directly addressing cold-start problems.

Content-based filtering (CB) and collaborative filtering (CF) are well-known examples of recommendation approaches. As demonstrated by its performance in the KDD Cup [19] and Netflix competition [8], the most successful recommendation technique used is collaborative filtering. This technique exploits the users' opinions (e.g., movie ratings) and/or purchasing (e.g., watching, reading) history in order to extract a set of interesting items for each user. In factorization based CF methods, both users and items are mapped into a latent feature space based on observed ratings that are later used to make predictions.

Some methods are applied that try to predict item ratings using global, and easy to calculate, item characteristics. Such algorithms have been based on the popularity of the items [67] or even through random selection [51]. By approaching the cold-start users/items in this global manner, any nuance between distinct groups of users is lost. With this lack of filtering, a great amount of accuracy in the recommendations is lost.

To alleviate the concerns raised by the lack of historical data for users/items, methods were developed present users with a set of items that they must rate. These warm-start methods may also import user preferences from side information. Such work has been sucessful [51, 89, 17, 87], but as these methods explicitly force a new user to provide ratings for k representative items (or a new item being forced on k users) there is significant drawbacks in using warm-start methods. In commerce, the views spend learning item characteristics could have better been spent on presenting items a user may want to have purchased. For social networks, the site may be uninteresting to a new user because they are not presented immediately with relevant or entertaining interactions.

To avoid the warm-start pitfalls, there has been a large amount of interest in using side information to complete the rating or adjacency matrices [82]. This side information can even provide context for cold-start users/items without requiring direct user feedback. These methods of feature combination combine the features of the users and items to increase the model's accuracy. The determination of these features is found using available user information (e.g. profile, location, web history) and item information (e.g. metadata, manufacturer specifications, web traffic).

Because user-user (or user-item) feature spaces can be described using the features extracted from the side information, there has been a push to develop methods that exploit the overlapping subspaces therein. These methods are often called matrix co-factorization and have been used to sucessfully exploit rich sources of side information to increase model accuracy. In [52, 31, 35, 34] rating and side information matrices are decomposed at the same time to expose shared latent features. The work of [64] imputed missing values and used these in the matrix factorization to boost the performance on the cold-start problem.

A kernelized matrix factorization was given by Zhou et al. [90]. Here auxiliary information is incorporated into matrix factorization to assess the similarity of the latent features. Saveski et al. [79] developed a matrix factorization method that collectively decomposed rating and side information matrices within a common space of low dimension.

By mapping the features from the side information into the latent features, another group of feature combination methods have been found. Elbadrawy et al.'s approach is made to learn a function that transforms the feature vectors of items into their latent space [20]. Gantner et al. further gave a matrix factorization model that maps the features directly to latent features [28]. Finally, Boltzmann machines [33] and aspect models [80] also utilize side information to be used explicitly for cold-start recommendation problems.

Some other models compute similarity between users/item explicitly and use these similarity matrices to apply feature combination methods [49, 84]. Specifically, Trevisiol et al. studied users' consumption of news articles and then built a special graph dubbed BrowsGraph [84]. BrowsGraph included both structural and temporal properties that both served news to its users but was also able to provide relevant articles to even cold-start users.

Another common approach is to allow many different recommender algorithms to execute on the data and combine their results using various methods. These approaches require building and running several different models leading to necessarily higher computational overhead [33]. However, some success has been found in the combination of these models due to weighted outputs [16], reapplying a rank function [12], applying different recommenders at each phase [9], and creating a voting framework [70].

One of the more effective approaches for recommender systems is to efficiently factor the adjacency or rating matrices into multiplicative of k-rank matrices. This approach heavily influenced this dissertation. Considering the item recommendation case, the general idea is to factor the rating matrix into user and item specific matrices such that their product approximates the complete rating matrix. The two major approaches to this end are optimization techniques [76, 50, 53, 46] and probabilistic [61] in nature.

As another form of side information, over the last five years several taxonomic approaches to recommender systems have been attempted. These methods attempt to extract implicit user preferences based on the hierarchy of the items themselves. The first

taxonomic matrix factorization approaches tried to grab these explicit taxonomies and use them as side information. As the field evolved, sophisticated methods to learn the taxonomies themselves from the item-user interactions were developed.

Zhou et al., using their Kernelized Probabalistic Matrix Factorization (KPMF) algorithm, have leveraged side information within the item-taxonomic framework [91]. In their work, they accurately make item recommendations using the users' social network as side information. Dror et al. created a matrix factorization scheme wherein the categorical and temporal information about music was used as side information for music rating predictions [42]. Kanagal et al. developed a taxonomy-aware dynamic latent feature model that not only allowed such taxomonies to be used in finding items to recommend to users but also provided a framework for finding similar latent features between parent and sibling items in the taxonomy [40]. Zhang et al. automated the taxonomy creation process via learning techniques that created taxonomies with no user input [88]. Their algorithm even performs better than latent factor models on human-created taxonomies.

Finally, we note that although various hybrid methods such as factorization machines [74], content-boosted collaborative filtering [57], probabilistic models [71], pairwise kernel methods [7], and filterbots-based methods [68] have been developed to blend collaborative filtering with side information, they are specifically designed to address the data sparsity problem and fail to cope with cold-start users or items problem, which is the main focus of this work.

1.4 Overview

In this introduction, we defined a new concept called the $\mathcal{P}$-impact process in addition to our motivation for introducing this concept. In Chapter 2 we describe a detailed problem statement while raising open questions about $\mathcal{P}$-impact edges and describing implications in network completion, link prediction, and recommender system problems. In Chapter 3 the $\mathcal{P}$-impact process is explored. In Chapter 4 distance and count impact are

compared in empty graphs. While in Chapter 5 $\mathcal{P}$-impact is applied to trees. Chapter 6 serves as a bridge between $\mathcal{P}$-problems and applications in social networks. Chapter 7 examines network completion in graphs. Chapter 8 extends to recommender systems and Chapter 9 contains the conclusions of this work.

Chapter 2

Problem Statement

The goal of this work is optimize invariant values under edge addition and to use such techniques to provide predictions for new edge additions in real world applications. Because of this, the scope of this work focuses on two general facets. First, the mathematical basis for the $\mathcal{P}$-impact problem is described. Second, the applications to link prediction and recommender systems are given.

For a general invariant, our meaning of $\mathcal{P}$-impact optimization may not be clear. Optimality as a measure of goodness where each added edge will try to optimally preserve an invariant value, structure, or concept. As input a graph G, a target size k, are considered and a graph H of size k with the desired invariant value is created. Given the wide and varied nature of graph invariants, this may be understood to hold as each edge is added, or only after the entire set of edges is added. There may not be an optimal set of edges for a given G and k under a particular invariant; the prescribed value is approached as closely as possible with the addition of k edges.

To begin this exploration the three invariants are considered. The first counts the number of spanning trees and the other two are concerned with distance variants. For the number of spanning trees, the goal is to maximize the total number of spanning trees via a fixed number of edge additions. For the distance invariants, a goal is set to determine both

the set of edges that reduce the sum of all distances and the highest number of vertices with reduced distances, respectively.

To discover the optimal set of edges to add in the case of a general graph, first each of these invariants are considered under the $\mathcal{P}$-impact process with G restricted to specific graph classes. An empty graph is considered with $k = \binom{n}{2}$. The process then adds edges throughout its $\mathcal{P}$-impact process to eventually reach K_n in the simple case. Here, an attempt probabilistically identify what graphs are created through this evolutionary process with our chosen invariants is made. A further open problem is to introduce a function to randomly add edges of highest and lowest impact in an attempt to create graphs with interesting properties in a process similar to that which creates small-world networks.

The highest consideration is made for when G as a tree on n nodes with $k = n$. A tree is chosen because it is a minimally connected graph and our invariants rely on connectedness for reasonable results. In this case the spanning tree impact problem is trivial and can be determined by only finding the diameter of the graph. However, the count and distance impact edge sets are provably non-trivial and their determination is to be explored with some results presented.

When dealing with trees in the $\mathcal{P}$-impact problem, connectivity is heavily leveraged. However, questions rise as to how to handle vertices that are disconnected from the graph. Once connectivity was dropped vertices become cold-start (isolated). This lead to an exploration of cold-start vertices and also determination of edges to be added to a graph. From these link prediction exercises a practical application arose that considered matrix completion techniques for adjacency matrices using graph invariants as side information. From here the problems became more concrete and were explored on a variety of real world datasets.

The initial aim of the work on network completion (link prediction) applications was to exploit side information from the nodes. This side information is generally available in some form from the social networks at hand. A problem thus arises wherein the choice of

side information and methods of transduction deeply affect the quality of the link prediction. Variations are also included where there are cold-start nodes and differing choices for side information.

Similar to link prediction problems, this work also explores the challenges of predicting ratings in recommender systems. Again, efficient matrix factorization methods exist, but are generally intolerant to cold-start problems. A further challenge is explored in recommender systems where cold-start users *and* items may be present. With sparsity and cold-start issues present, this work seeks to provide a framework for incorporating side information using shared subspaces in the matrix completion techniques to provide accurate rating prediction.

The results of this lay the groundwork for future exploration of $\mathcal{P}$-impact edges. It provides many insights about the difficulty of assessing distance-impact in graphs and shows the non-trivial nature of distance-impact edges in trees with a collection of counterexamples to various conjectures. The mathematical basis culminates with a proof that the maximum distance-impact edge cannot be incident to two leaves in a tree. Considerable progress was made when considering the application-based work in both network completion and recommender systems. Both experimental and mathematical results are given showing a decoupled transduction approach to incorporating side information in matrix completion is efficient and accurate. The algorithms presented also handle cold-start problems and non-randomly distributed sparse data sources.

Chapter 3

The $\mathcal{P}$-Impact Process

The $\mathcal{P}$-impact process refers to an algorithm that adds edges from the permissible set to a graph while attempting to maintain some invariant value until a prescribed graph size is achieved. The general $\mathcal{P}$-impact process algorithm is described here as well as a randomized variant. Although running times depend on the complexity of computing the invariants themselves, computational improvements through meaningful reductions in the permissible set are explored. These reductions are introduced here and further detailed in later chapters. Generally, maximum or minimum $\mathcal{P}$-impact edges are considered, but any impact value of an edge may be used. Lastly, the underlying graphs may control much of the $\mathcal{P}$-impact process, and these are also discussed in later chapters.

3.1 The Basics

The $\mathcal{P}$-impact process is most simply imagined as a brute force algorithm. Generally, the concern is with single edge addition that at the time satisfies some invariant condition. Here, once an edge is added it may not be removed. The algorithm itself is simple in that it adds each edge from the permissible set one at a time, computes the invariant value, and removes the edge. From this, it then takes the edge with the desired invariant value.

The maximum $\mathcal{P}$-impact edge is usually the one most used, but the max function may be replaced with any condition as required by the invariant.

Algorithm 1 The $\mathcal{P}$-Impact Process

```
procedure FindImpactEdge
    I={}
    for e ∈ P do
        I ∪ (P(G + e), e)
    return max_Condition(P(G + e), e)
procedure FindImpactfulSet
    while |E(G)| < k do
        e = FindImpactEdge(G)
        P = P\e
        G = G + e
    return E(G)
```

This is the case that we generally pursue. However, there is also interest in finding an 'optimal' set of edges to add. Notice that the process generally picks the edge that is best in any particular time, but does not consider its impact on future edges. This greedy approach may not always choose the best edge set for a given invariant as a whole. A relaxation of this addition will try to find the optimal set of edges to add all at once. For this formulation of the problem we attempt to determine a single set of edges from the permissible set that, when added, maintain or achieve an invariant value. These results may be better, but they rely on a higher burden of calculation that will be discussed next.

3.1.1 Time Complexity

Notice that the $\mathcal{P}$-impact process must compute the invariant value after the addition of each edge of the permissible set, and then repeat this process for every edge addition until G is size k. This cost is fixed by the number of times that the best known algorithm to compute each invariant is called. If an invariant $\mathcal{P}(G)$ can be computed in $f(n)$ time this call must be made for each possible edge in the permissible set, which is $O(n^2)$, and $k-n$ times to choose all of the required edges. Thus, the algorithm will take $O((k-n)\cdot n^2 \cdot f(n))$

time to compute. Throughout this dissertation, this model will be used unless explicitly stated otherwise.

Compare this to the relaxed case, and it is clear that the invariant calculations are fixed but there is a need to consider all size $k-n$ subsets of the permissible set. As the permissible set can be up to $O(n^2)$ in the simple case this is $O\left(\binom{n^2}{k-n} \cdot f(n)\right)$. This considers all possible *subsets* of size $k-n$ of $\mathcal{P}$. Looking for an optimal subset of a *fixed size* will generally be more time-consuming than the previous case, but is still polynomial in n if $f(n)$ is.

In Figure 3.1 the maximum distance-impact process on a random graph G with $n = 10$, $m = 14$, and $k = 17$ is outperformed by finding an optimal set of three edges to add. The added edges are shown in red dashed lines, and the order that the edges are added are given.

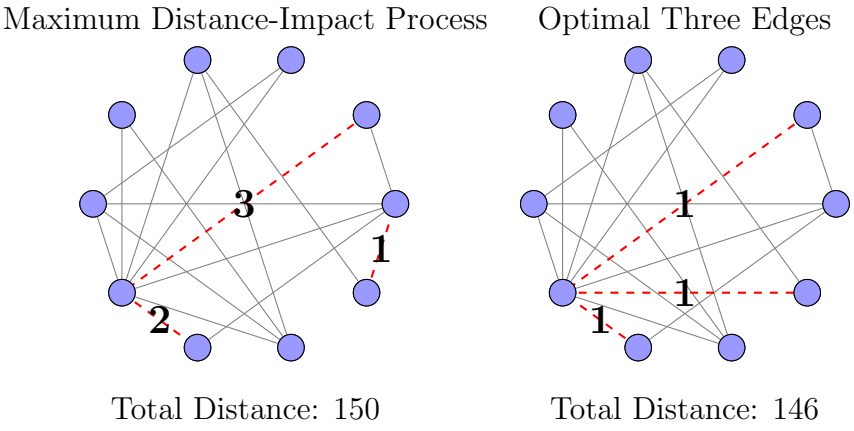

Figure 3.1 Maximum distance-impact does not yield an optimal solution

3.1.1.1 Running Time of SPtree-Impact

The sptree-impact of a graph G can be computed by calculating the number of spanning trees under addition of each edge in the permissible set, respectively. On a given graph the number of spanning trees is computed in $O(V+E+E \cdot T)$ where T is the number

of spanning trees [27]. This process can be repeated $k-n$ times to find the desired graph in $O((k-n)\cdot V + E + E\cdot T))$.

If the best overall set is taken, as in the relaxed case, there is a need to check the number of spanning trees under the addition of all possible subsets of size $k-n$. This would take $O\left(\binom{n^2}{k-n}\cdot(V+E+E\cdot T)\right)$ time.

3.1.1.2 Running Time of Distance-Impact

To find an edge of particular count or distance-impact, each possible edge in the permissible set must be checked and the all-pairs shortest paths after its addition must be found. Using the FloydWarshall Algorithm, or several calls to Dijktra's Algorithm, we can compute all of the distances in a graph in $O(n^3)$. For the distance-impact the sum of the total distance in each step is found, and for the count-impact the sum of the total number of changes is found. Combining these, finding any one edge may be done in $O(n^5)$. This is the special case when $k = n+1$. In general, there is a need for this process to be repeated $k-n$ times and that gives a running time of $O((k-n)\cdot n^5)$.

Again for the relaxed case, all of the distances under the addition of all possible subsets of size $k-n$ must be checked. This would take $O\left(\binom{n^2}{k-n}\cdot n^3\right)$ time.

3.2 Random $\mathcal{P}$-Impact Process

What happens if instead of adding the maximum distance-impact edge at each step, a biased coin is flipped and then the decision to add the maximum or minimum-impact edge is made? This is the question that may be solved by the random distance-impact process.

3.2.1 Random Distance-Impact Algorithm

This process is quite straightforward. First, two properties are chosen along with a probability threshold. Second, use the $\mathcal{P}$-impact process as before, but choose the edge to add based on which condition is randomly selected. Condition one is considered to be a maximum impact edge and condition two to be a minimum impact edge under both distance and sptree invariant. Here, consider two conditions and a threshold, $0 \leqslant t \leqslant 1$, for deciding between them.

Algorithm 2 The Random $\mathcal{P}$-Impact Process

```
procedure FindImpactEdge
    I={}
    Choose random p ∈ [0, 1]
    for e ∈ P do
        I ∪ (P(G + e), e)
    if p ⩽ t then
        return max_Condition 1(P(G + e), e)
    else
        return max_Condition 2(P(G + e), e)
procedure FindImpactfulSet
    while |E(G)| < k do
        e = FindImpactEdge(G)
        P = P\e
        G = G + e
    return E(G)
```

Note that there is no reason to restrict to only two conditions. The minimum and maximum and/or distance and count $\mathcal{P}$-impact are often paired as our conditions, but arbitrarily many conditions could be imposed.

3.3 Improvements

The costs of calculating invariants is considered to be fixed, and the time complexity required to compute invariant values is given to the best algorithms available at the time. Because of this, any improvements to running time must be achieved by reducing the size

of the permissible set meaningfully. If such improvements can be achieved, fewer edges must be examined and, therefore, fewer calls to calculate the invariant values are made. By determining which edges in the permissible set may be safely removed, the practical running time can be reduced even in some cases where the worst-case time complexity is unchanged. Most conjectures as to permissible set reductions are given in trees, but other graph classes could be explored.

3.4 Conclusion

The $\mathcal{P}$-impact process is a method for determining which edges to add to a graph until a desired size is achieved. The algorithm for both the standard and random process is described here along with running time observations. Provided an invariant can be found in polynomial time, this process can be completed in polynomial time as well. The focus on the remaining $\mathcal{P}$-impact chapters will revolve around finding fast ways to reduce the permissible set to improve the computational complexity of the process.

Chapter 4

Construction from Empty Graphs

In this chapter an empty graph is considered and a random impact process is used to add in new edges. The minimal structure present in trees can be used to make several observations, however, in this chapter there is no initial structure. Unless otherwise stated, edges of the maximum impact for our $\mathcal{P}$-impact process are considered at each step. However, one variant of the random $\mathcal{P}$-impact process will alternate between minimum and maximum impact edges. The evolution of maximum distance and count-impact is also described when the initial graph is empty.

4.1 Observations

Many invariants are not necessarily well defined on disconnected graphs. For one, there are no spanning trees of a disconnected graph because the spanning tree itself must be connected and contain all vertices. Conversely, the distance invariants can be defined on disconnected graphs. If two vertices, x and y, have no path connecting them then it is the case that $d(x, y) = \infty$. For the purposes of this work, with the same vertices, consider $d(x, y) = C$. Here C will be an arbitrarily large constant. With this real-valued finite distance, effectively a large penalty is set up for leaving a graph disconnected.

In the distance-impact case this restriction will force the first $n+1$ edges added to create a star, but in the count-impact version any tree will be initially created.

4.2 Distance Invariants

4.2.1 Count-Impact

The count-impact process from an empty graph moves through several fully classified phases. In the first phase it is important to restate that disconnected vertices in a graph are of arbitrarily large distance apart. This represents the infinite distance that they have from each other and forces the first $n-1$ count-impact to create a tree. This is because whenever at least one vertex is disconnected from the others, the maximum count-impact edge must connect it to the rest of the graph because the distance reduction is 'near-infinite'. This implies that isolated vertices will be connected to each other or connected components, but unlike the maximum distance-impact edge, which will be described later, these connections do not have to occur on the highest degree vertices.

In the first phase a truly random tree is constructed. Once this random tree is constructed, maximum count-impact edges mostly tend to fall into two general archetypes. In phase two, new edges will generally be connected to peripheral vertices or to vertices of maximum degree. Similar to the results on trees, there are counterexamples, but computational testing has shown that approximately 82% of the edges in the second phase will be of one of these classes. This percentage was computed from all graphs up to order eleven, but is not a claim for all graphs. Further, note that phase two will be skipped when the random tree created in phase one produces a star. A conjecture here is that the length of the diameter in the random tree will control how many edges will be added in phase two.

Once a graph is diameter two, phase three begins. No vertices, other than the newly connected ones, will benefit from the additional edge. Thus, the count-impact of any edge is again one. A graphical description of the phases is given in Figure 4.1.

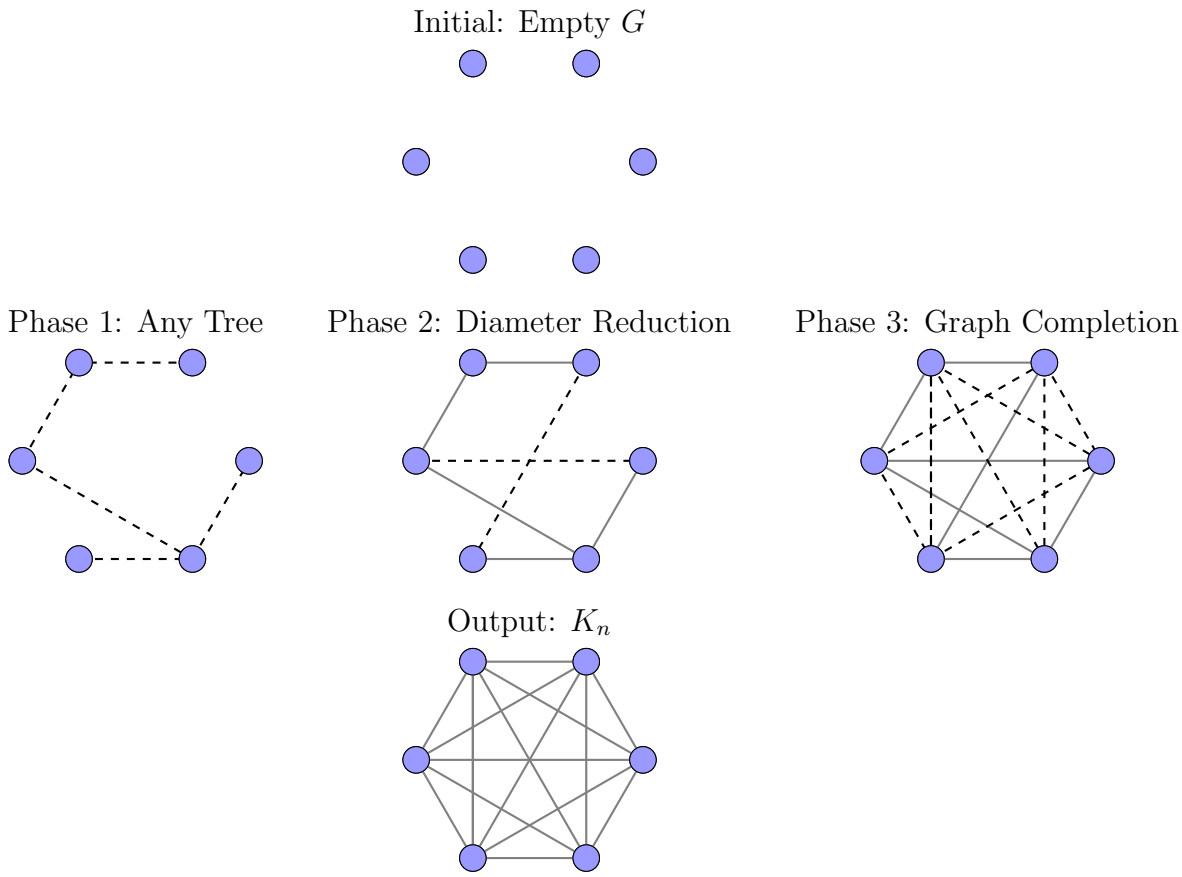

Figure 4.1 The Phases of the Maximum Count-Impact Process

Proposition 4.2.1. *After the addition of the first $n-1$ maximum count-impact edges to an empty graph of order n a tree is created from the collection of all possible trees on n vertices.*

Proof. Consider the set of disconnected vertices to be $\mathcal{D}$ of order d, $0 \leqslant d \leqslant n$. The set of vertices that make up the connected component are $\mathcal{C}$ of order c, $1 \leqslant c \leqslant n$. Because the unlabeled graph is initially empty, the first edge connects two arbitrary vertices. There is no choice to be made, however, once the initial edge is added, it must be shown that all forthcoming distance-impact edges will create a tree. If there are only two vertices the process is complete and a star $K_{1,1}$ is created. Count-impact is fairly simple because it only measures the number of vertices that have reduced. New edges can be added in one of three cases.

Case 1: The edge is incident to two vertices in $\mathcal{C}$. In this case the count-impact can be at most c because no vertices in $\mathcal{D}$ are added to the connected component.

Case 2: The edge is incident to two vertices in $\mathcal{D}$. This has a count-impact of exactly two because both vertices were in the disconnected set so only a K_2 component is created.

Case 3: The edge e_{vu} is added where $v \in \mathcal{D}$ and $u \in \mathcal{C}$. The count-impact is $c+1$ because all vertices in $w \in \mathcal{C}$ have reduced their distance from f to $d(w,u)+1$. This is a reduction of the distances of $c+1$ vertices.

Clearly case 3 is the optimal case because it has a count-impact of $c+1$ while case two is at most c and case 1 is exactly two. Further, the edges from $\mathcal{D}$ to $\mathcal{C}$ can be added at random because the count-impact does not take into account the total distance. Thus, a random tree of order n is created after the addition of $n-1$ count-impact edges. □

4.2.2 Distance-Impact

The maximum distance-impact process initially appears as though it must move through the same phases as the maximum count-impact process. However, significant differences in phase one lead to the elimination of the diameter reduction phase from the count-impact variant. Specifically, the distance-impact edge will always form a star after tree-many edges are added. New edges will still try to minimize the total distance and must be added from an unconnected vertex to the center of the largest component. Because of this, the final phase remains the same and will add edges at random to the graph because its diameter is two. This implies that all possible edges have distance-impact of one. This process is described in Figure 4.2.

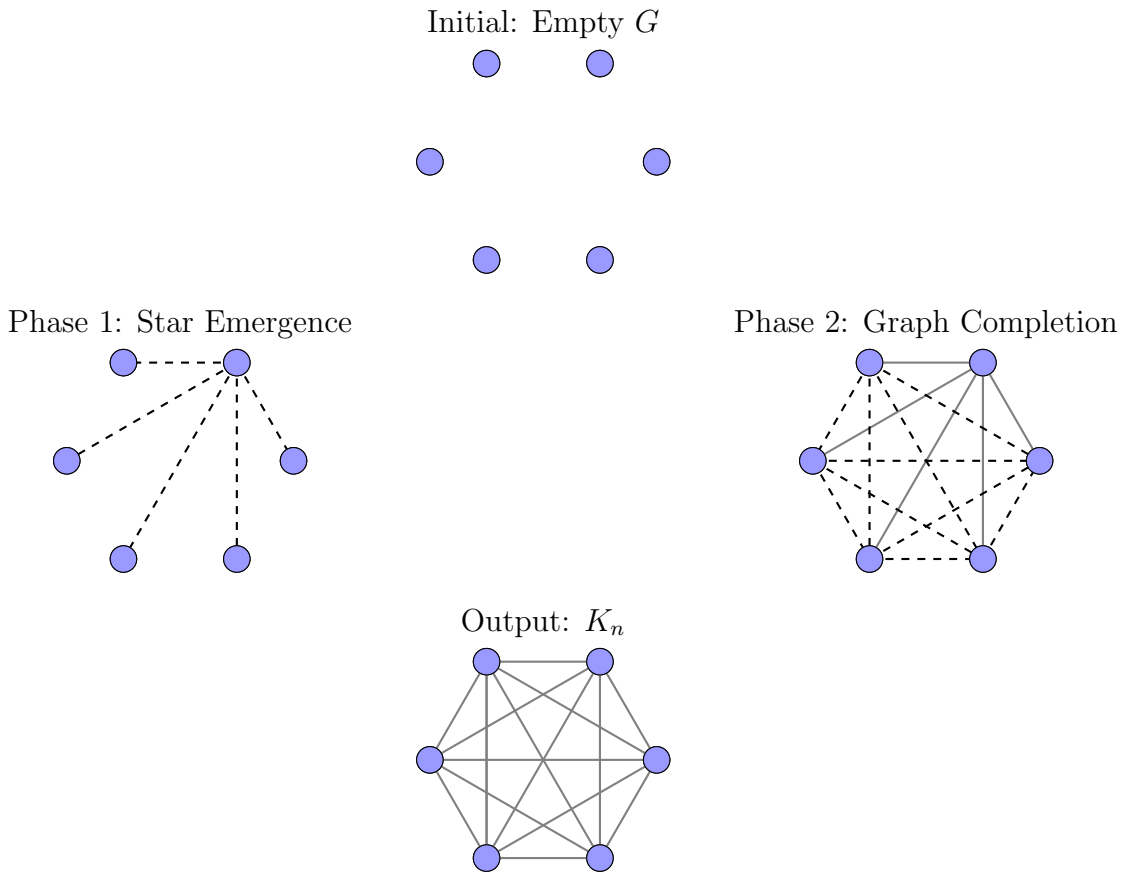

Figure 4.2 The Phases of the Maximum Distance-Impact Process

Proposition 4.2.2. *After the addition of the first $n-1$ maximum distance-impact edges to an empty graph of order n the graph $K_{1,n-1}$ is created.*

Proof. Consider the set of disconnected vertices to be $\mathcal{D}$ of order d, $0 \leqslant d \leqslant n$. The set of vertices that make up the connected component are $\mathcal{C}$ of order c, $c \neq 1$ and $1 < c \leqslant n$. The graph in question is $\mathcal{T}(V, E)$ where $V = \mathcal{D} \cup \mathcal{C}$. Disconnected vertices are of arbitrarily large distance apart denoted f. The total distance in $\mathcal{C}$ is the constant $td(\mathcal{C})$.

As the unlabeled graph is initially empty the first edge connects two arbitrary vertices. There is no choice to be made, however, once the initial edge is added, it must be shown that all forthcoming distance-impact edges will create a tree. If there are only two vertices the process is complete and a star $K_{1,1}$ is created. After the initial K_2 is created there are three choices for a possible new distance-impact edge additions:

Case 1: The new edge is between two vertices in $\mathcal{C}$. This is not possible. Either the set $\mathcal{D}$ is empty in which case there have already been $n+1$ additions, or there is at least one vertex left in $\mathcal{D}$. Because the vertices in $\mathcal{D}$ are of arbitrary distance from those of $\mathcal{C}$, no matter how much the reduction of the distances in $\mathcal{C}$, the single reduction of e_{vu}, with $v \in \mathcal{D}$ and $u \in \mathcal{C}$, from f to 1 makes such a edge have a higher distance-impact.

Case 2: Two vertices in $\mathcal{D}$ are connected. This reduces the distance of the set $\mathcal{D}$ from $\binom{d}{2} \cdot f$ to $\left(\binom{d}{2} - 1\right) \cdot f + 1$, a total difference of $f - 1$. Note that the other distances in $td(\mathcal{C})$ and the distances between $\mathcal{D}$ and $\mathcal{C}$ is $d \cdot c \cdot f$ and is unchanged.

Case 3: The new edge connects a vertex of $\mathcal{D}$ to one of $\mathcal{C}$. This reduces the total distance of the disconnected and connected sets from $d \cdot c \cdot f$ by at least $(d-1) \cdot c \cdot f$. This is a distance savings of $c \cdot f$.

Case 1 will not occur and it is also the case that the order of the connected component is at least two. This implies that the reduction in case 3 of $c \cdot f$ is greater than that of case 2 because it only has a reduction of $f - 1$ as $c \geqslant 2$. Now all that remains to be shown is that the case 2 edge additions always form a star.

Clearly, the only choices for the first two edges are $K_{1,1}$ and $K_{1,2}$, respectively. These are the base cases for induction. Consider now that i edges are added creating an optimal $K_{1,i-1}$. Consider the addition of another edge. By the inductive hypothesis it must be connected to the star $K_{1,i-1}$. This leaves no non-isomorphic choices.

Case 1: The new edge is connected to a leaf of $K_{1,i-1}$. The total distance in the star $K_{1,i-1}$ is $(i-1)+(i-2)(i-1) = (i-1)^2$. The added distance of the new vertex to that of $K_{1,i-1}$ is comprised of the distance to the incident leaf, the distance to the center, and the distance to the other $i-2$ leaves. This gives a total distance of $(i-1)^2+1+2+3(i-2) = (i+2)(i-1)$.

Case 2: The new edge is connected to the center of $K_{1,i-1}$. The added distance of the new vertex to that of $K_{1,i-1}$ is comprised of the distance to the center and the distance to the other $i-1$ leaves. This is $(i-1)^2 + 1 + 2(i-1) = i^2$.

Comparing the cases the total distances are $i^2 \leqslant (i+2)(i-1)$ if and only if $i \geqslant 2$ which is

covered by the base cases. This new edge must always be added to the center and a star is thus created.

□

4.3 Conclusion

The $\mathcal{P}$-impact process on empty graphs provides some insight on the evolution of invariants as graphs increase in size. However, at the moment much of this work is observational in nature. Note that maximizing distance-impact tends to reduce the diameter whereas the maximum count-impact tends to increase the maximum degree. Both of these claims have been supported by some computational results, but remain 'rules of thumb' rather than proven statements. Further observations can be made to some general trends for graphs, but each stage of the process for distance and count-impact edge addition from empty graphs is given.

Chapter 5

Trees

Many invariants are undefined or uninformative in disconnected graphs; trees provide connected graphs of the smallest size. Because of this minimal extremal structure trees provide a reasonable initial graph for the $\mathcal{P}$-impact process. This chapter also focuses on the graph invariants of the number of spanning trees and the total distance because they are computationally simple to compute and determine interesting graph properties.

It is shown that finding the maximum sptree-impact edge is trivial while contrasting this with the more complex task of determining the maximum count and distance-impact edge. Several conjectures and counterexamples to the a priori determination of the maximum distance-impact edge are provided and they provide some intuition to the role of the maximum distance-impact edge in $G + e$. Finally, a proof is given that shows the maximum distance-impact edge must not be incident to two leaves of a tree.

5.1 Number of Spanning Trees

Trees are acyclic and as such there is only one spanning tree of a tree T, namely T itself. Further, the addition of a single edge to a tree creates only one cycle. This makes the task of finding an arbitrary sptree-impact edge quite simple because it reduces to finding the distances between the vertices in a tree.

Lemma 5.1.1. *The number of spanning trees in an order n connected graph with n edges is equal to the girth of the graph.*

Proof. There must be exactly one cycle in a connected graph with n edges on n vertices. Removing any single edge of this cycle creates a tree, and removing any other edges disconnects the graph. Thus we can create spanning trees only by removing one edge at a time from the only cycle in G. The length of of this cycle is the girth. □

As an immediate result of this lemma, to find a sptree-impact edge of value i in T the edge e_{xy} where $d_T(x, y) = i - 1$ must be added. This result is trivial, but it will be used to highlight the difference between sptree-impact and distance-impact.

5.2 Distance

Where sptree-impact only must create a cycle of an appropriate length, the distance-impact edge must balance the length of the cycle created with the distribution of vertices along that cycle. Because tress are acyclic, total distance is again simple to compute. However, when an extra edge is added, not only does the distance between the newly connected vertices drop to one but a 'shortcut' may be created for many other vertices. This problem becomes difficult because knowing whether any other vertices use this 'shortcut' is a non-trivial problem.

5.2.1 Tree Partitions

In the simple case any edge in $\overline{G}$ may be considered as an $\mathcal{P}$-impact edge in the permissible set. Thus, $O(E) = O(n^2)$ edges must be added individually along with the invariant calculations. Again, as the complexity for invariant computation is fixed, the goal will be to try and reduce the size of the permissible set instead. This can reduce the practical, if not asymptotic, complexity of finding $\mathcal{P}$-impact edges. To this end, graph

structures that would allow the examination very specific vertices of a tree a priori were targeted.

Several observations can be made about what edges cannot be in the permissible set for, say, calculating the maximum impact edge. However, many observations will take at least a call to depth-first search to identify important structures. Such a call will nullify any complexity savings in reducing the size of the permissible set. Instead, some structure in the graph that is fast to compute and can partition the set of edges in $\overline{G}$ into edges to consider and edges to immediately discount should be found.

In trees, several common graph invariants were considered to split the graph's edges. These included looking at degree, centers, and distance. For most of these considerations, counterexamples were found and are given in the next section. However, these counterexamples helped to refine the basis for selecting partitions on the edge set, and led to a proof regarding distance-impact edges among the leaves in Section 5.2.3.

5.2.2 Counterexamples in Distance Impact

To determine the appropriate location for the distance-impact edge several conjectures were put forward, but counterexamples were found and are presented below. These give insight as to what structure is important to finding the distance-impact of edges. In each figure the solid lines represent the edges in G and the dashed lines are the edge(s) of maximum distance-impact in $G + e$. An effort was made to provide the extremally interesting counterexamples where the order is the smallest.

Conjecture 5.2.0.1. *The maximum distance-impact edge is incident to a pair of peripheral vertices.*

Counterexample. In the graph in Figure 5.1 the maximum impact edges do not connect the maximum degree vertices.

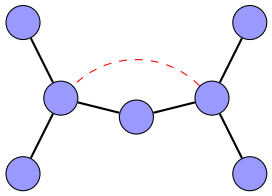

Figure 5.1 Counterexample to conjecture 5.2.0.1

□

Conjecture 5.2.0.2. *The maximum distance-impact edge is incident to two of the maximum degree vertices in trees.*

Counterexample. In the graph in Figure 5.2 the maximum impact edges do not connect the maximum degree vertices.

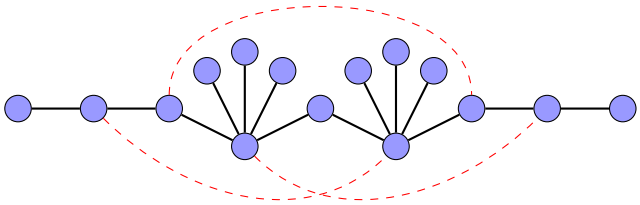

Figure 5.2 Counterexample to conjecture 5.2.0.2

□

Conjecture 5.2.0.3. *The maximum distance-impact edge is incident to a pair of leaves.*

Counterexample. Figure 5.2 also provides a counterexample. □

Conjecture 5.2.0.4. *The maximum distance-impact edge is incident to at least one of the maximum degree vertices in trees.*

Counterexample. Given the following graph G in Figure 5.3 the maximum impact edges do not connect the maximum degree vertices.

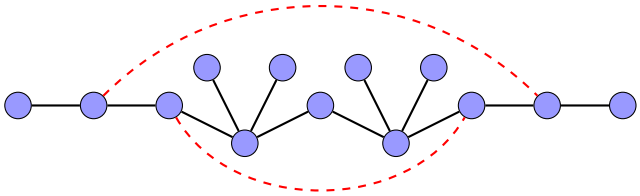

Figure 5.3 Counterexample to conjecture 5.2.0.4

□

Conjecture 5.2.0.5. *The maximum distance-impact edge is incident to at least on vertex of maximum degree or at least one of the leaves.*

Counterexample. Figure 5.3 also provides a counterexample. □

Conjecture 5.2.0.6. *The maximum distance-impact edge is incident to the maximum degree vertices with the long path between them in trees.*

Counterexample. Figure 5.3 also provides a counterexample. □

Conjecture 5.2.0.7. *The maximum distance-impact edge is incident to the maximum degree vertices of the furthest distance from each other.*

Counterexample. Figure 5.3 also provides a counterexample. □

Conjecture 5.2.0.8. *The maximum impact edge in a tree, T, is either incident to the center, C, or its endpoints are in separate components in $T - C$.*

Conjecture 5.2.0.9. *The maximum distance-impact edge joins components in $T-k$, where k is the edge of maximum index.*

Counterexample. In Figure 5.4 the maximum distance-impact edge does not join components of $T - k$.

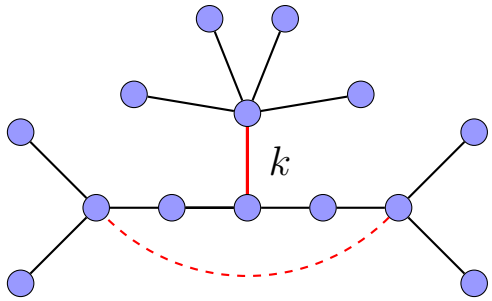

Figure 5.4 Counterexample to Conjecture 5.2.0.9

□

Conjecture 5.2.0.10. *The maximum distance-impact edge joins components in $T - k$ where k is the maximum distIndex edge.*

Counterexample. In Figure 5.5 the maximum distance-impact edge does not join components of $T - k$.

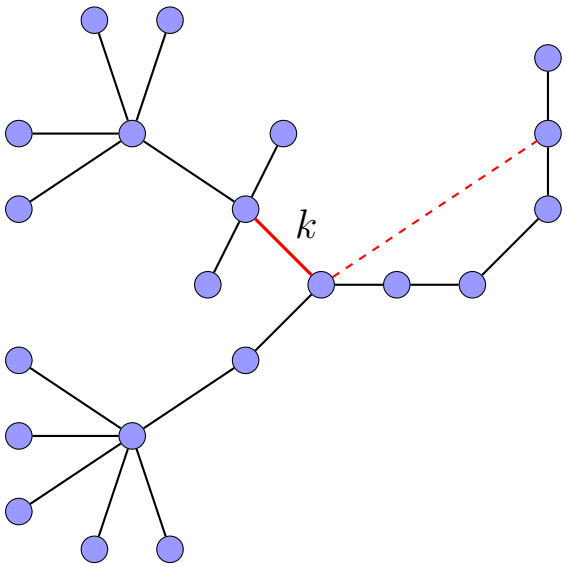

Figure 5.5 Counterexample to Conjecture 5.2.0.10

□

Conjecture 5.2.0.11. *The maximum distance-impact edge joins disjoint components in $T - C$ or is incident to C where C is the center vertex.*

Counterexample. In Figure 5.6, the maximum distance-impact edge is not incident to the center, and $(T - C)$ with the maximum distance-impact edge is disconnected. The vertex labeled 'C' is the center.

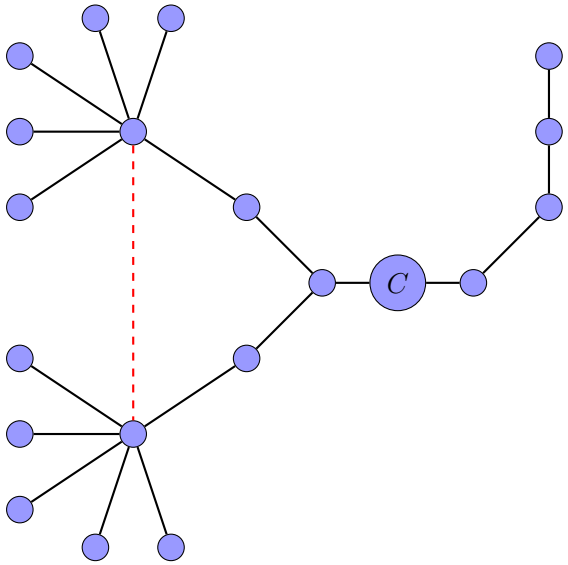

Figure 5.6 Counterexample to Conjecture 5.2.0.11

□

Conjectures 5.2.0.8-10 yield very few counterexamples. The following tables show the computational results for small trees.

Order	3	4	5	6	7	8	9	10	11
Number of Trees	1	2	3	6	11	23	47	106	235
Counterexample	1	0	0	0	0	0	0	0	0
Adjacent to Max Index	1	2	2	4	9	14	33	63	151
Percent Counterexample	100	0	0	0	0	0	0	0	0

12	13	14	15	16	17
551	1301	3159	7741	19320	48629
0	0	3	1	2	16
310	732	1600	3919	8961	22760
0	0	0.095	0.012	0.010	0.033

Table 5.1 Analysis of the counterexamples to the conjecture that the maximum distance-impact edge joins components in $T - k$ where k is the maximum index edge.

Order	3	4	5	6	7	8	9	10	11
Number of Trees	1	2	3	6	11	23	47	106	235
Counterexamples	1	0	0	0	0	0	0	0	0
Adjacent to Max disIndex	1	2	2	4	8	13	27	55	123
Percent Counterexamples	100	0	0	0	0	0	0	0	0

12	13	14	15	16	17
551	1301	3159	7741	19320	48629
0	0	3	1	3	11
267	584	1334	3069	7248	17512
0	0	0.095	0.012	0.016	0.023

Table 5.2 Analysis of the counterexamples to the conjecture that the maximum distance-impact edge joins components in $T - k$ where k is the maximum distIndex edge.

Order	3	4	5	6	7	8	9	10	11	12
Number of Trees	1	2	3	6	11	23	47	106	235	551
Counterexamples	1	0	0	0	0	0	0	0	0	0
Adjacent to Center	0	1	1	2	4	7	12	25	48	112
Percent Counterexamples	100	0	0	0	0	0	0	0	0	0

13	14	15	16	17	18	19	20
1301	3159	7741	19320	48629	123867	317955	823065
0	0	0	0	0	0	0	1
216	491	1112	2608	6266	8583	12401	17686
0	0	0	0	0	0	0	1.2×10^{-6}

Table 5.3 Analysis of the counterexamples to the conjecture that the maximum distance-impact edge joins components in $T - c$ or is incident to c where c is the center vertex.

These tables are not sufficient to prove the number of counterexamples to the conjectures are vansishingly small, but this appears to be the case. This is especially important because the center can be quickly determined. This could imply that the permissible set must be of the form described in 5.2.0.10 with only exceptionally rare cases falling into the class of counterexamples.

5.2.3 Non-Leaf Distance-Impact Edges in Trees

The previous section highlights the underlying slippery nature of maximum distance-impact edges. Specifically, they show how path length and degree can be misleading when trying to find the maximum distance-impact edge. In many trees the maximum distance edge is on the peripheral vertices or incident to the center, but unfortunately this is not the case in general. These examples seem to indicate that a more sophisticated conjecture is needed that will take into account path length and degree.

In this section a proof is presented to reduce the search space for maximum distance-impact edges in trees. Such a reduction is useful in speeding computational evaluations of distance-impact graphs and begins to move towards a method for finding the maximum distance-impact edge in the general case.

Lemma 5.2.1. *The maximum distance impact edge in stars must be incident to two leaves.*

Proof. Note that all vertices are leaves except for the single vertex that is connected to all leaves. This central vertex is of maximum degree, thus any new edge must be between two leaves in the simple case. Because this graph is diameter two, any added edge has edge impact of exactly one. □

Path graphs will be considered to develop the central proposition for proving that the maximum distance-impact edge must not be between two leaves in all but vanishingly rare cases. Consider the path graph, P_n and the following claim:

Claim 5.2.2. *The maximum distance-impact edge in P_n for $n \in \mathbb{Z}^+$ is not incident to the two leaves.*

This claim needs refinement as some small cases do not hold. They are considered exhaustively and Claim 5.2.2 will be refined into Lemma 5.2.7. These small cases encompass $0 \leqslant n \leqslant 6$. When $n = 0$, P_0 is the empty graph and no edges may be added. When $n = 1$ or $n = 2$ the graphs are the complete graphs K_1 and K_2, respectively. Again, these graphs do not allow additional edges in the simple case.For $n = 3$ the graph $P_n \simeq S_{1,2}$ and thus the maximum distance-impact edges must be incident to two leaves.

The first non-degenerate case is P_4 (Figure 5.7). There are two non-isomorphic edges that both have equal distance-impact of 2 as shown in Table 5.4. One of these edges is incident to the two leaves and the other is not. Thus, the maximum distance impact edge can be chosen so as not to be incident to two leaves. However, an edge between the two leaves is also possible.

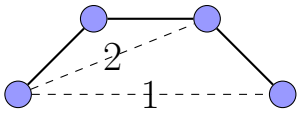

Figure 5.7 P_4 with all non-isomorphic possible edge additions as dashed edges

Edge Addition	Distance-Impact
1	2
2	2

Table 5.4 The distance-impact of all non-isomorphic edge additions in P_4

The case of P_5 is a true exception. Considering the non-isomorphic edge additions in Figure 5.8, the distance-impact is given in Table 5.5. The maximum distance-impact edge is incident to the leaves, and every other possible edge addition yields lower distance-impact values.

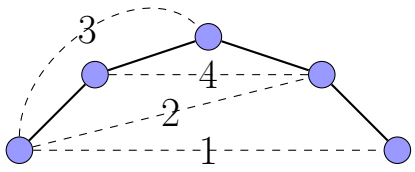

Figure 5.8 P_5 with all non-isomorphic possible edge additions as dashed edges

Edge Addition	Distance-Impact
1	5
2	4
3	2
4	4

Table 5.5 The distance-impact of all non-isomorphic edge additions in P_5

The edge additions to P_6 (Figure 5.9) do not violate Claim 5.2.2 as shown exhaustively in Table 5.6. However, note that the maximum distance-impact edge is incident to one leaf. For $n > 6$, it will be shown that the maximum distance-impact edge is not incident to either leaf in paths to prove Lemma 5.2.7.

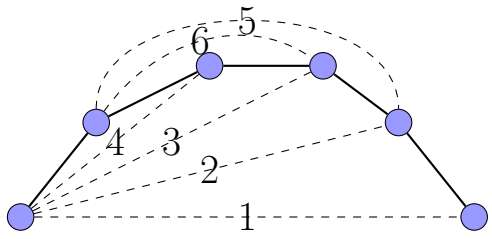

Figure 5.9 P_6 with all non-isomorphic possible edge additions as dashed edges

To prove a modified version of Claim 5.2.2, the total distance in C_n for n odd or even is shown combinatorially. A general graph of C_n is shown on the left of Figure 5.10 The distance values computed here will be used in a later proof.

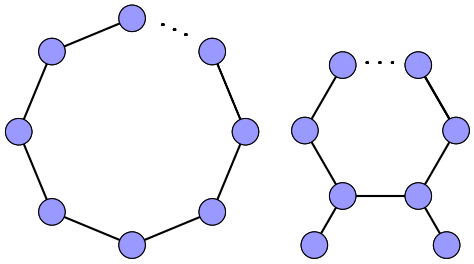

Figure 5.10 C_n (left) and DpC_{n-2} (right), each of order and size n

Lemma 5.2.3. *The total distance in C_n, where n is odd, is*

$$n\left(\sum_{i=1}^{\frac{n-1}{2}} 2i\right) = \frac{n^3 - n}{4}$$

Proof. Due to symmetry, only one vertex is considered, v. The maximum shortest-path distance to any vertex in an odd cycle is $\frac{n-1}{2}$. There are exactly two vertices of this distance

from v. Further, the distances to the vertices on these paths can be found as the summation of $\sum_{i=1}^{\frac{n-1}{2}} i$. Because there are two such paths, the vertex v has distance, from every other vertex, of:

$$\sum_{i=1}^{\frac{n-1}{2}} i + \sum_{i=1}^{\frac{n-1}{2}} i = \sum_{i=1}^{\frac{n-1}{2}} 2i = \frac{n^2 - n}{4}$$

There are n such vertices so the total distance in an odd cycle is:

$$n \sum_{i=1}^{\frac{n-1}{2}} 2i$$

□

Lemma 5.2.4. *The total distance in C_n, where n is even, is*

$$n \left[\left(\sum_{i=1}^{\frac{n}{2}} 2i \right) - \frac{n}{2} \right] = \frac{n^3}{4}$$

Proof. Due to symmetry, only one vertex is considered, v. The maximum shortest-path distance to any vertex in an odd cycle is $\frac{n}{2}$. There is exactly one vertex of this distance from v. Further, the distances to the vertices on these paths can be found as the summation of $\sum_{i=1}^{\frac{n}{2}} i$. Because there are two such paths, the vertex v has distance, from every other vertex, of:

$$\sum_{i=1}^{\frac{n-1}{2}} i + \sum_{i=1}^{\frac{n-1}{2}} i = \sum_{i=1}^{\frac{n-1}{2}} 2i$$

However, this double counts only the single vertex of distance $\frac{n}{2}$ from v. This is corrected by the subtraction of $\frac{n}{2}$ distance. Thus, considering the n such vertices in C_n the total distance in an even cycle is:

$$n \left[\left(\sum_{i=1}^{\frac{n}{2}} 2i \right) - \frac{n}{2} \right]$$

□

One final auxiliary graph is needed. Given a cycle C_n where $n > 5$, the endpoints of one edge are moved such that two pendants of distance 3 from each other are created. This forms the graph on the right in Figure 5.10 dubbed the double distance-three-pendant cycle denoted DpC_{n-2}. The total distance for odd and even n are given in the following lemmas:

Lemma 5.2.5. *The total distance in DpC_{n-2}, for $n > 5$ odd, is:*

$$(n-2)\left(\sum_{i=1}^{\frac{n-3}{2}} 2i\right) + 4\left[\left(\sum_{i=1}^{\frac{n-1}{2}} 2i\right) - 1\right] + 6$$

and is equivalent to:

$$\frac{n^3 - 2n^2 + 11n - 2}{4}$$

Proof. This proof will process combinatorially by assessing the distance between two subgraphs of DpC_{n-2}. The two subgraphs of DpC_{n-2} are the cycle part and the pendants. They are denoted as c and p, respectively, in Figure 5.11. To find the total distance in this graph it suffices to find the total-distance in the cycle part, the total distance in the pendant part, and the total distance between these subgraphs. Note that, although the path between the pendants lies on the cycle, it is not counted twice in the calculations of the total distance of the cycle.

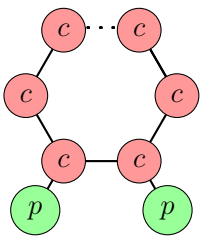

Figure 5.11 The regions of DpC_{n-2}

The three total distance calculations are as follows:

1. **Pendant-Pendant Total Distance:** The definition of the construction gives that the pendants are of distance 3 from each other. Thus, the total distance for this section is $3 + 3 = 6$ exactly.

2. **Cycle-Cycle Total Distance:** The cycle is of length $n-2$ with two pendants. Because n is odd, so is $n-2$. The total distance for odd cycles was given in Lemma 5.2.3. Substituting in $n-2$, the total distance is:

$$(n-2)\left(\sum_{i=1}^{\frac{n-3}{2}} 2i\right) = \frac{n^3 - 6n^2 + 11n - 6}{4}$$

3. **Pendant-Cycle Total Distance:** The paths in this set are all of the paths to the vertices on the cycle plus the single edge addition of the pendant. With this extra edge the calculation of distance is almost identical to that of a cycle with an extra correction term. From the pendant, the half-way point on the cycle is reached after $1 + \frac{n-3}{2} = \frac{n-1}{2}$ edges (reach the cycle and then traverse it). Because $n-2$ is odd (and the pendant is one edge from a the cycle) there are exactly two vertices of this distance, however, this formulation double counts the path from the pendant to the first vertex of the cycle. This sum is thus:

$$\left(\sum_{i=1}^{\frac{n-1}{2}} 2i\right) - 1 = n^2 - 5$$

Because total distance is being calculated, this value must be doubled to count all pendant-to-cycle and cycle-to-pendant paths. Thus, for each pendant there is a total distance of:

$$2\left[\left(\sum_{i=1}^{\frac{n-1}{2}} 2i\right) - 1\right]$$

Finally, because there are two pendants of the same form, this expression must be doubled again to give:

$$4\left[\left(\sum_{i=1}^{\frac{n-1}{2}} 2i\right) - 1\right] = n^2 - 4$$

The sum of all three parts gives the total distance for DpC_{n-2}, n odd, as:

$$(n-2)\left(\sum_{i=1}^{\frac{n-3}{2}} 2i\right) + 4\left[\left(\sum_{i=1}^{\frac{n-1}{2}} 2i\right) - 1\right] + 6$$

□

Lemma 5.2.6. *The total distance in DpC_{n-2}, for $n > 5$ even, is:*

$$(n-2)\left[\left(\sum_{i=1}^{\frac{n-2}{2}} 2i\right) - \frac{n-2}{2}\right] + 4\left[\left(\sum_{i=1}^{\frac{n}{2}} 2i\right) - \left(1 + \frac{n}{2}\right)\right] + 6$$

and is equivalent to:

$$\frac{n^3 - 2n^2 + 12n}{4}$$

Proof. This proof will process combinatorially by assessing the distance between two subgraphs of DpC_{n-2}. The two subgraphs of DpC_{n-2} are the cycle part and the pendants. They are denoted as c and p, respectively, in Figure 5.11. To find the total distance in this graph it suffices to find the total-distance in the cycle part, the total distance in the pendant part, and the total distance between these subgraphs. Note that, although the path between the pendants lies on the cycle, it is not counted twice in the calculations of the total distance of the cycle.

The three total distance calculations are as follows:

1. **Pendant-Pendant Total Distance:** The definition of the construction gives that the pendants are of distance 3 from each other. Thus, the total distance for this section is $3 + 3 = 6$ exactly.

2. **Cycle-Cycle Total Distance:** The cycle is of length $n - 2$ with two pendants. Because n is even, so is $n - 2$. The total distance for even cycles was given in Lemma 5.2.4. Substituting in $n - 2$, the total distance is:

$$(n-2)\left[\left(\sum_{i=1}^{\frac{n-2}{2}} 2i\right) - \frac{n-2}{2}\right] = \frac{n^3 - 6n^2 + 12n - 8}{4}$$

3. **Pendant-Cycle Total Distance:** The paths in this set are all of the paths to the vertices on the cycle plus the single edge addition of the pendant. With this extra edge calculation of distance is almost identical to that of a cycle with an extra correction term. From the pendant, the half-way point on the cycle is reached after $1 + \frac{n-2}{2} = \frac{n}{2}$ edges (reach the cycle and then traverse it). Because $n - 2$ is even (and the pendant is one edge from a the cycle) there is exactly one vertex of this distance. Thus, an extra $\frac{n}{2}$ path is double counted. The path from the pendant to the first vertex of the cycle is also double counted as there is only one vertex of distance one from the pendant. To correct this over counting, a correction term of $1 + \frac{n}{2}$ is subtracted. The distance from the pendant to all of those in the cycle is thus:

$$\left(\sum_{i=1}^{\frac{n}{2}} 2i\right) - \left(1 + \frac{n}{2}\right)$$

This expression is quadrupled (doubled to also count the cycle-pendant distances, and doubled again for the second pendant's distances). This gives a total distance in this subgraph as:

$$4\left[\left(\sum_{i=1}^{\frac{n}{2}} 2i\right) - \left(1 + \frac{n}{2}\right)\right] = n^2 - 5$$

The sum of all three parts gives the total distance for DpC_{n-2}, n even, as:

$$(n-2)\left[\left(\sum_{i=1}^{\frac{n-2}{2}} 2i\right) - \frac{n-2}{2}\right] + 4\left[\left(\sum_{i=1}^{\frac{n}{2}} 2i\right) - \left(1 + \frac{n}{2}\right)\right] + 6$$

□

Edge Addition	Distance-Impact
1	8
2	9
3	6
4	4
5	8
6	6

Table 5.6 The distance-impact of all non-isomorphic edge additions in P_6

Proposition 5.2.7. *The maximum distance-impact edge in P_n for $n \geqslant 6$ is not incident to both leaves. For $n > 6$, the maximum-distance-impact edge is not incident to either leaf.*

Proof. The lemma is proven using the previous combinatorial results, and will be considered in even and odd cases. Consider first that $n > 6$.

1. **Odd Case:** Consider an n-path, P_n. The maximum distance-impact edge is not between two leaves if the total-distance in C_n is greater than the total distance in DpC_{n-2}.

   Consider the following if and only if equivalences for $n > 6$. If the equivalences hold, then the odd case is proven. On the left is the total distance in C_n (odd) from

Lemma 5.2.3 and on the right is the total distance of DpC_{n-2} (odd) from Lemma 5.2.5.

$$n\left(\sum_{i=1}^{\frac{n-1}{2}} 2i\right) = \frac{n^3-n}{4} > (n-2)\left(\sum_{i=1}^{\frac{n-3}{2}} 2i\right) + 4\left[\left(\sum_{i=1}^{\frac{n-1}{2}} 2i\right) - 1\right] + 6$$

$$\frac{n^3-n}{4} > \frac{n^3-6n^2+11n-6}{4} + n^2 - 5 + 6$$

$$n^3 - n > n^3 - 6n^2 + 11n - 6 + 4n^2 + 4$$

$$n^3 - n > n^3 - 2n^2 + 11n - 2$$

$$n^2 - 6n + 1 > 0$$

The final inequality holds for $n > 3+2\sqrt{2} \approx 5.828$, thus the whole set of equivalences are true for the given bounds of $n > 6$. The negative case is not considered because n is a graph's order.

2. **Even Case:** Consider an n-path P_n. The maximum distance-impact edge is not between two leaves if the total-distance in C_n is greater than the total distance in DpC_{n-2}.

   Consider the following if and only if equivalences for $n > 6$. If the equivalences hold, then the even case is proven. On the left is the total distance in C_n (even) from Lemma 5.2.4 and on the right is the total distance of DpC_{n-2} (even) from Lemma 5.2.6. The notation is long, so note that the total distance in an even cycle is:

$$n\left[\left(\sum_{i=1}^{\frac{n}{2}} 2i\right) - \frac{n}{2}\right] = \frac{n^3}{4}$$

   and the total distance in the even double distance-three pendant cycle is:

$$(n-2)\left[\left(\sum_{i=1}^{\frac{n-2}{2}} 2i\right) - \frac{n-2}{2}\right] + 4\left[\left(\sum_{i=1}^{\frac{n}{2}} 2i\right) - \left(1+\frac{n}{2}\right)\right] + 6$$

and is equivalent to:

$$\frac{n^3 - 2n^2 + 12n}{4}$$

Combining these into a series of inequalities, the following if and only if inequalities are evaluated:

$$\begin{aligned}
\frac{n^3}{4} &> \frac{n^3 - 2n^2 + 12n}{4} \\
n^3 &> n^3 - 2n^2 + 12n \\
0 &> -2n^2 + 12n \\
n^2 - 6n &> 0 \\
n(n-6) &> 0
\end{aligned}$$

The final inequality holds for $n > 6$, thus the whole set of equivalences are true for the given bounds of $n > 6$. The negative case is not considered because n is a graph's order. Both cases are now evaluated.

With the even and odd cases, it is shown that the maximum distance-impact edge is not incident to either leaf in P_n with $n > 6$. Along with Figure 5.9 and Table 5.6, when $n = 6$ the maximum distance-impact edge is not incident to both leaves, the proposition is proven. □

There is one additional tree to note before proceeding with the theorem. The graph in Figure 5.12 has no edge that has higher distance-impact than one incident to the leaves as shown in Table 5.7. This graph will be denoted T_{bad} for reasons that will become apparent in the statement of Theorem 5.2.8.

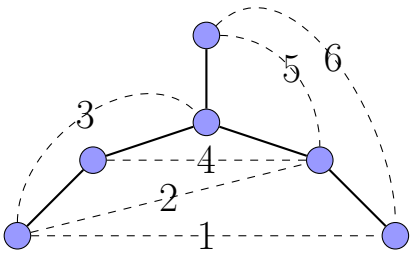

Figure 5.12 T_{bad} with all non-isomorphic possible edge additions as dashed edges

Edge Addition	Distance-Impact
1	5
2	4
3	4
4	4
5	2
6	2

Table 5.7 The distance-impact of all non-isomorphic edge additions in T_{bad}

With the development of Proposition 5.2.7 and the lemmas of this chapter, the principle theorem can be proven.

Theorem 5.2.8. *The maximum distance-impact edge is not incident to two leaves in all trees except stars, P_n, and T_{bad} (given in Figure 5.12).*

Proof. The proof is greatly simplified with the inclusion of Proposition 5.2.7, namely C_n has larger total distance than DpC_{n-2}. Consider a tree that is not a star, T. This tree must have two leaves, but it may have many more. Call any two leaves l_1 and l_2 where $l_1, l_2 \in \{E(G))|l_1 \neq l_2, deg(l_1) = deg(l_2) = 1\}$. An edge not incident to two leaves that has higher distance-impact will be determined for every pair of leaves. The cases will be partitioned based the value of $d_T(l_1, l_2)$. For ease of notation the subscript graph on the distance function is suppressed when the graph is simply T.

1. $\mathbf{d(l_1, l_2) = 2}$: Because the distance between these leaves is 2, the distance between them, upon the addition of $e_{i_1 i_2}$, is reduced to one. However, because they are both leaves, their neighborhood consists of the same vertex, v. This implies that no other path through the tree will use the new edge. Suppose it did. Then for two vertices, x and y, to use the new edge the path $P(x, y)$ must be $P(x, v) + e_{vl_1} + e_{l_1 l_2} + e_{l_2 v} + P(v, y)$. This is trivially shortened as $P(x, v) + P(v, y)$. Because the tree is not a star, move the added edge to any pair of non-leaf vertices that are not adjacent. Their new distance is 1. This is a reduction of at least 1 as those vertices were not previously connected. In the worst case, the total distance is the same, but the added edge is not incident to two leaves.

2. $\mathbf{d(l_1, l_2) = 3}$: When the distance between the selected leaves is 3, they are connected by a copy of P_4. If T is isomorphic to P_4 then, as in Figure 5.7, there is a maximum distance-impact edge not incident to two leaves. In the general tree, because there are no loops, any vertices not contained in the P_4 form subgraphs that only connect to the cut vertices inside of the path. Consider this structure in Figure 5.13 where the dashed edge is the potential non-double-leaf-adjacent candidate edge. The total orders of the vertices connected to each path vertex are $|A|$ and $|B|$. Note, the induced subgraphs $G[A]$ and $G[B]$ are not necessarily connected.

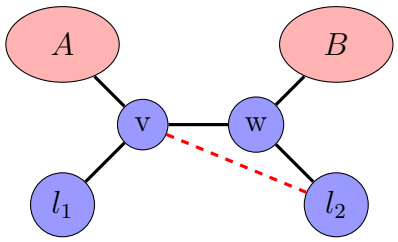

Figure 5.13 Proposed distance-impact edge when selected leaves are of distance 3

If both A and B are empty the graph is P_4. In this case, the dashed edge is of the same distance-impact as the leaf-leaf edge and is only incident to one leaf. Consider

then that at least one of A and B is non-empty. Without loss of generality, A contains at least one vertex. By moving the edge from l_1 to the path vertex v, the distance from l_1 to l_2 increased by 1. However, previously the distance from l_2 to any vertex $i \in A$ was $|P(i,v) + e_{vl_1} + e_{l_1l_2}| = |P(i,v) + 2|$, but this was reduced to $|P(i,v) + e_{vl_2}| = |P(i,v) + 1|$ for every vertex in A. This is a reduction of at least 1. The change of the edge does not affect any distances on P_4. No paths from A to B can use the added edge as it would increase the distance between the subgraphs by one. Therefore, there is always an edge of higher impact than one incident to two leaves of distance 3.

3. $\mathbf{d(l_1, l_2) = 4}$: In this case the underlying path connected l_1 and l_2 is P_5. From Figure 5.8 this graph is known to have a unique maximum distance-impact edge between its leaves. Consider the underlying path as $\{l_1, u, v, w, l_2\}$, and candidate edges i and j. Figure 5.14 details the graph and proposed edges. There are other edges that could be considered, but these are sufficient for the proof.

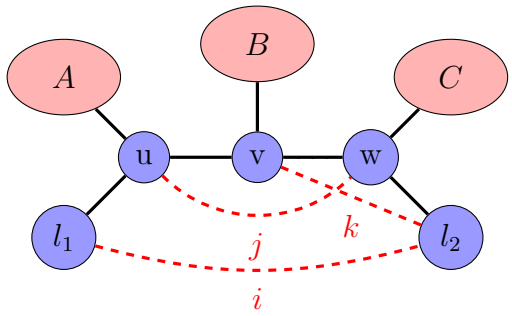

Figure 5.14 Proposed distance-impact edge when selected leaves are of distance 4

Note that edge i is incident to the leaves and should be avoided if possible. Using Figure 5.14, the following three equations represent the total distance savings of each edge addition and are derived from combinatorial methods. Further, $|A| = a$, $|B| = b$, and $|C| = c$. Because all vertices in the sets A, B, and C connect to the P_5 path via a cut vertex, any reduction in path (of length α) between the sets and a vertex on the

reduced path improves the total distance by α for all possible paths. Without loss of generality, consider the affected set to be A. Then the distance drop is $\alpha|A|$ from the set to the vertex. This is $2\alpha|A|$ in total distance.

(a) **Distance-Impact of i:** The leaves were initially of distance 4, this has been reduced to 1 via the edge $e_{l_1 l_2}$. The edge $e_{l_1 l_2}$ also reduces $d(l_1, w)$ and $d(l_2, u)$ from 3 to 2. Finally, the path $P(l_1, w)$ allows access to C by a path that is 1 shorter. This is a reduction of $2|A|$. By symmetry, with $P(l_2, u)$ there is also a reduction in $2|C|$. Considering all other paths in the graph, no other ones can use the additional edge. The distance-impact of edge i is $2|A| + 2|C| + 10$.

(b) **Distance-Impact of j:** The leaves were initially of distance 4, this has been reduced to 3 via the edge e_{uw} a savings of 1 for each. Similarly, the paths $P(u, l_2)$ and $P(l_1, w)$ are also reduced by 1 each for a total of 4. The path between sets A and C are reduced to 1 as $P(u, w)$ is now present. This yields $2|A||C|$ as all vertices in A have a reduced path to C and vice versa. Further, this reduces by 2 more for the $P(u, w)$ path. The vertices l_1 and u are now one closer to C via w. This is a distance-impact of $2|C| + 2|C| = 4|C|$. By symmetry the same reduction happens for l_2 and w to A. Considering all other paths in the graph, no other ones can use the additional edge. The total distance-impact of j is $2|A||C| + 4|A| + 4|C| + 8$.

(c) **Distance-Impact of k:** The addition of k is the only case where paths to B are reduced. The distance from l_1 to l_2 is reduced by 1 saving 2. Also, $k = e_{v,l_2}$ so their distance is reduced by 1 saving an additional 2. The distance from l_2 to v and u are reduced by 1 so the sets A and B are also closer. This reduces the total distance by $2|A| + 2|B|$. Considering all other paths in the graph, no other ones can use the additional edge. The distance-impact of edge i is $2|A| + 2|B| + 6$.

From these combinational arguments three functions on the size of the partitions A,B, and C were derived. These functions are:

$$f_i(a,b,c) = 2a + 2c + 10$$
$$g_j(a,b,c) = 2ac + 4a + 4c + 8$$
$$h_k(a,b,c) = 2a + 2b + 6$$

Here the function $f_i(a,b,c)$ computes the distance-impact of the addition of edge i to a tree where at least two leaves are of distance 4 (similar for g_j and h_k). It suffices to find all (a,b,c) such that for that triple one of the following is true:

$$g_j(a,b,c) \geqslant f_i(a,b,c) \text{ or } h_k(a,b,c) \geqslant f_i(a,b,c)$$

because this implies the distance-impact of edges not incident to two leaves are of higher impact. There are two triples where this does not hold corresponding to P_5 and T_{bad}. Recall that a, b, and c are simply the number of vertices in the sets A, B, and C. Then the case of P_5 occurs when the triple is $(0,0,0)$. This gives:

$$g_j(0,0,0) \ngeqslant f_i(0,0,0)$$
$$2 \cdot 0 \cdot 0 + 4 \cdot 0 + 4 \cdot 0 + 8 \ngeqslant 2 \cdot 0 + 2 \cdot 0 + 10$$
$$8 \ngeqslant 10$$

and

$$h_k(0,0,0) \not\geqslant f_i(0,0,0)$$

$$2 \cdot 0 + 2 \cdot 0 + 6 \not\geqslant 2 \cdot 0 + 2 \cdot 0 + 10$$

$$6 \not\geqslant 10$$

Similarly for T_{bad}, consider the triple $(0,1,0)$. There is only one graph of this form. The inequalities again fail.

$$g_j(0,1,0) \not\geqslant f_i(0,1,0)$$

$$2 \cdot 0 \cdot 0 + 4 \cdot 0 + 4 \cdot 0 + 8 \not\geqslant 2 \cdot 0 + 2 \cdot 0 + 10$$

$$8 \not\geqslant 10$$

and

$$h_k(0,1,0) \not\geqslant f_i(0,1,0)$$

$$2 \cdot 0 + 2 \cdot 1 + 6 \not\geqslant 2 \cdot 0 + 2 \cdot 0 + 10$$

$$8 \not\geqslant 10$$

This confirms the counterexamples and also allows for proof that these are the only two trees that force the additional of a leaf-leaf edge.

To show that there are the only such counterexamples first consider $f_i(a,b,c) = 2a + 2c + 10$ and $g_j(a,b,c) = 2ac + 4a + 4c + 8$. Because these entries correspond to

the size of sets in terms of vertices, all of $a, b, c \in \mathbb{Z}^{0,+}$. Then consider the following if and only if conditionals:

$$f_i(a,b,c) \leqslant g_j(a,b,c)$$
$$2a + 2c + 10 \leqslant 2ac + 4a + 4c + 8$$
$$2a + 2c + 10 \leqslant 4a + 4c + 8$$
$$2 \leqslant 2a + 2c$$

The following holds whenever $a > 0$ or $c > 0$. Note that $2ac$ could be safely dropped as $2ac \geqslant 0$. Regardless of the value of b, if $a > 0$ or $c > 0$ adding edge j will always have a at least the distance-impact of adding i. This single counterexample P_5 is the only one that holds in this case.

Finally, consider $f_i(a,b,c) = 2a + 2c + 10$ and $h_k(a,b,c) = 2a + 2b + 6$. The following if and only if conditionals give:

$$f_i(a,b,c) \leqslant h_k(a,b,c)$$
$$2a + 2c + 10 \leqslant 2a + 2b + 6$$
$$2c + 10 \leqslant 2b + 6$$
$$2c + 4 \leqslant 2b$$
$$c + 2 \leqslant b$$

Comparing the inequality of f_i and g_j, it is known that if $c > 0$ adding edge j is sufficient. This implies that the only case of concern is when $2 \leqslant b$. There are two

such cases where $a = 0, b \leqslant 2, c = 0$. The first is a known counterexample, T_{bad}, from $(0, 1, 0)$. All that must be considered is $(0, 2, 0)$. However, in this case, equality holds because of the following:

$$f_i(0, 2, 0) = 2 \times 0 + 2 \times 0 + 10 = 10 = 2 \cdot 0 + 2 \cdot 2 + 6 = h_k(0, 2, 0)$$

Thus, for this triple the distance-impact of i and k is equal, and k is chosen to avoid a leaf-leaf edge. Therefore, with the stated exceptions of P_5 and T_{bad}, if the $d(l_1, l_2) = 4$ there is an edge of higher distance-impact than $e_{i_1 i_2}$.

4. $\mathbf{d(l_1, l_2) \geqslant 5}$: In this case the underlying path connecting l_1 and l_2 in T is P_6. This allows the use of Proposition 5.2.7. Consider the underlying path as $\{l_1, u, v, w, x, l_2\}$. It is know from Proposition 5.2.7 that the adding the edge e_{uv} yields a larger reduction of total distance than $e_{l_1 l_2}$. Figure 5.15 details the graph and proposed edge.

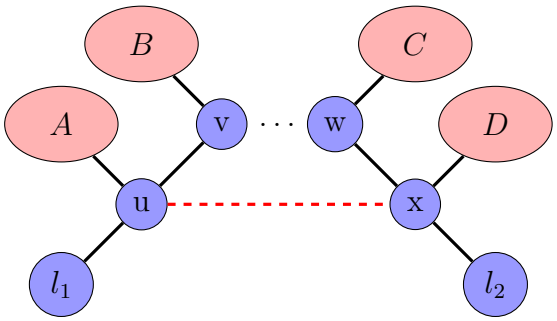

Figure 5.15 Proposed distance-impact edge when selected leaves are of distance at least 5

The distance from the leaves to the subsets A,B,C, and D, has not changed. This implies there are no distance changes to, or from, the leaves and the subsets. Further, if $|A|$ or $|D| > 1$ then the total distance reduces as the previous path for $i \in A$ and $j \in D$ was reduced from $|P(iu) + e_{ul_1} + e_{l_1 l_2} + e_{l_2 x} + P(x, j)|$ to $|P(iu) + e_{ux} + P(x, j)|$. This is a reduction of two for all vertices in A and B. If $d(l_1, l_2) > 5$, then there

are further total distance reductions among the other subsets, but in any case the proposed edge is of greater distance-impact than a leaf-leaf edge where $d(l_1, l_2) \geqslant 5$.

These cases have shown, for all but the degenerate trees ($S_{1,n-1}, P_5$, and T_{bad}), that any edge that is adjacent to two leaves is not the unique maximum distance-impact edge whenever $d(l_1, l_2) \geqslant 2$. This covered all possible cases for the leaves and thus the theorem is proven. □

Random trees are composed of approximately 30% leaves. Because of this fact and Theorem 5.2.8, when finding the maximize distance-impact edge there is no need to search the $\binom{n/3}{2}$-many leaf-leaf possible edges. Because leaves can be quickly found in trees, this result yields an appreciable computational time decrease- even if the asymptotic behavior is unchanged.

5.3 Conclusion

Many aspects of tree structure were considered for the $\mathcal{P}$-impact edges. Even as determining the spanning-impact of edges is trivial, the distance-impact is not. Several negative results along with their respective counterexamples were presented along with a proof about the lack of maximum distance-impact edges incident to leaves in trees. Finding the distance-impact of an edge, a priori, in a tree is still an open question, but settling this question in trees is a useful starting point to explore the same question in general graphs.

It is clear that simple consideration of degree or distance separately is not sufficient to determine the maximum distance-impact edge. From the observations and results here the maximum distance-impact edge may be decided in trees based on an unknown function that chooses an edge based on the degree and distance of its endpoints. Observation seems to imply that distance is less important than degree, but this claim has not been proven.

Chapter 6

$\mathcal{P}$-Impact in Network Completion

The $\mathcal{P}$-impact process has been shown to use edge addition to maintain or optimize a given invariant. Using the distance-related invariants, maximum $\mathcal{P}$-impact edges reduced the global distance measures in graphs. These specific edges seem to have some salient properties regarding the graph structure. To explore this notion, a setting is considered where a partially unobserved graph is taken. The feasibility of identifying missing edges that most improve the accuracy of standard network completion algorithms when edges are added to the graph is examined. This chapter serves as a bridge between the $\mathcal{P}$-impact process and the matrix completion work of the later chapters, and its results on small random graphs should be viewed as motivation to explore matrix completion problems for graph-modeled applications. These matrix completion problems will take the form of link prediction and network completion in Chapter 7 and recommender systems in Chapter 8.

6.1 Experimentation

The $\mathcal{P}$-impact question was designed to determine the most relevant edges to an invariant value in a graph. Instead of finding the optimal edge sets to add to graphs, these experiments measure the error that occurs when edges are obfuscated and network completion is performed. The goal is to determine whether there is an inherent hierarchy

of predicative power among the edges of various graph classes. The classes used were: Erdős-Rényi random graphs, random trees, and random power law (scale-free) graphs.

6.1.1 Network Completion

The network completion process was achieved by using matrix factorization with regularization. Consider a graph G with adjacency matrix A. This method tries to learn the latent features of the adjacency matrix of the graph in the form of two matrices where their product approximates A. This process is detailed much further in Chapter 7, but the results were found by solving the following optimization:

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{A} - \mathbf{U}\mathbf{V}^T\| + \lambda \left(\|\mathbf{U}\|_F^2 + \|\mathbf{W}\|_F^2\right)$$

6.1.2 Evaluation Metrics

The performance of the different algorithms were measured by finding the discrepancy between the original and completed matrices. The widely adopted Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) metrics [41] were used for evaluation. Let $\mathbf{A}$ and $\widehat{\mathbf{A}}$ denote the full and estimated adjacency matrices, respectively. Let $\mathcal{T}$ be the set of unobserved test links. Then,

$$\text{MAE} = \frac{\sum_{(i,j)\in\mathcal{T}} |\mathbf{A}_{ij} - \widehat{\mathbf{A}}_{ij}|}{|\mathcal{T}|},$$

The RMSE metric is defined as:

$$\text{RMSE} = \sqrt{\frac{\sum_{(i,j)\in\mathcal{T}} \left(\mathbf{A}_{ij} - \widehat{\mathbf{A}}_{ij}\right)^2}{|\mathcal{T}|}}.$$

6.1.3 The Process

Five order 25 random graphs of each class were generated. From each of these, nine other graphs were created. In each of these nine graphs, between 10-90% (step size of 10%) of the entries in the adjacency matrix corresponding to potential edges were eliminated. Note that for non-directed graphs the adjacency matrix is symmetric, so in practice only the upper-triangle adjacency matrix was considered. From each of these adjacency matrices two processes occurred. First, the adjacency matrix was completed as-is via matrix factorization. Second, one entry at a time was revealed and replaced with its true value (0 or 1). At this point, the adjacency matrix was completed (again via matrix factorization). In the first case, the basal error rates were calculated (i.e. error without revealing any edges). In the second case, each time an entry was revealed the error of completion was reported as in instance. For the ease of language, in the original graph both absent and present edges will be referred to as 'edges'. The difference being, absent 'edges' will have a true value of 0 in the adjacency matrix while present edges will have a value of 1.

6.1.4 Predictive Edges on Erdős-Rényi Random Graphs

Erdős-Rényi random graphs are one of the most common random graph models. The characterization considered herein was $G(n, p)$. This probabilistic model creates a graph of order n where each possible edge exists with probability p. The size of such graphs is expected to be $\binom{n}{2}p$. The value of n was fixed as 25 and p was allowed to vary from 0.1 to 0.9 with step size of 0.1. All results in this section are for $G(25, 0.5)$, while the results for the other values of p are contained in Appendix A. The results for RMSE are reported in Figure 6.1 while the MAE values are in Figure 6.2. The red bar in each of the runs is the basal completion error. The basal completion error, along with the percent of added edges that result in poorer network completion than the basal rate, are given in Table 6.1.

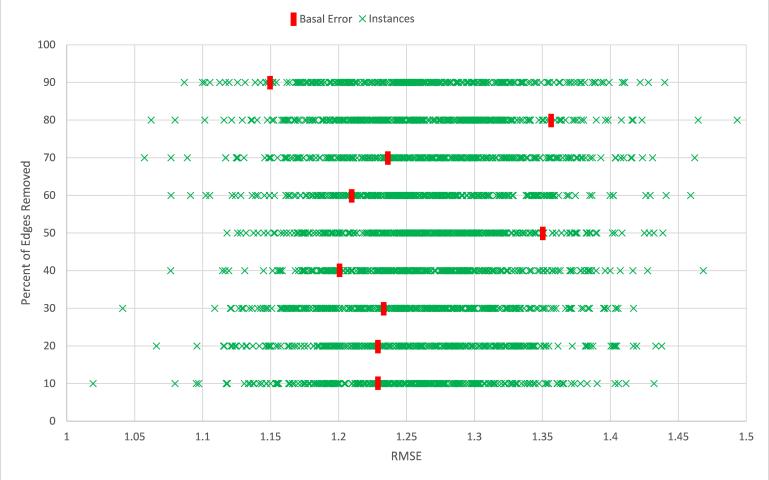

Figure 6.1 RMSE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.5$)

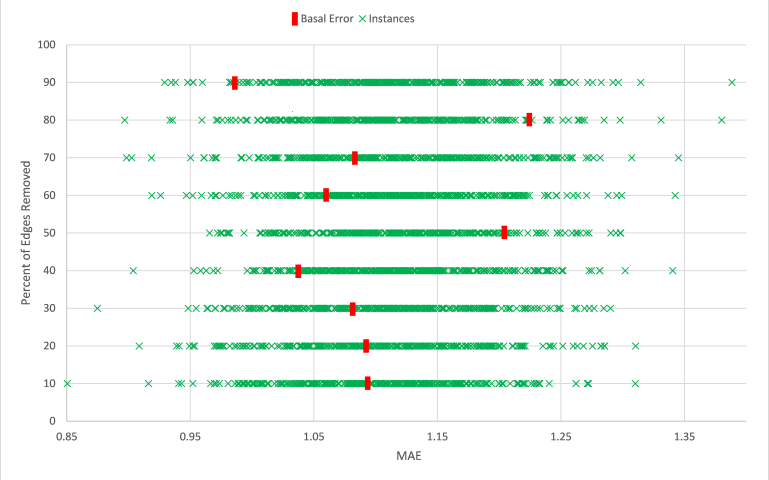

Figure 6.2 MAE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.5$)

Edges Removed (%)	Basal RMSE	Worse Edges (%)	Basal MAE	Worse Edges (%)
10	1.2395	39.66	1.0962	45.66
20	1.2360	44	1.0961	50.66
30	1.2419	41.33	1.0850	37.33
40	1.2048	23.66	1.0474	19.33
50	1.3547	94.33	1.2188	95.33
60	1.2280	31	1.0670	25.66
70	1.2400	35.33	1.0819	30.0
80	1.3679	95.33	1.2317	95.66
90	1.1527	6	0.9886	5.333

Table 6.1 The basal completion error and the percent of edges that produce higher error for all percentages of edges removed for Erdős-Rényi random graphs of the form $G(25, 0.5)$

6.1.5 Predictive Edges on Random Trees

Trees are minimally connected graphs and they also form sparse adjacency matrices. Because of this minimal structure, trees were the focus of the proofs and exploration in Chapter 5. Trees are further explored here for the the predictive power of revealing their edges. The results for RMSE are reported in Figure 6.3 while the MAE values are in Figure 6.4. The red bar in each of the runs is the basal completion error. The basal completion error, along with the percent of added edges that result in poorer network completion than the basal rate, are given in Table 6.2.

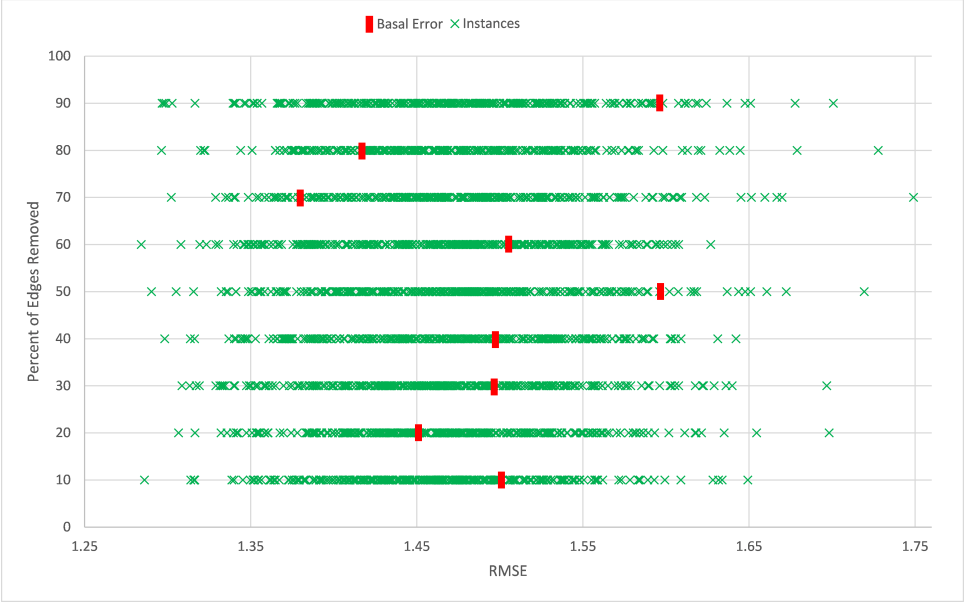

Figure 6.3 RMSE on single edge addition for varied percentages of removed edges for random trees graphs

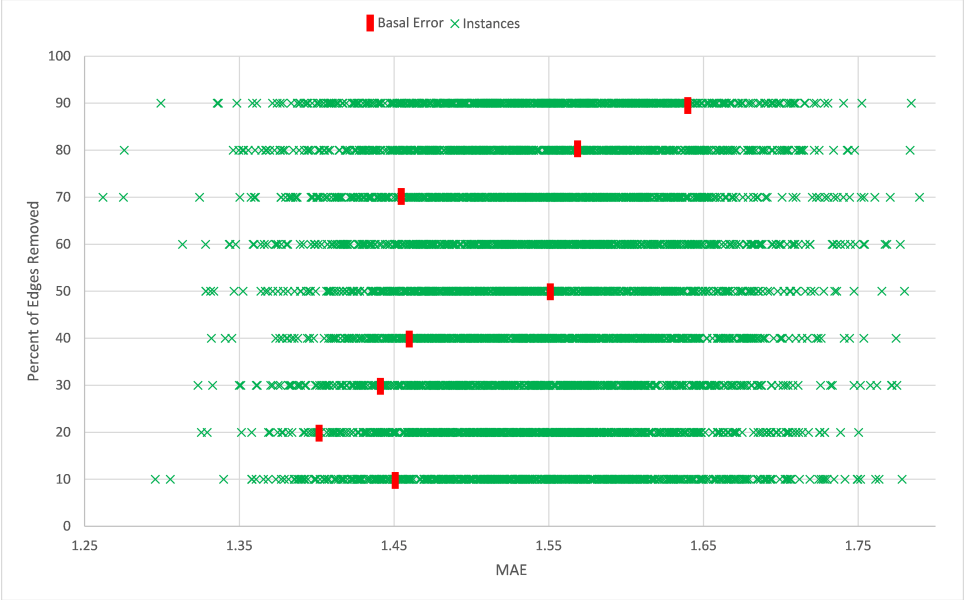

Figure 6.4 MAE on single edge addition for varied percentages of removed edges for random trees

Edges Removed (%)	Basal RMSE	Worse Edges (%)	Basal MAE	Worse Edges (%)
10	1.5105	26.66	1.4515	24.0
20	1.4564	5.666	1.4094	6.333
30	1.5083	25.66	1.4488	23.0
40	1.5159	32	1.4682	33.33
50	1.6186	81.66	1.5582	79.0
60	1.5249	36.33	1.4619	32.33
70	1.3873	0.67	1.3292	0.666
80	1.4209	2.0	1.5662	69.0
90	1.5999	70.66	1.5442	69.33

Table 6.2 The basal completion error and the percent of edges that produce higher error for all percentages of edges removed for random trees

6.1.6 Predictive Edges on Random Power Law Graphs

Power law graphs are a common colloquialism for scale-free networks where the degree sequence follows a power law distribution. This naming convention aligns with the later chapters. Many degree sequences in social media network follow essentially power law distributions [4]. Because of the real world utility of power law graphs, the predictive power is measured here. The results for RMSE are reported in Figure 6.5 while the MAE values are in Figure 6.6. The red bar in each of the runs is the basal completion error. The basal completion error, along with the percent of added edges that result in poorer network completion than the basal rate, are given in Table 6.3.

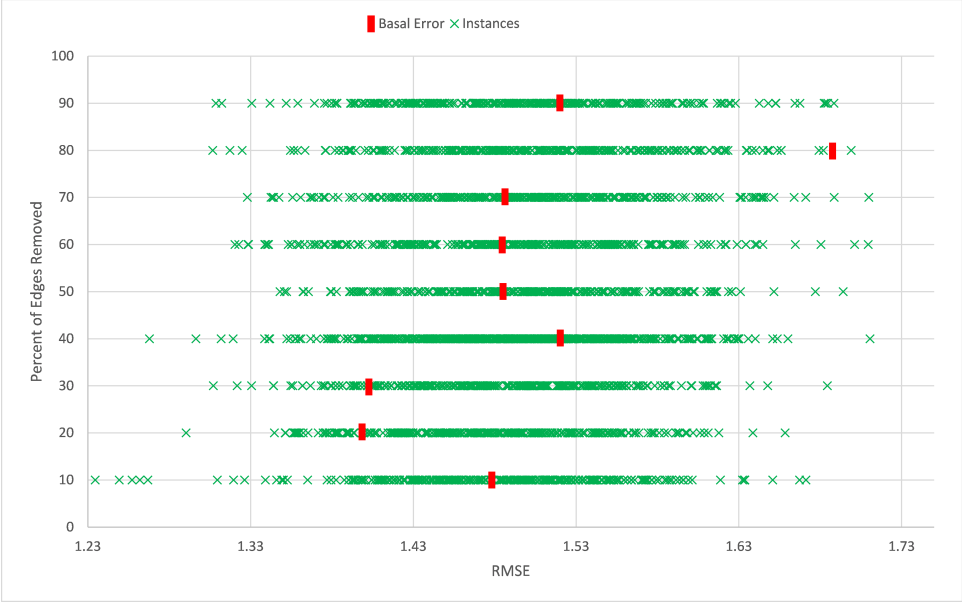

Figure 6.5 RMSE on single edge addition for varied percentages of removed edges for random power law graphs

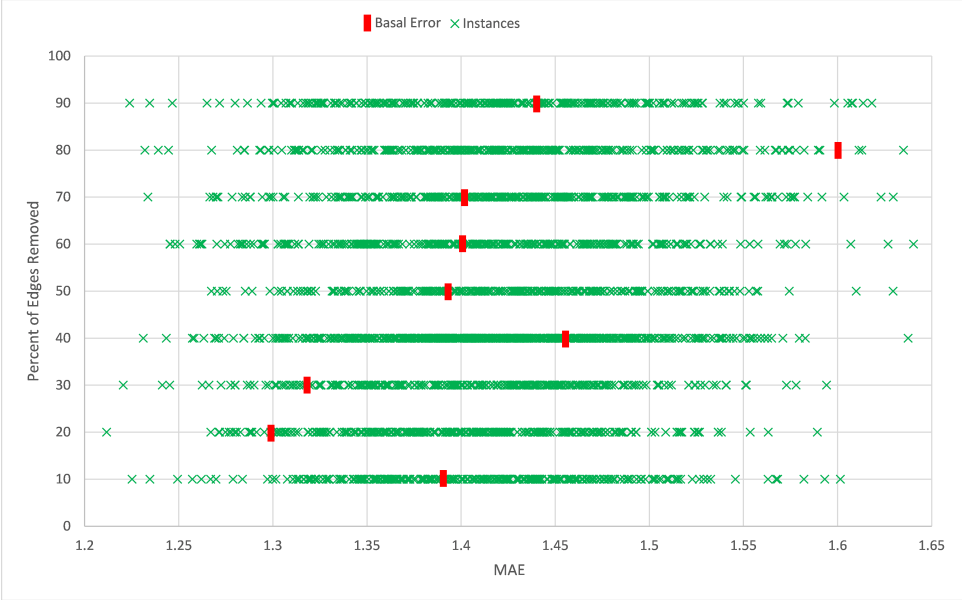

Figure 6.6 MAE on single edge addition for varied percentages of removed edges for random power law graphs

Edges Removed (%)	Basal RMSE	Worse Edges (%)	Basal MAE	Worse Edges (%)
10	1.4787	43	1.3933	37.66
20	1.3979	8.666	1.3068	7.333
30	1.4077	13	1.3263	12.0
40	1.5287	71.66	1.4613	76.0
50	1.4853	37	1.3967	32.33
60	1.4819	39	1.4067	39.33
70	1.4920	50.66	1.4119	48.66
80	1.7013	99.66	1.6267	99.66
90	1.5256	63.66	1.4471	62.0

Table 6.3 The basal completion error and the percent of edges that produce higher error for all percentages of edges removed for random power law graphs

6.2 Power Law Motivation

This experimentation was motivated by examining behavior on power law (scale-free) graphs. Because of their real world utility, another test was undertaken. Considering five random power law graphs of order 15, each of the edges were removed in turn, and the completion error was measured. The $105 = \binom{15}{2}$ edges were then sorted from low-to-high resultant error. The results are presented in Figure 6.7. Intuition seems to imply that the entries of '1' in the sparse adjacency matrix would carry more predictive power. However, in Figure 6.8 the resulting error for only the '1' entries is given showing that these entries can sometimes be omitted without increasing the completion error.

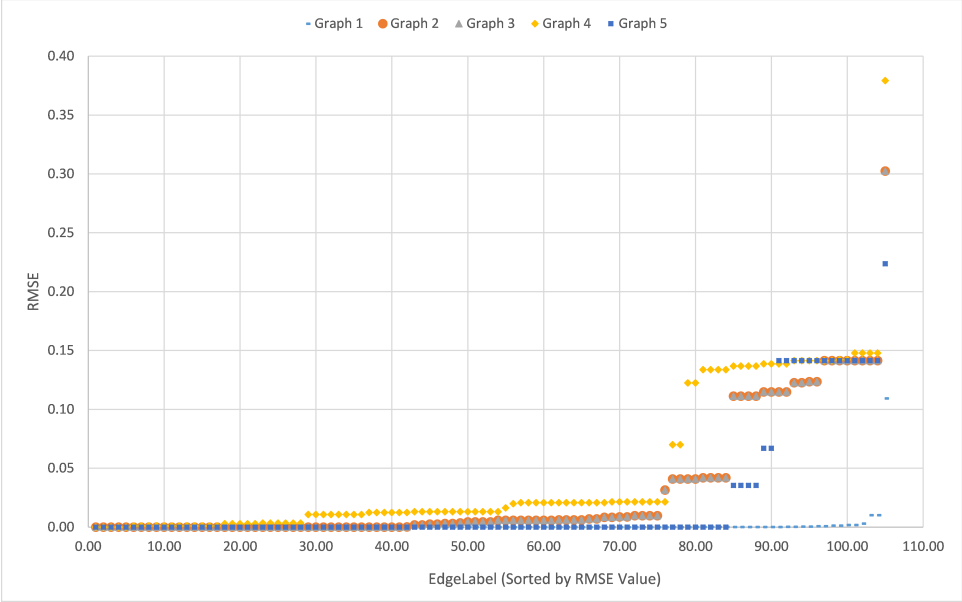

Figure 6.7 RMSE caused by adjacency matrix element deletion for five random power law graphs of order 15

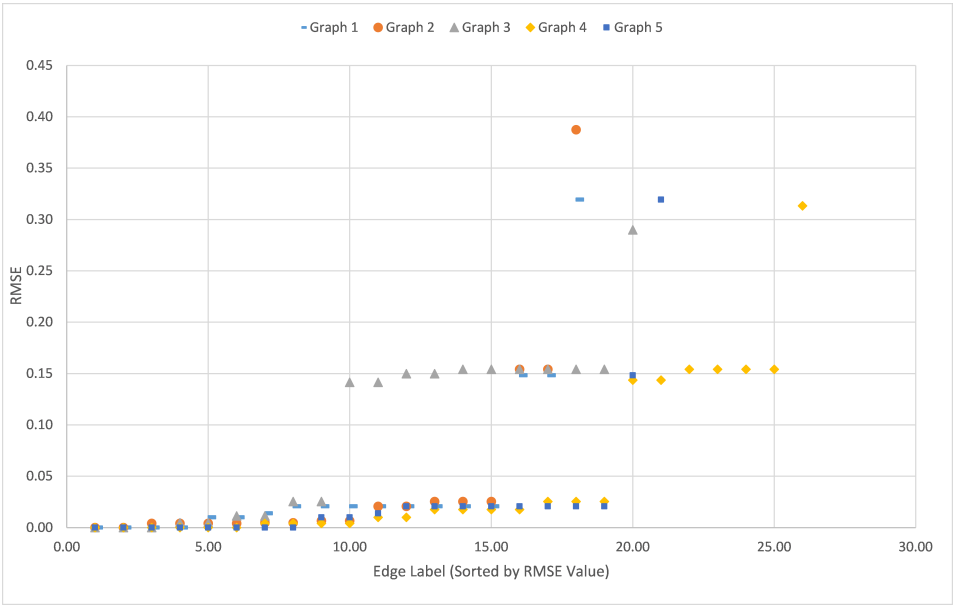

Figure 6.8 RMSE caused by adjacency matrix element deletion (1's only) for five random power law graphs of order 15

6.3 Conclusion

All of the different graph classes produced similar results in the edge removal experiments. Further, the scope of testing was limited. However, as this chapter is meant to serve as motivation for further exploration, there are still some useful observations to be made. Specifically, the predictive power of edges in graphs seems to vary in standard ways that may be proven in the future.

In all cases of edge deletion, approximately 5% of the edges produced significantly better results after network completion was performed. This did not vary much over each of the graph classes, nor was there a significant difference based on percent of obscured edges. The should motivate future work to target those 5% of edges via graph theoretic methods. It further showed that many edges fall below the basal error rate. This also provides motivation to determine a set of edges to avoid adding to a graph.

For the second power law graph examples under single edge deletion, there is also a useful caution given regarding the predictive power of individual edges. Although the edges corresponding to the value of 1 in the adjacency matrix seem to hold more predictive power than the 0 entries, there is a wide variation among the set of 1's. Specifically, they fell into what appears to be three classes of predictive power. What is most interesting here is that these classes were consistent over the testing and had very sharp boundaries between them. This further implies that in graphs there may be a set of highly predictive edges that, by their discovery, network completion techniques would perform with much higher accuracy. It is an open problem to discover the distribution of the predictive edges in general.

Regarding the basal network completion error rate, some interesting observations are made. Principally, the number of edges where their addition causes more error in the matrix factorization than the basal rate is nontrivial. These limited experiments note that the basal error threshold never is lower than the error reduction of the best edge

addition, but it varies wildly. In practical terms this implies that single edge addition can, in many cases, actually make the network completion results worse than had that edge remained unobserved. This may be due to the edges holding undue power in determining graph structures in sparser graphs, thereby biasing the matrix factorization, however more research to explore this must be done. Perhaps this process could be guided via the addition of side information in form the $\mathcal{P}$-impact edges extra-adjacency data sources, but this is beyond the scope of this work.

This ends the exploration of edge impact, but by exploring the role that individual edges play in graphs and matrix factorization techniques led to more questions about network completion. Chapters 7 and 8 extend current matrix factorization algorithms by incorporating side information. They are initially used for network completion, as influenced by Chapters 3 through 6, but are also extended to consider link prediction and recommender system problems.

Chapter 7

Network Completion in Graphs

The need for effective and efficient network completion algorithms occurs in a wide variety of settings such an information retrieval, social network analysis, and computational biology. With the massive growth of big data problems, mere collection of data is generally simple while characterization of said data presents meaningful challenges. There may be natural structure in the data, but only a small sampling may be feasibly obtained. Using this sample, a question arises: can this small sample determine the rest of the network's links?

Because cold-start issues may be present, it is important to also consider side information for transduction of knowledge to the network. Using this side information essentially identifies similar users, and will assist in determining the graph structures that should exist in the unobserved network edges. Using side information also benefits the calculations as the network information is simultaneously incorporated. Thus, if either source has noise or missing entries the other may be able to compensate.

To complete graphs of information in networked systems, an efficient algorithm that significantly departs from the existing methods is proposed. This algorithm uses side information and employs shared subspace learning. From this matrix completion with transduction is applied. This algorithm differs from previous attempts as it decouples the completion and transduction stages. Specifically in phase one, all cold-start nodes are

initially discarded and a sub-network is completed perfectly under some non-restrictive conditions. In phase two, the available side information is used in the transduction stage to complete the entire network– including the unobserved nodes. Although this method is similar to subspace sharing, an important distinction does exist. Namely, a submatrix is perfectly completed before transduction occurs. This allows the proposed algorithm to avoid unbounded error propagation that can occur in subspace sharing methods.

The algorithm, along with theoretical analysis of its recovery error from Barjasteh et al. [5], will be described. There is also extensive analysis on real world datasets that compared the proposed algorithm with many state-of-the-art methods. The contents of this chapter were adapted from the author's contribution to a journal article [1].

7.1 The Assumptions

It is assumed that there is an undirected unweighted graph $G = (V, E)$ where $|V| = n$ are distinguishable nodes. The resultant ground truth adjacency matrix is $A \in \{0, 1\}^{n \times n}$. To approximate real world use, an assumption is made that there is only a partial observation of A, namely $\mathbf{O} \in \{0, 1, ?\}^{m \times m}, 1 \leqslant m \leqslant n$. Note that this sub-matrix is partially observed and the rows and columns of index $n - m$ to n represent cold-start users. The set of users in $\mathbf{O}$ can be said to induce an edge-labeled sub-graph on G where edges are explicitly present, explicitly absent, or unknown. Keeping with observations in real networks, no assumption is made on the distribution of the missing entries. Many classical methods rely on such uniform randomness. There is, however, an assumption that the entries within $\mathbf{O}$ are sampled uniformally at random. If this is not the case, $\mathbf{O}$ may be subsampled, but with abuse of notation for simplicity the subsampled $\mathbf{O}'$ is still denoted $\mathbf{O}$.

There is a further assumption that there is side information available about every node via a social network graph. From this, features of the nodes are extracted and pairwise

[1] **"Network Completion with Provable Guarantees by Leveraging Side Information"**. Iman Barjasteh, Rana Forsati, Dennis Ross, Abdol-Hossein Esfahanian, Hayder Radha, Farzan Masrour. Springer Social Network Analysis and Mining (SNAM). 2016. Submitted.

similarity betwix each of the users is found. This information is stored in the similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$.

Although it has not yet been stated, there is an assumption that the similarity matrix and the networks structure are correlated. This assumption is further refined as the row vectors of $\mathbf{A}$ share some subspace that is spanned by the leading eigenvectors of the similarity matrix $\mathbf{S}$. If such an assumption could not be made, there would be no use in incorporating the side information into the predictions. The extent to which these subspaces are shared will be parameterized later.

7.2 The Algorithm

7.2.1 Previous Approach

A similar method to the proposed algorithm casts the problem as a shared subspace learning framework [79]. This method exploits knowledge from the similarity matrix and transfers this to make structural predictions on the network. This is done by joint matrix factorization where a common subset of basis vectors of $\mathbf{A}$ and $\mathbf{S}$ are learned.

These matrices are factored into three subspaces. The first is shared between the adjacency and similarity, whereas the other two are specific to the matrices themselves. This is formulated as the following optimization problem with λ as the regularization parameter for the norms of the solution matrices:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V},\mathbf{W}} \quad & \frac{1}{2}\|\mathbf{A} - \mathbf{U}\mathbf{V}^\top\|_{\mathrm{F}}^2 + \frac{1}{2}\|\mathbf{S} - \mathbf{U}\mathbf{W}^\top\|_{\mathrm{F}}^2 \\ & + \lambda \left(\|\mathbf{U}\|_{\mathrm{F}}^2 + \|\mathbf{V}\|_{\mathrm{F}}^2 + \|\mathbf{W}\|_{\mathrm{F}}^2\right) \end{aligned}$$

The main issue with the subspace sharing method is that the completion of unobserved entries from the adjacency matrix are made from sampled observed ones, and

the transduction of knowledge from these entries to fully unobserved nodes is carried out simultaneously. Because of this, completion and transduction errors are propagated repetitively and in an uncontrolled way that hinders the effectiveness of incorporating similarity information.

7.2.2 Overview

Diverging from traditional approaches, the proposed algorithm fully recovers the submatrix $\mathbf{O}$ before using similarity information to complete adjacency matrix. This technique decouples the matrix completion from side information transduction. More specifically, the first phase completes the partially observed submatrix $\mathbf{O}$ perfectly due to the assumptions of the distribution of known entries in $\mathbf{O}$. Phase two transducts the links from both the recovered submatrix $\mathbf{O}$ and the complete similarity matrix $\mathbf{S}$. The algorithm is described in Figure 7.1.

7.2.3 Algorithm Details

The first step in this algorithm extracts representative subspace from the similarity matrix $\mathbf{S}$. This has the effect of taking an orthogonal matrix, $\mathbf{U}_s \in \mathbf{R}^{n \times s}$, from the similarity matrix. Here the column space subsumes the column space of adjacency matrix. Further, s is chosen so that it is larger than rank of adjacency matrix. To reduce dimensionality and increase the salience of the similarity information, $\mathbf{U}_s$ takes only the first s-many largest eigenvectors of the singular value decomposition (spectral clustering) of the similarity matrix.

In the second step, the partially observed submatrix $\mathbf{O}$ is fully recovered. Note here, with the presence of cold-start users, matrix completion techniques are not applicable to $\mathbf{A}$. However, because of the assumptions imposed on the distribution of known entries in $\mathbf{O}$, standard matrix completion techniques apply and it can be fully recovered. This completion is done via a convex optimization algorithm [14]. $\widehat{\mathbf{O}}$ denotes the optimal solution to this

1. **Input:**
   - n: the number of nodes in netwrok $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
   - $\mathbf{O}$: the adjacency matrix of subgraph with m nodes
   - $\mathbf{S}$: the partially observed pairwise similarity matrix
   - $s \geqslant \text{rank}(\mathbf{A})$: number of eigenvectors in subspace [Extraction]
2. Extract $\mathbf{U}_s$ from $\mathbf{S}$ by spectral clustering
3. Complete the submatrix $\mathbf{O}$ by solving the convex optimization problem in (8.1) to get $\widehat{\mathbf{O}}$ [Completion]
4. Sample m rows of $\mathbf{U}_s \in \mathbb{R}^{n \times s}$ uniformly at random to create matrix $\widehat{\mathbf{U}}_s \in \mathbb{R}^{m \times s}$
5. Set $\widehat{\Lambda} = \left(\widehat{\mathbf{U}}_s^\top \widehat{\mathbf{U}}_s\right)^\dagger \widehat{\mathbf{U}}_s^\top \widehat{\mathbf{O}} \widehat{\mathbf{U}}_s \left(\widehat{\mathbf{U}}_s^\top \widehat{\mathbf{U}}_s\right)^\dagger$
6. **Output:** $\widehat{\mathbf{A}} = \mathbf{U}_s \widehat{\Lambda} \mathbf{U}_s^\top$ [Transduction]

Figure 7.1 Algorithm for network completion with side information via the proposed algorithm for decoupled completion and transduction

optimization problem. If Ω is the set of observed links in the induced submatrix $\mathbf{O}$, then the matrix may be recovered by solving:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{X}\|_* \\ \text{s.t.} \quad & \mathbf{X}_{ij} = \mathbf{O}_{ij}, \ \forall \ (i,j) \in \Omega. \end{aligned}$$

The third, and final, step is transduction of knowledge from the side information and the completed $\mathbf{O}$ to $\mathbf{A}$. The specific information to be used is the extracted subspace $\mathbf{U}_s$ and the optimal submatrix $\widehat{\mathbf{O}}$. As both the social network adjacency matrix (of rank r) and the recovered submatrix are low rank, a decomposition can be found as follows,

$$\mathbf{A} = \sum_{i=1}^{r} \mathbf{a}_i \mathbf{a}_i{}^\top \quad \text{and} \quad \widehat{\mathbf{O}} = \sum_{i=1}^{r} \widehat{\mathbf{a}}_i \widehat{\mathbf{a}}_i^\top$$

To see this, we can consider $\mathbf{a}_i \in \{1,0\}^n$ and $\widehat{\mathbf{a}}_i \in \{1,0\}^m$ as the corresponding membership assignments to the ith hidden components of the graph. We note that if the similarity matrix is set to be equivalent to the adjacency matrix, then the indicator vectors of connected components are exactly $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_r$.

To formalize the correlation of similarity information and the adjacency matrix, it is assumed that both matrices share a common subspace. Because of this, each vector $\mathbf{a}_i$ with $i \in [r]$ can be uniquely decomposed into parallel and orthogonal components to the shared subspace. This is formalized as $\mathbf{a}_i = \mathbf{a}_i^{\parallel} + \mathbf{a}_i^{\perp}$ where $\mathbf{a}_i^{\parallel}$ is in the column span of $\mathbf{U}_s$ and $\mathbf{a}_i^{\perp}$ is exactly orthogonal to column span of $\mathbf{U}_s$. Because $\mathbf{a}_i^{\parallel}$ belongs to column span of $\mathbf{U}_s$, it can be rewritten as a basis of the shared subspace and the left-singular coefficients such that, for some $\mathbf{b}_i \in \mathbb{R}^s$:

$$\mathbf{a}_i^{\parallel} = \mathbf{U}_s \mathbf{b}_i, i = 1, 2, \cdots, r.$$

The adjacency matrix can be related to the similarity matrix via $\mathbf{a}_i^{\parallel}$ and the decomposition into parallel and orthogonal components.

$$\begin{aligned}
\mathbf{A} &= \sum_{i=1}^{r} \mathbf{a}_i \mathbf{a}_i^{\top} \\
&= \sum_{i=1}^{r} (\mathbf{a}_i^{\parallel} + \mathbf{a}_i^{\perp})(\mathbf{a}_i^{\parallel} + \mathbf{a}_i^{\perp})^{\top} \\
&= \sum_{i=1}^{r} \mathbf{a}_i^{\parallel} \mathbf{a}_i^{\parallel^{\top}} + \mathbf{a}_i^{\parallel} \mathbf{a}_i^{\perp^{\top}} + \mathbf{a}_i^{\perp} \mathbf{a}_i^{\parallel^{\top}} + \mathbf{a}_i^{\perp} \mathbf{a}_i^{\perp^{\top}} \\
&= \sum_{i=1}^{r} \mathbf{a}_i^{\parallel} \mathbf{a}_i^{\parallel^{\top}} + \mathbf{a}_i^{\perp} \mathbf{a}_i^{\perp^{\top}} \\
&= \mathbf{A}_* + \mathbf{A}_E \\
&\approx \sum_{i=1}^{r} \mathbf{a}_i^{\parallel} \mathbf{a}_i^{\parallel^{\top}}
\end{aligned}$$

Here $\mathbf{A}_*$ is fully captured by the similarity information, but the matrix $\mathbf{A}_{\mathrm{E}}$ is pure error as singular vectors are orthogonal to the subspace spanned by the similarity subspace. Because of this, $\mathbf{A}_{\mathrm{E}}$ does benefit from the side information. The error contributed by this matrix into the recovery error of final inferred adjacency matrix is thus unavoidable, but, by disregarding the error term, the adjacency matrix can now be written as:

$$\mathbf{A} = \sum_{i=1}^{r} \mathbf{a}_i^{\parallel} \mathbf{a}_i^{\parallel\top} + \mathbf{A}_{\mathrm{E}} \approx \mathbf{a}_i^{\parallel} \mathbf{a}_i^{\parallel\top}$$

In an observation of the above algorithm, the key to the recovery of the matrix $\mathbf{A}$ is in the estimation of the vectors $\mathbf{b}_i, i = 1, 2, \cdots, r$. Given some reasonable conditions on the number of sampled nodes available in G, the fully recovered submatrix $\widehat{\mathbf{O}}$ and extracted subspace $\mathbf{U}_s$ can be utilized to estimate these $\mathbf{b}_i$ vectors perfectly.

The linear system $\mathbf{a}_i^{\parallel} = \mathbf{U}_s \mathbf{b}_i$ cannot be directly solved for the values of $\mathbf{b}_i$ to estimate $\mathbf{a}_i$. This is because $\mathbf{U}_s$ and $\mathbf{a}_i^{\parallel}$ are not accessible. However, $\mathbf{a}_i^{\parallel}$ can be replaced with a basis $\widehat{\mathbf{U}}_s$ and an accessible vector $\widehat{\mathbf{a}}_i$. Together these can estimate the values of the $\mathbf{a}_i^{\parallel}$ vectors perfectly (with high probability). The subspace $\widehat{\mathbf{U}}_s$ is obtained by taking a uniformly random permutation of $\mathbf{U}_s$ where the selected rows correspond to the rows in $\mathbf{O}$. Then the corresponding ordinary least-squares regression may be solved for an estimation of $\widehat{\mathbf{b}}_i$.

$$\widehat{\mathbf{b}}_i = \min_{\mathbf{b} \in \mathbb{R}^s} \left\| \widehat{\mathbf{a}}_i - \widehat{\mathbf{U}}_s \mathbf{b} \right\|_2^2 = \left(\widehat{\mathbf{U}}_s^{\top} \widehat{\mathbf{U}}_s \right)^{\dagger} \widehat{\mathbf{U}}_s^{\top} \widehat{\mathbf{a}}_i$$

The last equality holds because $\left(\widehat{\mathbf{U}}_s^{\top} \widehat{\mathbf{U}}_s \right)^{\dagger} \widehat{\mathbf{U}}_s^{\top}$ is the optimal solution to the ordinary least-squares regression problem above. Having computed an estimate for each $\mathbf{b}_i$, the adjacency matrix can be fully computed as:

$$\widehat{\mathbf{A}} = \mathbf{U}_s \left(\sum_{i=1}^{r} \widehat{\mathbf{b}}_i \widehat{\mathbf{b}}_i^{\top} \right) \mathbf{U}_s^{\top}.$$

7.3 Experimental Evaluation

Two broad categories of experimentation are taken with the algorithm. Using well known social network and synthetic datasets, the algorithm's performance against standard baseline is tested for link prediction problems. It is also compared against state-of-the art algorithms for network completion. The main reason for splitting the experiments into two parts is the unique challenges that exist in network completion due to the sparseness in the information of the links, which makes it a difficult problem.

7.3.1 Datasets

Four real world datasets were used for the experiments. Two were used for link prediction and the other two were used for network completion. For link prediction a product rating social network site called Epinions and the popular Chinese microblogging site Tencent Weibo were used. The network completion experiments were performed on two large social networks: Facebook and Google+.

1. **Epinions:** Epinions is an explicit trust/distrust social network that yields a directed network. Because the number of distrust relations is very small, only explicit trust relationships between users are considered. The statistics of this dataset are summarized in Table 7.1. This dataset is available through Jiliang Tang at Arizona State University [2]. Generally, the Epinions dataset is used to predict ratings given the trust graph of its users as side information. Here it used to predict the existence of links via a link prediction process that the proposed algorithm found while using the rating information as side information. To extract the features, each user was endowed with a collection of their ratings of the items. These ratings contained scores and timestamps. To extract more features, additional item information (e.g. names, categories, ratings) was included when the users were interested in such items. A final

[2] `http://www.public.asu.edu/~jtang20/datasetcode/truststudy.htm`

feature included a normalized triple for each item category. This triple gave the total number of reviews in that category, and the number of positive and negative reviews per category (normalized by the maximum value of each present in the data). Again, pairwise similarities between users were found via cosine similarity to generate the similarity matrix of users.

2. **Tencent Weibo:** Tencent Weibo (Weibo) is a Chinese microblogging site similar to Twitter. There is a network structure where users can 'follow' another to create a link. In addition to the large network, the side information is quite rich.

   Features for the anonymized users were extracted that included personal information (age, gender, and interests), social information (number of messages sent, messages reposted, users followed), and item classification (categories and keywords). Items and users are distinguished by their features, but for all practical purposes items may act like users in the social network. Items are generally understood to be corporate entities or celebrities. A follower-followee directed graph is given, and the proposed algorithm is used to determine the existence of a directed edge that represents whether or not a given user will follow another. Only considered positive links (where a user follows another) are used, and all negative links (where a user explicitl y choses not to follow another) are ignored for prediction. The data is available from the KDD cup [3]. The original Weibo social directed graph has 1.4 million users and over 73 million links. A small subset of approximately 4,000 users with many observed links among them, and a reasonable amount of side information, was chosen. Find such a set in this large dataset, a modified breadth-first search that penalized adding nodes that did not contain links going back into the explored nodes was deployed. Once this process terminated, other (possibly isolated) nodes at random were added if they had a large amount of side-information until the graph's order threshold was met.

[3] `http://www.kddcup2012.org/c/kddcup2012-track1/data`

The observed average ratio of directed links to nodes for random directed graphs of order approximately 4,000 was 0.46. With this BFS method the directed subgraph of 4,052 vertices that was found had a significantly higher-than-average ratio of directed links to nodes at 0.98. The statistics of data-subsets are summarized in Table 7.1. To generate the similarity matrix of users, cosine similarity between all pairs of users was used.

3. **Facebook:** Facebook is one of the most popular real world social network datasets. On Facebook, people have directed friendship relationships with each other and each user has a profile containing information about themselves. For each user (a node in the network) personal features (e.g. gender, job title, age) are extracted from their profiles. The network of users was produced by combining ego-networks of the Facebook dataset available at the Stanford Large network Dataset Collection [4]. The statistics of datasets are summarized in Table 7.1. The pairwise similarities between users were found via cosine similarity to generate the similarity matrix of users.

4. **Google+:** Google+ is similar to Facebook in terms of relationships between users. Features were also extracted from the individual user profiles in the same way as Facebook. The ego-networks of users were combined together to have a network of users along with their features. This dataset is also available at the Stanford Large network Dataset Collection [5] and the statistics are shown in Table 7.1. The pairwise similarities between users were found via cosine similarity to generate the similarity matrix of users.

[4] `http://snap.stanford.edu/data`
[5] `http://snap.stanford.edu/data`

Dataset	# of Nodes	# of Links	# of Node Features
Weibo	4,052	3,957	16,786
Epinions	90,879	365,422	90,087
Facebook	4,089	170,174	175
Google+	250,469	30,230,905	690

Table 7.1 Statistics of Weibo, Epinions, Facebook, and Google+ datasets

7.3.2 Evaluation Metrics

The performance of the different algorithms was measured by finding the discrepancy between the original and completed matrices. The widely adopted Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) metrics [41] were used for evaluation. Let $\mathbf{A}$ and $\widehat{\mathbf{A}}$ denote the full and estimated adjacency matrices, respectively. Let $\mathcal{T}$ be the set of unobserved test links. Then,

$$\text{MAE} = \frac{\sum_{(i,j)\in\mathcal{T}} |\mathbf{A}_{ij} - \widehat{\mathbf{A}}_{ij}|}{|\mathcal{T}|},$$

The RMSE metric is defined as:

$$\text{RMSE} = \sqrt{\frac{\sum_{(i,j)\in\mathcal{T}} \left(\mathbf{A}_{ij} - \widehat{\mathbf{A}}_{ij}\right)^2}{|\mathcal{T}|}}.$$

7.3.3 Baseline Algorithms

To evaluate the performance of our proposed algorithm, a variety of baseline approaches were considered. The baseline algorithms fall into two categories. The first set included those that were used in the link prediction experiments, and the second set included those that were used in the network completion experiments.

7.3.3.1 Link Prediction Baseline Algorithms

The baseline algorithms are chosen from a wide variety of types of link prediction algorithms to have a fair comparison to the proposed algorithm. All implementation are available in the MyMediaLite Package [29].

1. **BPRMF**: Matrix factorization with Bayesian personalized ranking (BPR) from implicit feedback produces rankings from pairwise classifications. The matrix factorization model provides item prediction optimized for BPR.

2. **BiPolar Slope One (BPSO):** Slope One rating prediction methods weighted by bipolar frequency.

3. **Matrix Factorization (MF):** Matrix factorization where learning is provided by stochastic gradient descent factoring the observed ratings into user and item factor matrices.

4. **Slope One (SO)** Slope One rating prediction method with frequency rating.

5. **User-Item Baseline (UIB):** Assigns an average rating value, and regularization, for baseline predictions. Uses the average rating value, plus a regularized user and item bias for prediction.

6. **CoClustering (CC):** Performs simultaneous clustering on both the rows and the columns of the rating matrix.

7. **Latent Feature log-linear Model (LFM):** Rating prediction method that uses a log-linear model on the latent features of the system.

8. **Biased Matrix Factorization (BMF):** Matrix factorization with parameters for biases of users and items. Utilizes learning provided by stochastic gradient descent.

9. **SVD Plus Plus (SPP):** Singular value decomposition matrix factorization method that makes use of implicit feedback. Further considered what items and users each user has rated.

10. **Sigmoid SVD Plus Plus (SSPP):** Singular value decomposition matrix factorization method that makes user of implicit feedback and utilizes a sigmoid function.

11. **SigmoidItem Asymmetric Factor Model (SFM):** Asymmetric factor model that represents the items based on how the users rated them.

7.3.3.2 Network Completion Baseline Algorithms

The baseline algorithms were chosen from three different types of network completion algorithms. One method only utilize the observed links, one uses network structure as well as node features, and third one learns the shared subspaces.

1. **MF:** Considers only the network structure and ignores the node features. Matrix Completion (MF) algorithm in the standard in this class [83].

2. **MF-NF:** Employing both node features and the structure of network for network completion, a matrix factorization based algorithm that factorizes the adjacency matrix and combines the latent features with explicit features of nodes and links using a bilinear regression model [60]. The implementation of this algorithm is provided by the authors [6].

3. **MF-SS:** Combining the network structure with node features by sharing a subspace. Matrix Factorization with Subspace Sharing (MF-SS) [24] is most like the proposed algorithm.

4. **MC-DT:** This refers to the proposed algorithm formally called Matrix Completion with Decoupled Transduction (MC-DT).

[6] `http://cseweb.ucsd.edu/~akmenon/code/`

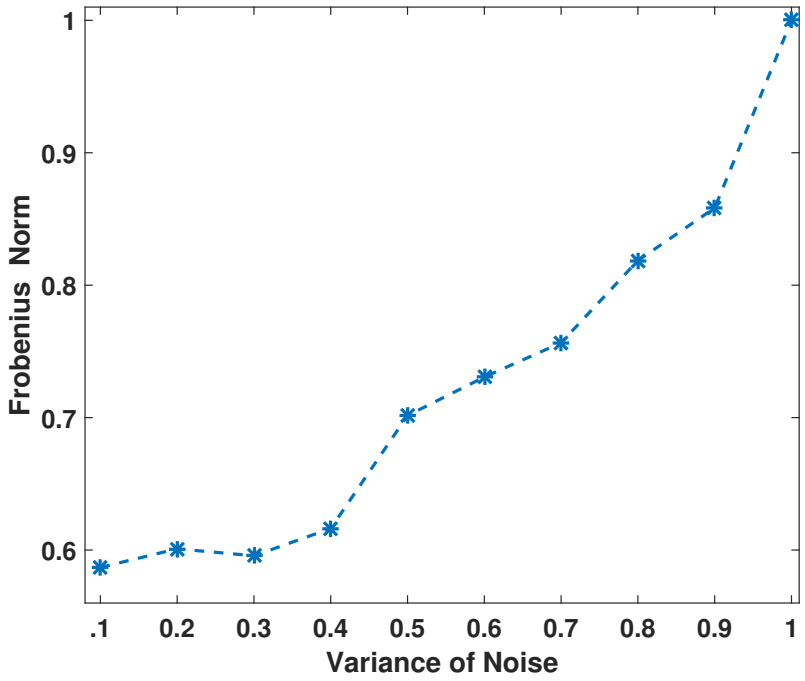

Figure 7.2 The recovery error of the proposed MC-DT algorithm noise variance values

7.3.4 Experiments on Synthetic Datasets

To examine the algorithms on a synthetic dataset, an adjacency matrix $\mathbf{A}$ with 2,048 nodes was generated. To create this network, the rank was fixed at $r = 10$ yielding ten components. The nodes were evenly distributed among these components. A pairwise similarity matrix for the nodes was generated by adding a noise term to the adjacency matrix $\mathbf{S} = \mathbf{A} + \mathbf{N}$, where each entry of noise matrix follows a uniform distribution $\mathbf{N}_{ij} \sim \mathrm{U}(0, 0.5)$. To generate an incomplete network, 20% of all links were randomly removed. The network then had 119,999 links.

7.3.4.1 Effect of Noise in Side Information

To better understand the effect of similarity information on the recovery error, the effect of noise in similarity matrix on the performance of MC-DT algorithm was investigated. The added noise on the similarity matrix followed a zero-mean Gaussian distribution

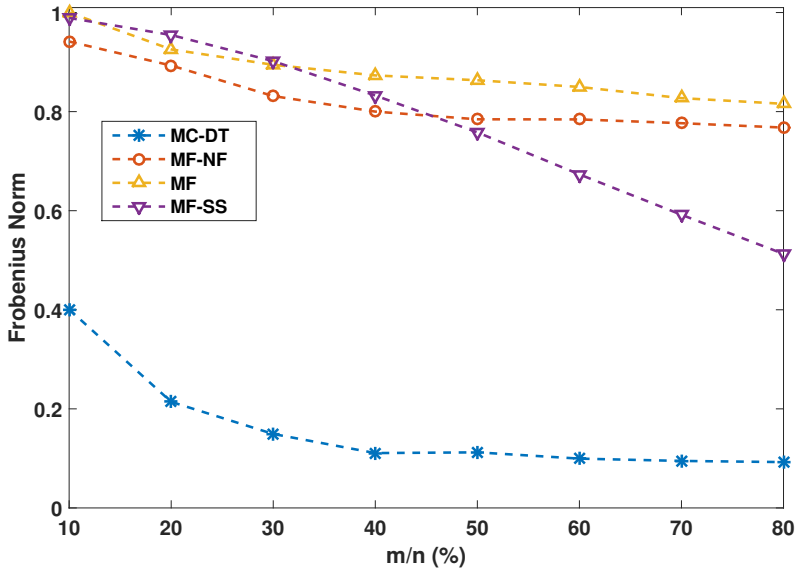

Figure 7.3 The recovery error of different algorithms on a synthetic dataset for different sizes of partially observed submatrix with m nodes

$\mathcal{N}(0, \sigma)$ with different values of variance σ. At the same time, the size of observed submatrix was fixed to $m = 400$. In particular, the variance ranged from $\sigma = 0.1$ to $\sigma = 1$ with step size of 0.1. As Figure 7.2 shows, by increasing the noise, the recovery error increases linearly. This is consistent with the theoretical result of Forsati et al. [55].

7.3.4.2 Effect of Training Size

Investigation on the size of observed submatrix $\mathbf{O}$ as it applies to recovery error was also undertaken. The size of the observed submatrix was varied from 200 to 1600 with step-size of 200. The reported recovery error for the different algorithms is given in Figure 7.3. It can be observed that by increasing m, the recovery error decreases for all of the methods. The fewer unobserved elements we have, the lower the recovery error is. The proposed MC-DT algorithm performs the best, verifying its reliable performance. The significant difference between the recovery error of MC-DT and three other algorithms implies the effectiveness of MC-DT in exploiting similarity information. Because MC-DT

ignores the missing entries of the network and completes the submatrix purely from the observed part of the network, the completion error becomes zero and is not propagated in transduction stage. It is of interest to note that, by increasing the size of observed submatrix (as m approaches n), the effect of similarity information on decreasing recovery error for all methods became less influential. This follows from the fact that existence of links in the network provide much richer information than the pairwise similarity between users.

7.3.5 Evaluation of Link Prediction

The results of applying the proposed algorithm along with different baseline algorithms on the link prediction problem for Epinions and Weibo are contained in this section.

7.3.5.1 Link Prediction on Epinions

To perform the comparison with the baseline algorithms, first the user-user binary trust matrix was created from the available dataset. In this matrix, a '1' indicated a trust relation between two users and '0' indicated unobserved relations. Recall, distrust was not considered. In each dataset, a fraction that varied from the set $\{40\%, 50\%, 60\%, 70\%, 80\%\}$ was chosen to be training, with 10% as validation, and the remaining part was the test set. The test set had its entries set to zero. The data were broken down randomly into five different sets per test size.

The average values of the experiments are given in Table 7.2. The proposed algorithm outperformed all other baseline algorithms in all training sizes. By increasing the training size the error also reduced but there is not a significant drop in the error. This allowed a conclusion to be drawn that by selecting smaller training sizes, accuracy can be maintained. Added benefits of faster computations and overfitting avoidance are also achieved. This further assists in situations where there is not enough data (or computational power) available to use large training sets.

RMSE

Method	Parameters	Train Percentage				
		40%	50%	60%	70%	80%
BPSO	r=20, λ_u=15, λ_i=10, T=50	0.7708	0.7683	0.7683	0.7626	0.7610
MF	r=20, λ=0.015, γ=0.01, T=50	0.7063	0.7127	0.7191	0.7173	0.7269
SO	—	0.7851	0.7780	0.7726	0.7650	0.7611
UIB	—	0.7026	0.7022	0.7032	0.6993	0.7022
CC	C_u=10,C_i=20, T=50	0.7849	0.7776	0.7712	0.7679	0.7625
LFM	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, γ=0.01, T=50	0.7212	0.7110	0.7188	0.7150	0.7122
BMF	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, T=50	0.7086	0.7084	0.7094	0.7061	0.7087
SPP	r=10, λ=0.05, β_λ=0.01, T=50	0.6897	0.6903	0.6926	0.6892	0.6929
SSPP	r=10, λ=0.05, β_λ=0.01, T=50	0.9879	0.9867	0.9851	0.9836	0.9856
SFM	r=10, λ=0.015, β_λ=0.33, γ=0.00, T=50	0.6738	0.6743	0.6770	0.6731	0.6777
MC-DT	r = 37	**0.6623**	**0.6495**	**0.6428**	**0.6430**	**0.6551**

MAE

Method	Parameters	Train Percentage				
		40%	50%	60%	70%	80%
BPSO	r=20, λ_u=15, λ_i=10, T=50	0.9404	0.9318	0.9245	0.9156	0.9078
MF	r=20, λ=0.015, γ=0.01, T=50	0.8363	0.8422	0.8474	0.8468	0.8543
SO	—	0.9515	0.9370	0.9244	0.9129	0.9047
UIB	—	0.8280	0.8278	0.8280	0.8259	0.8276
CC	C_u=10,C_i=20, T=50	0.9285	0.9160	0.9071	0.9013	0.8932
LFM	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, γ=0.01, T=50	0.8425	0.8340	0.8402	0.8377	0.8342
BMF	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, T=50	0.8318	0.8317	0.8319	0.8302	0.8315
SPP	r=10, λ=0.05, β_λ=0.01, T=50	0.8211	0.8215	0.8227	0.8206	0.8227
SSPP	r=10, λ=0.05, β_λ=0.01, T=50	1.2617	1.2605	1.2599	1.2566	1.2596
SFM	r=10, λ=0.015, β_λ=0.33, γ=0.00, T=50	0.8165	0.8167	0.8181	0.8155	0.8182
MC-DT	r = 37	**0.7625**	**0.7591**	**0.7562**	**0.7562**	**0.7573**

Table 7.2 Link prediction results on Epinions dataset and the effects of training size.

7.3.5.2 Link Prediction on Weibo

The Tencent Weibo dataset is given as a series of link recommendation for the users along with side information. For the link recommendations a triple is presented that gives a user, a recommendation of a user or item, and a Boolean. The Boolean represents whether or not a user accepts the recommended user or item. An acceptance can be viewed as a directed edge, from the user to the recommendation, in the social network graph.

To perform the comparison with the baseline algorithms, first the user-user binary trust matrix was created from the available dataset. In this matrix, a '1' indicated a trust relation between two users and '0' indicated unobserved relations. Recall, distrust was not considered. In each dataset, a fraction that varied from the set $\{40\%, 50\%, 60\%, 70\%, 80\%\}$ was chosen to be training, with 10% as validation, and the remaining part was the test set. The test set had its entries set to zero. The data were broken down randomly into five different sets per test size.

The link prediction experiments on Weibo used different training sizes similar to the Epinions experimentation. Table 7.3 shows the results of RMSE and MAE measures on Weibo dataset. The results shown in the Table 7.3 confirmed that the proposed algorithm is the best performing algorithm among the baseline algorithms. There is a significant gap between the results of our proposed algorithm and other baseline algorithms. This was not the case in Epinions dataset results. This significant difference in accuracy was attributed to having more available side information available in Weibo than what was available in Epinions. Again, after having 50% as training and 10% as validation, there is not a significant drop in the accuracy of the results. These results support the efficacy of the decoupled approach to link prediction.

RMSE

Method	Parameters	Train Percentage				
		40%	50%	60%	70%	80%
BPSO	r=20, λ_u=15, λ_i=10, T=50	0.5023	0.5027	0.5017	0.5041	0.5063
MF	r=20, λ=0.015, γ=0.01, T=50	0.4992	0.4999	0.4991	0.5000	0.5018
SO	—	0.5017	0.5034	0.4988	0.4999	0.5072
UIB	—	0.5007	0.5002	0.4996	0.4983	0.5000
CC	C_u=10,C_i=20, T=50	0.5020	0.5004	0.5036	0.4950	0.5022
LFM	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, γ=0.01, T=50	0.5008	0.5007	0.4997	0.4984	0.5000
BMF	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, T=50	0.5004	0.5003	0.4998	0.4986	0.5000
SPP	r=10, λ=0.05, β_λ=0.01, T=50	0.4996	0.5002	0.4998	0.4987	0.5002
SSPP	r=10, λ=0.05, β_λ=0.01, T=50	0.4991	0.5015	0.4943	0.4885	0.4884
SFM	r=10, λ=0.015, β_λ=0.33, γ=0.00, T=50	0.5001	0.5000	0.5001	0.4996	0.5000
MC-DT	r = 37	**0.3454**	**0.3285**	**0.3232**	**0.3237**	**0.3266**

MAE

Method	Parameters	Train Percentage				
		40%	50%	60%	70%	80%
BPSO	r=20, λ_u=15, λ_i=10, T=50	0.5509	0.5488	0.5444	0.5509	0.5496
MF	r=20, λ=0.015, γ=0.01, T=50	0.5012	0.5019	0.5013	0.5025	0.5053
SO	—	0.5489	0.5496	0.5414	0.5436	0.5488
UIB	—	0.5067	0.5066	0.5053	0.5038	0.5053
CC	C_u=10,C_i=20, T=50	0.5612	0.5574	0.5564	0.5511	0.5550
LFM	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, γ=0.01, T=50	0.5093	0.5100	0.5083	0.5068	0.5082
BMF	r=10, β_λ=0.01, $\lambda_{u,i}$=0.05, T=50	0.5021	0.5022	0.5017	0.5006	0.5019
SPP	r=10, λ=0.05, β_λ=0.01, T=50	0.5027	0.5036	0.5031	0.5020	0.5035
SSPP	r=10, λ=0.05, β_λ=0.01, T=50	0.6987	0.7003	0.6951	0.6908	0.6908
SFM	r=10, λ=0.015, β_λ=0.33, γ=0.00, T=50	0.5005	0.5002	0.5002	0.4997	0.5001
MC-DT	r = 37	**0.4624**	**0.4457**	**0.4419**	**0.4421**	**0.4437**

Table 7.3 Link prediction results on the Weibo dataset and the effects of varying training size

7.3.6 Evaluation of Network Completion

Now the proposed algorithm is compared to the state-of-the-art algorithms for network completion. The following results were found using the Facebook and Google+ social network datasets. In this scenario, varying training sizes were again utilized. Again the datasets were divided into training, validation, and testing as described in the last section. When the training size is 20%, 20% of the nodes and the corresponding links from the each social network are randomly selected to predict the rest of network. The performance of MC-DT along with the baseline network completion algorithms on these two datasets was evaluated.

7.3.6.1 Network Completion in Facebook

In Figures 7.4 and 7.5, the RMSE and MAE of different algorithms on Facebook dataset are shown, respectively. In these plots the improvement of all methods gradually decreased as more of the network structure was observed.

7.3.6.2 Network Completion in Google+

Table 7.4 shows the performance of the MC-DT in comparison with the other baseline algorithms on Google+'s social network. The algorithm's performance exhibits similar behavior to the one presented in previous experiment with Facebook. Specifically, MC-DT outperforms the other network completion algorithms.

Method	RMSE (30%)	MAE (30%)	RMSE (60%)	MAE (60%)
MF	0.97201	0.89132	0.81749	0.76601
MF-NF	0.86384	0.82094	0.71505	0.68935
MF-SS	0.81368	0.78820	0.58369	0.68160
MC-DT	**0.78806**	**0.50186**	**0.50851**	**0.30011**

Table 7.4 Comparison of different algorithms on the Google+ with different percentages of observed nodes

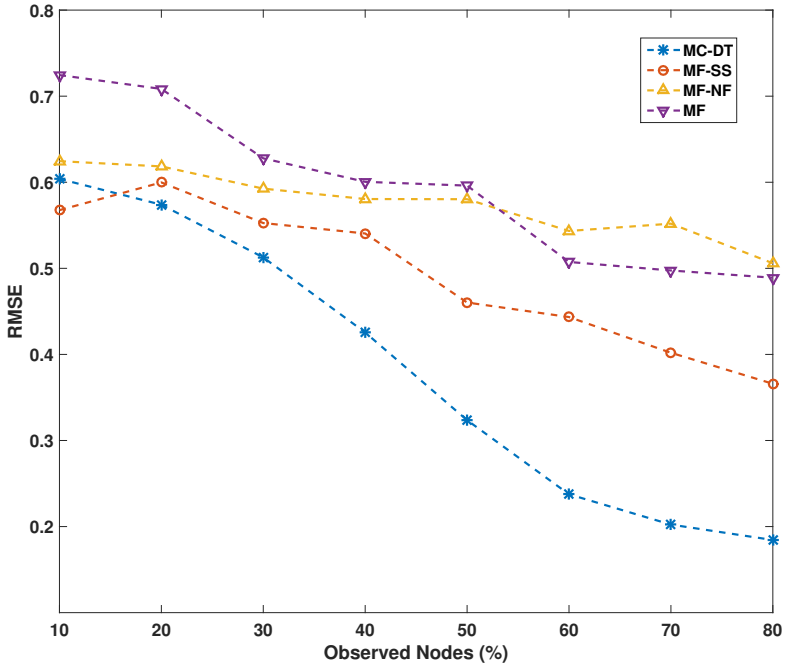

Figure 7.4 The recovery of four algorithms on the Facebook dataset measured in RMSE under different percentages of observed nodes

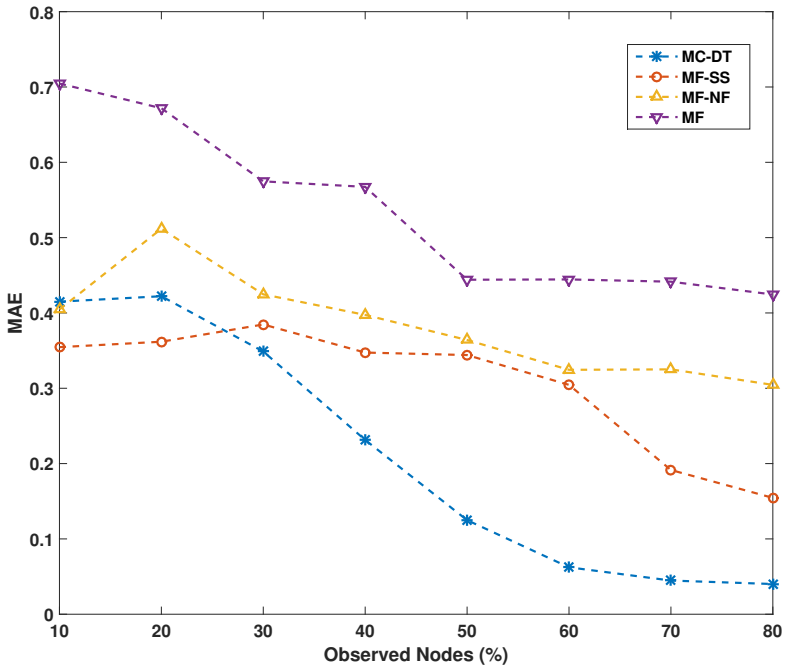

Figure 7.5 The recovery of four algorithms on the Facebook dataset measured in MAE under different percentages of observed nodes

7.4 Conclusion

An effective algorithm in MC-DT was developed for network completion and link prediction with auxiliary similarity information on the nodes. Its comparison to the state-of-the-art baselines on synthetic and real datasets reveals that this algorithm exhibits improved performance in terms of recovering the full network. This improvement is due to the decoupled completion of the observed submatrix from transduction of knowledge while exploiting similarity information.

MC-DT greatly outperformed the other baseline algorithms in of the evaluation metrics. Both the RMSE and MAE errors of the completed network are more than two times smaller for MC-DT than the others. These experiments demonstrated the effectiveness of MC-DT in exploiting the similarity information to recover the full network. Although both use matrix factorization techniques, MC-DT does utilize node features while naive MF does not. MC-DT achieves better performance, as it combines the information from the node features and the network structure. The significant gap between the performance of these two algorithms demonstrated the importance of similarity information in network completion problems. When networks are incomplete, the performance of MC-DT degrades gracefully. Because it can rely on the node features, the algorithm can compensate for the lack of links in the overall network structure. Finally, by comparing the results for synthetic and real datasets, in the synthetic dataset the decrease of error (by increasing the number of observed nodes) is significant initially, but then slowly shrinks. This contrasts to real world datasets where the decrease is roughly linear.

Chapter 8

Recommender Systems

As commerce has shifted to the internet, e-commerce websites (e.g. Amazon, Walmart.com), media consumption (e.g. Netflix, Amazon Video), and news media (e.g. NYtimes.com, CNN.com) have experienced large growth in both users and available services. With this growth, there became a demand for curated content for the end users. This gave rise to efficient recommender systems that would present users with items of personal relevance. However, just as in the network completion case, cold-start problems with the users and items have not been adequately addressed.

The goal of the algorithm proposed in this chapter is to give an efficient matrix factorization approach using similarity information derived from side information that is accurate and can seamlessly incorporate sparse or cold-start datasets. This is an extension of the work on network completion presented in Chapter 7. A two-stage algorithm that decouples the completion and transduction stages during the matrix completion is presented here. In the first phase cold-start items and users are excluded and a rating submatrix is fully completed. In the next phase, this submatrix and a similarity matrix extracted from the available side information are used to transduct information for the cold-start users and items. Unlike most subspace sharing approaches, there is no error propagation of completion during transduction.

The theoretical results are further enhanced with comprehensive experiments on a few benchmark datasets to demonstrate the merits and advantages of proposed framework in dealing with cold-start problems. The results demonstrate the superiority of the proposed framework over several of state-of-art cold-start recommender algorithms. The contents of this chapter were adapted from the author's contribution to a published journal article [1].

8.1 The Assumptions

There is some low-rank ratings matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$ where there are n users providing real-valued ratings for m items. For the purposes of this work the ratings were taken to be values from 1 to 5 inclusive; however, this assumption could be easily changed based on the problem domain. There is a partially observed submatrix $\mathbf{M} \subseteq \mathbf{R}$ where $\mathbf{M} \in \{0, 1, ?\}^{p \times q}, 1 \leqslant p \leqslant n, 1 \leqslant q \leqslant q$. The rows of $\mathbf{R}$ with no values are users who are cold-start (i.e. users who have not provided any historical ratings). The columns of $\mathbf{R}$ with no entries are cold-start items (i.e. items where no feedback has ever been provided). As with the network completion case, no assumptions have been made about the distribution of the ratings.

There is a further assumption that there is side information available about every user and item via a social network graph and item specification information. From this, features of the users and items, respectively, are extracted and pairwise similarity is computed for each. This information is stored in the similarity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ for the users and $\mathbf{B} \in \mathbb{R}^{m \times m}$ for the items.

There is a final assumption that the rating matrix and the similarity matrices are correlated. Thus, the pairwise share some latent information. Specifically, the row vectors of $\mathbf{R}$ share an underlying subspace with the leading eigenvectors of $\mathbf{A}$, and the column vectors share an underlying subspace with the leading eigenvectors of $\mathbf{B}$. The amount of

[1] **"Cold-Start Recommendation with Provable Guarantees: A Decoupled Approach"**. Iman Barjasteh, Rana Forsati, Dennis Ross, Abdol-Hossein Esfahanian, Hayder Radha. IEEE Transactions on Knowledge and Data Engineering (TKDE). 2016. Accepted.

subspace sharing will be parameterized, but if no such subspace existed there would be no benefit to applying this approach.

8.2 The Algorithm

8.2.1 Previous Approach

The previous methods [79] for predicting ratings via a shared subspace approach is very similar to that of the network completion approach. Again, the problem is cast as a shared subspace learning framework. This method exploits knowledge from the similarity matrix and transfers it to make structural predictions on the network. This is achieved by joint matrix factorization to learn a common subset of basis vectors for the rating matrix $\mathbf{R}$ and the similarity matrices $\mathbf{A}$ and $\mathbf{B}$ for users and items as formulated in the following optimization problem with λ as the regularization parameter for the norms of the solution matrices and common latent space representation is given by using the same matrices $\mathbf{W}$ and $\mathbf{Z}$:

$$\begin{aligned}\min_{\substack{\mathbf{U}\in\mathbb{R}^{n\times r},\mathbf{V}\in\mathbb{R}^{m\times r},\\ \mathbf{W}\in\mathbb{R}^{n\times r},\mathbf{Z}\in\mathbb{R}^{m\times r}}} & \frac{1}{2}\|\mathbf{R}-\mathbf{U}\mathbf{V}^\top\|_{\mathrm{F}}^2+\lambda\left(\|\mathbf{U}\|_{\mathrm{F}}^2+\|\mathbf{V}\|_{\mathrm{F}}^2\right)\\ &+\frac{1}{2}\|\mathbf{A}-\mathbf{U}\mathbf{W}^\top\|_{\mathrm{F}}^2+\frac{1}{2}\|\mathbf{B}-\mathbf{Z}\mathbf{V}^\top\|_{\mathrm{F}}^2\\ &+\lambda\left(\|\mathbf{W}\|_{\mathrm{F}}^2+\|\mathbf{Z}\|_{\mathrm{F}}^2\right)\end{aligned}$$

Again, this approach's performance is impaired because the completion of the unobserved entries in rating matrix $\mathbf{R}$ and transduction of knowledge from these entries to cold-start users/items via similarity matrices is carried out simultaneously. Completion and transduction errors are propagated repetitively and uncontrollably. The problem with

error propagation becomes even worse due to the non-convexity of optimization problems, which will be described later.

8.2.2 Overview

As noted in Chapter 7, the proposed algorithm fully recovers the submatrix $\mathbf{M}$ before using similarity information to complete the adjacency matrix. This technique decouples the matrix completion from side information transduction. More specifically, the first phase completes the partially observed submatrix $\mathbf{M}$ perfectly due to the assumptions of the distribution of known entries in $\mathbf{M}$. Phase two transducts the links from both the recovered submatrix $\mathbf{M}$ and the complete similarity matrices $\mathbf{A}$ and $\mathbf{B}$. The algorithm is described in Figure 8.1.

8.2.3 Algorithm Details

An orthogonal matrix $\mathbf{U_A} \in \mathbb{R}^{n \times s}$ was constructed where its column space subsumes the row space of the rating matrix. Similarly, $\mathbf{U_B} \in \mathbb{R}^{m \times s}$ was constructed for the column space. Both $\mathbf{U_A}$ and $\mathbf{U_B}$ were created by taking their leading s eigenvectors. The value of s relates to the extent to which the extracted subspaces subsume the row and column spaces of the rating matrix and is thus dependent on the quality of the side information. This allowed the rank r matrix to be decomposed as:

$$\mathbf{R} = \sum_{i=1}^{r} \mathbf{u}_i \mathbf{v}_i^\top.$$

Colloquially, this expressed the ratings matrix as its user and item components. Further, the i-th user's latent features vector $\mathbf{u}_i$ can be decomposed in a unique way into two parts that are parallel and orthogonal to the shared subspace as $\mathbf{u}_i = \mathbf{u}_i^{\parallel} + \mathbf{u}_i^{\perp}$. Here, $\mathbf{u}_i^{\parallel}$ is spanned by $\mathbf{U_A}$ while $\mathbf{u}_i^{\perp}$ is orthogonal to $\mathbf{U_A}$. For the j-th item's latent features $\mathbf{v}_i$ can be written as $\mathbf{v}_i^{\parallel} + \mathbf{v}_i^{\perp}$ where, $\mathbf{v}_i^{\parallel}$ is spanned by $\mathbf{U_B}$ while $\mathbf{v}_i^{\perp}$ is orthogonal to $\mathbf{U_B}$.

1. **Input:**
   - $\mathbf{R} \in \mathbb{R}^{n \times m}, r$: observed matrix and its rank
   - $\mathbf{A} \in \mathbb{R}^{n \times n}$: the users' similarity matrix
   - $\mathbf{B} \in \mathbb{R}^{m \times m}$: the items' similarity matrix
2. Extract the maximal recoverable rating sub-matrix $\mathbf{M} \in \mathbb{R}^{p \times q}$
3. Complete the sub-matrix $\mathbf{M}$ to get $\widehat{\mathbf{M}}$
4. Decompose $\widehat{\mathbf{M}}$ as $\widehat{\mathbf{M}} = \sum_{i=1}^{r} \hat{\mathbf{u}}_i \hat{\mathbf{v}}_i^\top$
5. Extract subspaces $\mathbf{U_A}$ and $\mathbf{U_B}$ by spectral clustering from similarity matrices $\mathbf{A}$ and $\mathbf{B}$, respectively
6. Compute $\hat{\mathbf{a}}_i = \left(\widehat{\mathbf{U}}_\mathbf{A}^\top \widehat{\mathbf{U}}_\mathbf{A}\right)^\dagger \widehat{\mathbf{U}}_\mathbf{A}^\top \hat{\mathbf{u}}_i, i = 1, 2, \cdots, r$
7. Compute $\widehat{\mathbf{b}}_i = \left(\widehat{\mathbf{U}}_\mathbf{B}^\top \widehat{\mathbf{U}}_\mathbf{B}\right)^\dagger \widehat{\mathbf{U}}_\mathbf{B}^\top \hat{\mathbf{v}}_i, i = 1, 2, \cdots, r$
8. Compute $\widehat{\mathbf{R}} = \mathbf{U_A} \left(\sum_{i=1}^{r} \hat{\mathbf{a}}_i \widehat{\mathbf{b}}_i^\top\right) \mathbf{U}_\mathbf{B}^\top$
9. **Output:** $\widehat{\mathbf{R}}$

Figure 8.1 Algorithm for rating prediction using side information via the proposed algorithm for decoupled completion and transduction

Having decomposed the latent features as above into two parallel and orthogonal components, the rating matrix can be rewritten as:

$$
\begin{aligned}
R &= \sum_{i=1}^{r} \mathbf{u}_i \mathbf{v}_i^\top \\
&= \sum_{i=1}^{r} (\mathbf{u}_i^{\parallel} + \mathbf{u}_i^{\perp})(\mathbf{v}_i^{\parallel} + \mathbf{v}_i^{\perp})^\top \\
&= \sum_{i=1}^{r} \mathbf{u}_i^{\parallel} \mathbf{v}_i^{\parallel^\top} + \mathbf{u}_i^{\parallel} \mathbf{v}_i^{\perp^\top} + \mathbf{u}_i^{\perp} \mathbf{v}_i^{\parallel^\top} + \mathbf{u}_i^{\perp} \mathbf{v}_i^{\perp^\top} \\
&= \sum_{i=1}^{r} \mathbf{u}_i^{\parallel} \mathbf{v}_i^{\parallel^\top} + \sum_{i=1}^{r} \mathbf{u}_i^{\parallel} \mathbf{v}_i^{\perp^\top} + \sum_{i=1}^{r} \mathbf{u}_i^{\perp} \mathbf{v}_i^{\parallel^\top} + \sum_{i=1}^{r} \mathbf{u}_i^{\perp} \mathbf{v}_i^{\perp^\top} \\
&= \mathbf{R}_* + \mathbf{R}_{\mathrm{L}} + \mathbf{R}_{\mathrm{R}} + \mathbf{R}_{\mathrm{E}} \\
&\approx \sum_{i=1}^{r} \mathbf{u}_i^{\parallel} \mathbf{v}_i^{\parallel^\top}
\end{aligned}
$$

Here $\mathbf{R}_*$ is fully spanned by the subspaces $\mathbf{U_A}$ and $\mathbf{U_B}$. The other matrices are sources of error as $\mathbf{R}_{\mathrm{L}}$'s right singular vectors are orthogonal to the subspace spanned by $\mathbf{U_B}$, and $\mathbf{R}_{\mathrm{R}}$'s left singular vectors are orthogonal to the subspace spanned by $\mathbf{U_A}$. Finally, $\mathbf{R}_{\mathrm{E}}$'s left *and* right singular vectors both are orthogonal to their respective similarity subspaces. The error contributed by this matrix into the estimation error of final recovered rating matrix is unavoidable, but is bounded by the results of Barjasteh et al. [6]. With this in mind, the ratings matrix is rewritten as:

$$
\mathbf{R} = \mathbf{u}_i^{\parallel} \mathbf{v}_i^{\parallel^\top} + \mathbf{R}_{\mathrm{L}} + \mathbf{R}_{\mathrm{R}} + \mathbf{R}_{\mathrm{E}} \approx \mathbf{u}_i^{\parallel} \mathbf{v}_i^{\parallel^\top}
$$

To begin the algorithm, a sub-matrix $\mathbf{M} \in \mathbb{R}^{p \times q}$ is extracted. To reconstruct $\mathbf{M}$ perfectly, the number of observed elements must be at least $\Omega(r(p+q)\log^2(2p))$ [73]. To meet these conditions, the rows are sorted by the number of ratings they contain. In this way the top rows are the least sparse. Then, the largest possible submatrix that meets the matrix completion conditions is taken for $\boldsymbol{m}$. This submatrix is then fully recovered as $\widehat{\mathbf{M}}$. This is accomplished via off-the-shelf matrix completion techniques of [83, 61, 81, 14]. This

leads to the following convex optimization, with $\Omega_{\mathbf{M}} \subseteq \Omega$ as the set of observed ratings:

$$\widehat{\mathbf{M}} = \arg \min_{\mathbf{X} \in \mathbb{R}^{p \times q}} \quad \|\mathbf{X}\|_* \\ \text{s.t.} \quad \mathbf{X}_{ij} = \mathbf{M}_{ij}, \ \forall \ (i,j) \in \Omega_{\mathbf{M}}. \tag{8.1}$$

This completed submatrix and the subspaces $\mathbf{U_A}$ and $\mathbf{U_B}$ can then be used to recover $\mathbf{R} = \sum_{i=1}^{r} \mathbf{u}_i \mathbf{v}_i^\top$ via transduction. Specifically, the rating information in the recovered matrix $\widehat{\mathbf{M}}$ is transduced to the cold-start users and items.

Because $\mathbf{u}_i^{\parallel}$ and $\mathbf{v}_i^{\parallel}$ are fully spanned by the subspaces $\mathbf{U_A}$ and $\mathbf{U_B}$, for $i = 1, 2, \ldots, r$ they can be rewritten as:

$$\mathbf{u}_i^{\parallel} = \mathbf{U_A} \mathbf{a}_i \text{ and } \mathbf{v}_i^{\parallel} = \mathbf{U_B} \mathbf{b}_i$$

Here $\mathbf{a}_i \in \mathbb{R}^s$ and $\mathbf{b}_i \in \mathbb{R}^s$ are the orthogonal projection of the singular vectors onto the corresponding subspaces. By substituting these equations into the decomposition of $\mathbf{R}_*$ the following is true:

$$\mathbf{R}_* = \sum_{i=1}^{r} \mathbf{U_A} \mathbf{a}_i \mathbf{b}_i^\top \mathbf{U_B}^\top = \mathbf{U_A} \left(\sum_{i=1}^{r} \mathbf{a}_i \mathbf{b}_i^\top \right) \mathbf{U_B}^\top$$

To recover the matrix $\mathbf{R}_*$, there must be an estimate for the vectors $\mathbf{a}_i, \mathbf{b}_i$. The subspaces extracted from the similarity matrices and the recovered rating sub-matrix $\widehat{\mathbf{M}}$ can be used to make these estimations.

To this end, first consider the decomposition of the recovered matrix as $\widehat{\mathbf{M}} = \sum_{i=1}^{r} \widehat{\mathbf{u}}_i \widehat{\mathbf{v}}_i^\top$.

The estimation of vectors $\mathbf{a}_i, \mathbf{b}_i$ and the matrix $\mathbf{R}_*$ now described. Let $\widehat{\mathbf{U}}_{\mathbf{A}} \in \mathbf{R}^{p \times s}$ be a random submatrix of $\mathbf{U_A}$ where the sampled rows correspond to the subset of rows in the matrix $\widehat{\mathbf{M}}$. Similarly, let $\widehat{\mathbf{U}}_{\mathbf{B}} \in \mathbf{R}^{q \times s}$ be created by sampling the rows of $\mathbf{U_B}$ corresponding to the columns in $\widehat{\mathbf{M}}$. An estimation of $\mathbf{a}_i, \mathbf{b}_i, i \in [r]$ vectors can be obtained by the

orthogonal projection of left and right singular vectors of $\widehat{\mathbf{M}}$ onto the sampled subspaces $\widehat{\mathbf{U}}_{\mathbf{A}}$ and $\widehat{\mathbf{U}}_{\mathbf{B}}$ by solving two optimization problems, namely:

$$\widehat{\mathbf{a}}_i = \arg\min_{\mathbf{a}\in\mathbb{R}^s} \left\| \widehat{\mathbf{u}}_i - \widehat{\mathbf{U}}_{\mathbf{A}}\mathbf{a} \right\|_2^2$$
$$\widehat{\mathbf{b}}_i = \arg\min_{\mathbf{b}\in\mathbb{R}^s} \left\| \widehat{\mathbf{v}}_i - \widehat{\mathbf{U}}_{\mathbf{B}}\mathbf{b} \right\|_2^2$$

The solutions to these ordinary-least squares regression problems are known. They are: $\left(\widehat{\mathbf{U}}_{\mathbf{A}}^\top \widehat{\mathbf{U}}_{\mathbf{A}}\right)^\dagger \widehat{\mathbf{U}}_{\mathbf{A}}^\top \widehat{\mathbf{u}}_i$ and $\left(\widehat{\mathbf{U}}_{\mathbf{B}}^\top \widehat{\mathbf{U}}_{\mathbf{B}}\right)^\dagger \widehat{\mathbf{U}}_{\mathbf{B}}^\top \widehat{\mathbf{v}}_i$, respectively. This allows $\mathbf{R}_*$ to be estimated by:

$$\begin{aligned}\widehat{\mathbf{R}}_* &= \mathbf{U}_{\mathbf{A}} \left(\sum_{i=1}^{r} \widehat{\mathbf{a}}_i \widehat{\mathbf{b}}_i^\top \right) \mathbf{U}_{\mathbf{B}}^\top \\ &= \mathbf{U}_{\mathbf{A}} \left(\widehat{\mathbf{U}}_{\mathbf{A}}^\top \widehat{\mathbf{U}}_{\mathbf{A}}\right)^\dagger \widehat{\mathbf{U}}_{\mathbf{A}}^\top \left(\sum_{i=1}^{r} \widehat{\mathbf{u}}_i \widehat{\mathbf{v}}_i^\top \right) \widehat{\mathbf{U}}_{\mathbf{B}} \left(\widehat{\mathbf{U}}_{\mathbf{B}}^\top \widehat{\mathbf{U}}_{\mathbf{B}}\right)^\dagger \mathbf{U}_{\mathbf{B}}\end{aligned}$$

The estimated rating matrix $\widehat{\mathbf{R}}$ is found by now setting $\widehat{\mathbf{R}} = \widehat{\mathbf{R}}_*$. There is dimension reduction via the submatrices, and this leaves only the spectral clustering and orthogonal projection as computationally expensive– even with a large rating matrix.

8.3 Experimental Evaluation

Several experiments were completed on multiple datasets. These compared DecRec over a set of baseline algorithms to demonstrate the merits and advantages of DecRec. The datasets are well-known and publicly available. These are: MovieLens (1M and 100K) [2], Epinions [3] and NIPS [4].

[2] `http://www.grouplens.org/node/73`
[3] `http://www.trustlet.org/wiki/Epinions\_dataset`
[4] `http://www.cs.nyu.edu/~roweis/data.html`

Several fundamental questions were addressed. Principally, the proposed algorithm is compared against state-of-the art methods for incorporating side information into recommendation of existing items to existing users. As this is a well researched area, the interesting results occur during the cold-start cases. Results are presented when there are cold-start items, cold-start users, and both cold-start users *and* items.

8.3.1 Datasets

Although this algorithm is made to provide recommendations, it is applicable to any user-item setting. Because of this, experiments were performed on well-known movie and item rating datasets for recommendation, and a further experiment was conducted on an author-publication graph to solve a network completion problem. The recommendation (including cold-start) experiments used MovieLens and Epinions while the NIPS dataset was used for network completion. The descriptions are given along with a table of the datasets' statistics in Table 8.1. There is also a synthetic dataset included in some of the experiments.

1. **MovieLens:** Two versions of the MovieLens datasets were used. They were the 100K and 1M variants. They consist of ratings (1-5) from users on movies. In addition to rating data, these datasets also contain features for both users and movies. For each movie the features (e.g title, year, genre) are given. Further feature data was extracted from the movie information website imdb.com. For each user, personal features (e.g. gender, age, occupation, location) were extracted. Then for both users and items we computed their cosine similarities to be used as side information.

2. **Epinions:** This dataset was obtained from a user-oriented product review website that has a trust network of users. Users can specify whether they trust other users or not explicitly. This trust network allows the creation of a 0/1 trust connectivity vector for each user with all other users. From this, the cosine similarity of trust

vectors was computed and became the similarity matrix of users. The items have categorical side information and the users can provide their ratings (1-5).

3. **NIPS:** For network completion, the co-author network at the NIPS conference [78] was taken. From this, the paper-author and paper-word matrices were extracted. There is no side information for the authors, but the papers' contents are pre-processed such that all words are converted to lower cases and stop-words are removed. Again, the cosine similarity is computed of the words between papers.

4. **Synthetic:** The synthetic dataset was generated by two matrices $\mathbf{U} \in [0,1]^{4{,}000 \times r}$ and $\mathbf{V} \in [0,1]^{2{,}000 \times r}$. Then $\mathbf{U}$ and $\mathbf{V}$ were used to generate a rating matrix $\mathbf{R}^{4{,}000 \times 2{,}000} = \mathbf{U}\mathbf{V}^\top$. In $\mathbf{R}$, there were 4,000 users and 2,000 items. A similarity matrix $\mathbf{A}^{4{,}000 \times 4{,}000} = \mathbf{U}\mathbf{U}^\top$ was generated for users and also $\mathbf{B}^{2{,}000 \times 2{,}000} = \mathbf{V}\mathbf{V}^\top$ for items. Finally, random noise was added to the all elements of $\mathbf{A}$ and $\mathbf{B}$ where the noise follows the Gaussian distribution with $\mathcal{N}(0, 0.5)$.

8.3.2 Evaluation Metrics

As is standard, the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) metrics for prediction accuracy [38] were again used. If $\mathcal{T}$ denotes the set of ratings to be predicted, i.e., $\mathcal{T} = \{(i,j) \in [n] \times [m], \mathbf{R}_{ij} \text{ needs to be predicted}\}$, and $\widehat{\mathbf{R}}$ denotes the prediction matrix obtained by a recommendation algorithm, then MAE is:

$$\text{MAE} = \frac{\sum_{(i,j)\in\mathcal{T}} |\mathbf{R}_{ij} - \widehat{\mathbf{R}}_{ij}|}{|\mathcal{T}|}$$

RMSE is similarly defined as:

$$\text{RMSE} = \sqrt{\frac{\sum_{(i,j)\in\mathcal{T}} \left(\mathbf{R}_{ij} - \widehat{\mathbf{R}}_{ij}\right)^2}{|\mathcal{T}|}}$$

Statistics	MovieLens 100K	MovieLens 1M	Epinioins	NIPS
Number of users	943	6,040	8,577	2,073
Number of items	1682	3,706	3,769	1,740
Number of ratings	100,000	1,000,209	203,275	3,990
Range of ratings	1 - 5	1 - 5	1 - 5	0-1

Table 8.1 Statistics of the real world datasets

Even small improvements in RMSE are considered valuable in the context of recommender systems. For example, the streaming video service Netflix, that relies heavily on predicting users' ratings on content, offered a prize of $1,000,000 to the first researchers to achieve a 10% reduction of RMSE.

There is one other metric to be considered. Given an item i, let r_i be the relevance score of the item ranked at position i, where $r_i = 1$ if the item is relevant to the i and $r_i = 0$ otherwise. The NDCG measure is a normalization of the Discounted Cumulative Gain (DCG) measure. DCG is a weighted sum of the degree of relevancy of the ranked users. The value of NDCG is between $[0, 1]$ and at position k is defined as:

$$\text{NDCG@}k = Z_k \sum_{i=1}^{k} \frac{2^{r_i} - 1}{\log(i+1)}$$

In all experiments, the value of k is set as the number of rated items by each user.

8.3.3 The Baseline Algorithms

To evaluate the performance of the proposed DecRec algorithm, a wide variety of baseline algorithms were deployed. The baseline algorithms were chosen from four types of categories: state-of-the-art algorithms for rating predictions, recommenders with cold-start items capacity, recommenders with cold-start users capacity, and algorithms with the

capacity to consider both cold-start items *and* users. The MyMediaLite implementations were used throughout the testing [29].

1. **RS** (Random Strategy) [51]: A simple baseline that selects at random a subset of users or items. The recommendation for cold-start users and items is a challenging case, where RS is one of the baseline methods.

2. **U-KNN** (User KNN) [45]: Predicts the ratings using the similarity with the K nearest neighbors where users have weights.

3. **I-KNN** (Item KNN) [45]: Is a weighted item-based KNN approach for rate prediction.

4. **GA** (Global Average): Uses the average of ratings over all ratings.

5. **I-A** (Item Average): Uses the average rating of an item for its prediction.

6. **U-A** (User Average): Uses the average rating of a user for its prediction.

7. **SO** (Slope One) [47]: Pre-computes the average difference between two items that are rated by users. SO is a frequency weighted slope one rating prediction.

8. **BPSO** (BiPolar Slope One) [47]: Is a Bi-polar frequency weighted slope one rating prediction.

9. **SMF** (Social Matrix Factorization) [39]: Is a matrix factorization algorithm that incorporates the social network for prediction.

10. **CC** (CoClustering) [30]: Is a weighted co-clustering algorithm that involves simultaneous clustering of users and items.

11. **LFLLM** (Latent Feature Log Linear Model) [59]: Is a scalable log-linear model that exploits the side information for dyadic prediction.

12. **U-I-B** (User Item Baseline) [45]: Is a rating prediction method that uses the average rating value along with a regularized user and item bias.

13. **FWMF** (FactorWise Matrix Factorization): Is a matrix factorization based model with a factor-wise learning.

14. **BMF** (Biased Matrix Factorization) [75]: Is a matrix factorization that learns by stochastic gradient descent with explicit bias for users and items.

15. **SVDPP** (SVD++) [44]: Is a matrix factorization that takes into account what users have rated and directly profiles users and items.

16. **SSVDPP** (Sigmoid SVD++) [44]: Is a version of SVD++ that uses a sigmoid function.

17. **SU-AFM** (Sigmoid User Asymmetric Factor Model) [69]: Is an asymmetric factor model that represents the items in terms of those users that rated them.

18. **SI-AFM** (Sigmoid Item Asymmetric Factor Model) [69]: Is an asymmetric factor model that represents users in terms of the items they rated.

19. **SCAFM** (Sigmoid Combined Asymmetric Factor Model) [69]: Is an asymmetric factor model that represents items in terms of the users that rated them, and users in terms of the items they rated.

20. **CBF** (Content Based Filtering) [79]: This algorithm builds a profile for each user based on the properties of the user's preferred items from the past.

21. **LCE** (Local Collective Embeddings) [79]: Is a matrix factorization method that exploits properties of items and past user preferences while enforcing the manifold structure exhibited by the collective embeddings.

22. **LCE-NL**(Local Collective Embeddings No Laplacian) [79]: Is LCE without laplacian regularization.

23. **ELCE** (Extended LCE): Is an extended version of LCE meant to handle the challenge of the presence of both cold-start users and items simultaneously.

24. **KMF** (Kernelized Matrix Factorization) [90]: Is a matrix completion based algorithm, which incorporates external side information of the users or items into the matrix factorization process.

25. **DecRec**: The proposed algorithm with decoupled completion and transduction.

8.3.4 Effects of Noise

The sensitivity to noise on the proposed algorithm is explored. With the synthetic dataset, recall $\mathbf{A}$ and $\mathbf{B}$ as the two similarity matrices between users and items, respectively. The variance of the Gaussian were varied by every 0.05 for 0.05 to 0.95. Figure 8.2 shows the increase in RMSE and MAE of the results on test data as the noise increases. However, the degradation of the results is graceful with respect to the incease of noise. The results are similar to those in a real world dataset.

Examining MovieLens 1M, noise was generated that also follows a Gaussian distribution. This noise, $\mathcal{N}(0, 0.5)$, was added to the similarity matrices. The variance was varied in the same way as in the synthetic dataset. Figure 8.3 shows the RMSE and MAE of the predictions. Similar to the previous results, by adding noise to the similarity matrices, the error grows gracefully.

8.3.5 Existing Users and Items

The standard case for recommender systems is that all users and items have a history of rating and being rated, respectively. Generally, users only rate a small number of items and the rest of the ratings are unknown. The stated goal of such recommender systems is to provide a prediction for all of the ratings of the unrated items. There are two

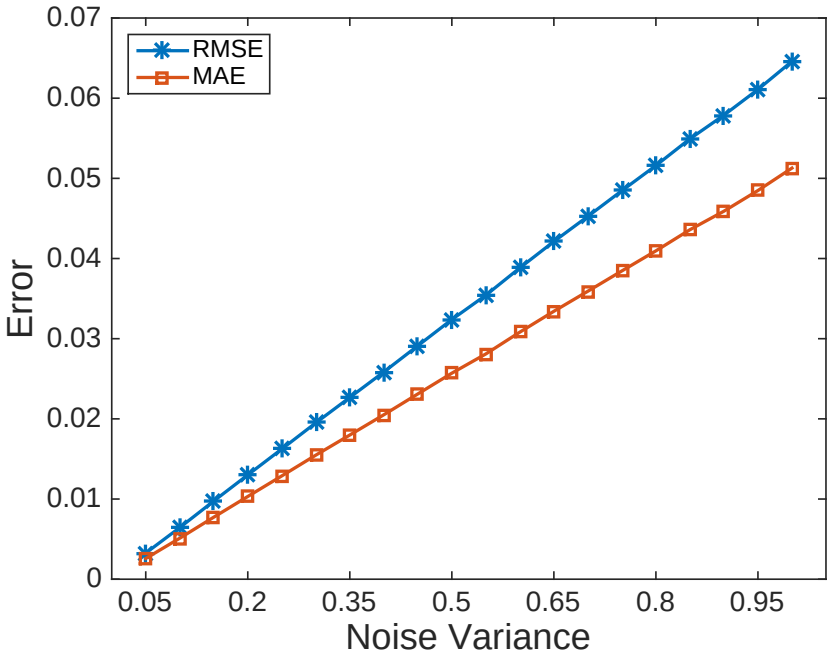

Figure 8.2 RMSE & MAE on the synthetic dataset for different noise variances on similarity matrices

broad approaches to this problem: neighbor-based and latent factor models. Many baseline algorithms from each type were used.

DecRec was deployed on the MovieLens 100K, MovieLens 1M and Epinions datasets. GroupLens Research [5] made five sets available, which are 80%/20% splits of the MovieLens 100K into training and test data. For MovieLens 1M and Epinions, the data were split into 80%/20% train-test sets randomly five times (for 5-fold cross-validation). The average results are reported. Table 8.2 shows this average RMSE and MAE resulting from the 5-fold cross-validation for all baseline algorithms. DecRec had the smallest RMSE and MAE values for each column (indicated by bold type-face). The results suggested that among neighbor-based approaches, I-KNN is the best-performing algorithm for MovieLens 1M and 100K and U-KNN is the second best-performing one. That is because of the similarity between movies (genre, director, etc) and similarity between taste of users play an important role in prediction accuracy. I-KNN and U-I-B outperformed other neighbor-based methods

[5] grouplens.org/

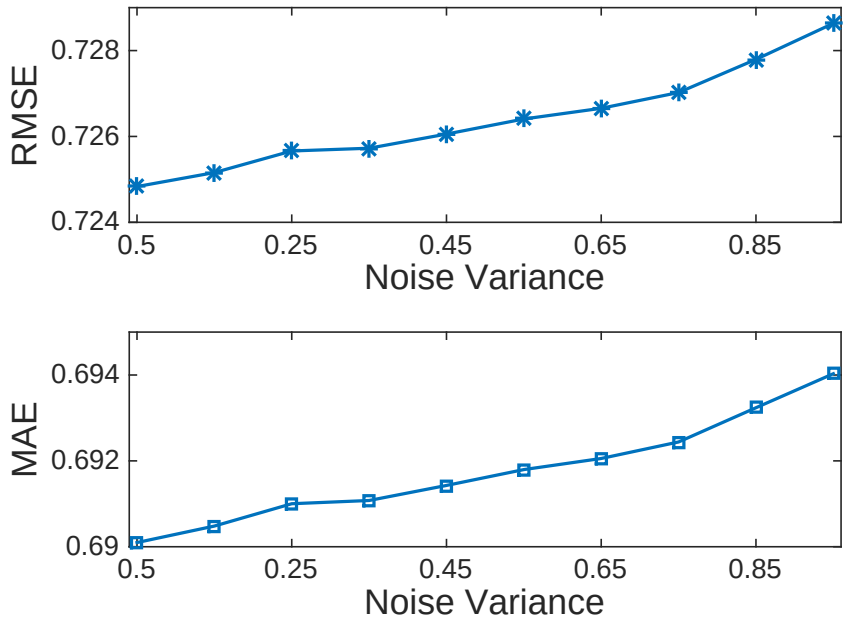

Figure 8.3 RMSE & MAE of MovieLens 1M for different noise variances on similarity matrices

on Epinions in respect to RMSE measures, while BPSO and I-KNN outperformed others in respect to MAE measures. Similarity of items and having the same pattern of ratings for similar items on Epinions helped I-KNN's results perform better than other neighbor based methods. Table 8.3 further shows the comparison between latent factor methods in which DecRec (KMF) algorithm achieved the first (second) best performance of RMSE and MAE for MovieLens 100K and RMSE for MovieLens 1M. It also achieved the second (first) best performance of MAE for MovieLens 1M. On Epinions, DecRec also outperformed other latent factor methods.

Tables 8.2 and 8.3 show that DecRec achieved the best performance for all datasets among all methods of both categories, latent factor and neighbor based methods (except for MAE on MovieLens 1M), confirming the performance advantage of DecRec over all baseline algorithms. Hence, the proposed decoupled method by incorporating side information reveals the need for preventing error propagation along with using side information to obtain more accurate predictions.

	Algorithms	Hyperparameters	MovieLens 100K		MovieLens 1M		Epinions	
			RMSE	MAE	RMSE	MAE	RMSE	MAE
Neighbor-Based Methods	GA	—	1.1190	0.9399	1.116	0.9327	1.1692	0.8878
	I-A	—	1.0220	0.8159	0.9759	0.7790	1.0695	0.8140
	U-A	—	1.0390	0.8350	1.034	0.8272	1.1276	0.8769
	U-KNN	k=80	0.9355	0.7398	0.8952	0.7030	2.3999	2.2229
	I-KNN	k=80, sh=10,λ_u=25, λ_v=10	0.9241	0.7270	0.8711	0.6830	1.0279	0.6993
	U-I-B	λ_u=5, λ_v=2	0.9419	0.7450	0.9081	0.7190	1.0290	0.8010
	SO	—	0.9397	0.7403	0.9020	0.7120	1.0865	0.7067
	BPSO	—	0.9744	0.7482	0.9390	0.7199	1.0449	0.6813
	CC	C_i=3, C_u=3, T=30	0.9559	0.7526	0.9118	0.7134	1.0573	0.7890

Table 8.2 Results on MovieLens 100K and 1M and Epinions for neighbor-based methods with no cold-start users/items

	Algorithms	Hyperparameters	MovieLens 100K		MovieLens 1M		Epinions	
			RMSE	MAE	RMSE	MAE	RMSE	MAE
Latent Factor Models	SMF	p=10, λ_u=0.015, λ_v=0.015, λ_b=0.01 λ_s=1, η=0.01, η_b=1, T=30	1.0134	0.7884	1.2284	0.9315	1.1224	0.8436
	FWMF	p=5, T=5, sh=150	0.9212	0.7252	0.8601	0.6730	1.5090	1.0597
	BMF	p=160, λ_b=0.003, η=0.07, T=100 λ_u=0.08,λ_v=0.1	0.9104	0.7194	0.8540	0.6760	1.0240	0.7918
	MF	p=10, T=75 λ_g=0.05, η=0.005	0.9133	0.7245	0.8570	0.6751	1.0908	0.8372
	KMF	σ_r=0.4, D=10, η=0.003, γ=0.1	0.7947	0.6893	0.7492	**0.6514**	0.9015	0.7873
	LFLLM	p=10, λ_b=0.01, λ_u=0.015, λ_v=0.015 η=0.01, T=30, η_b=1	0.9550	0.7617	0.9012	0.7082	1.2891	1.0386
	SI-AFM	η_b=0.7, λ_g=0.015, η=0.001, p=10 λ_b=0.33, T=1	0.9568	0.7628	1.035	0.8488	1.1534	0.8816
	SU-AFM	η_b=0.7,λ_g=0.015 λ_b=0.33, T=1, η=0.001, p=10	0.9569	0.7634	0.9062	0.7189	1.069	0.8398
	SCAFM	—	0.9499	0.7559	0.9121	0.7239	1.0600	0.8312
	SVDPP	η_b=0.07,λ_g=1 λ_b=0.005, p=50, η=0.01, T=50	0.9065	0.7135	0.8510	0.6680	1.0550	0.8220
	SSVDPP	η_b=0.7, T=30 p=10,λ_g=0.015, η=0.001, λ_b=0.33	1.185	0.9147	0.9402	0.7352	1.3328	0.9022
	RS	—	1.6960	1.3860	1.7070	1.3940	1.9096	1.5789
	DecRec	r=10	**0.7002**	**0.6628**	**0.7157**	0.6721	**0.7157**	**0.6796**

Table 8.3 Results on MovieLens 100K and 1M and Epinions for latent factor methods with no cold-start users/items

8.3.6 Cold-Start Items

To simulate cold-start item problems, the items were divided into two disjoint training and test subsets. Here, 80% of the items were considered existing items for training and the remaining 20% were cold-start items for testing.

This general framework can also be used on network completion challenges. The NIPS dataset was chosen to not only simulate the cold-start item scenario, but also to show the results of DecRec for network completion. NIPS has rich side information for the items (papers) and shows the relationship (0 or 1) between papers and authors. Because the values in NIPS are either 0 or 1, predicting the authors of new papers can be also perceived as a link prediction problem.

Four competitive recommendation methods were considered on NIPS dataset. They were CBF, KMF, LCE and LCE-NL. Table 8.4 shows the average RMSE and MAE of 5-fold cross-validation for these algorithms for the cold-start item scenario. The parameters from MyMediaLite are provided for reproduciblility.

These results indicate that DecRec is the best performing algorithm among all baseline algorithms in the cold-start item scenario with respect to RMSE, MAE and NDCG. Having the highest NDCG value among all competitive algorithms shows that DecRec can present the top-ranked items to users better than other algorithms. DecRec also yields the lowest RMSE and MAE values. From this, it may be concluded that DecRec's predictions of the ratings for cold-start items are more accurate than the other methods. DecRec can better suggest the top-ranked cold-start items to users with higher accuracy. The running times are also included in this instance. CBF must only create user profiles and is thus very fast. KPMF and DecRec are acceptably efficient, but LCE and LCE-NL are much slower due to their convergence conditions.

Datasets	Method	Hyperparameters	Measures			
			NDCG@k	RMSE	MAE	Time(s)
	CBF	—	0.3861	0.7943	0.8881	**0.17597**
	LCE	k=500, λ_g=0.5, ε = 0.001, T_m=500, β=0.05	0.4240	0.7692	0.8675	709.869
NIPS	LCE-NL	k=500, λ_g=0.5, ε=0.001, T_m=500, β=0	0.4186	0.7532	0.8562	1823.48
Cold-start item	KMF	σ_r=0.4, D=10, η=0.003, γ=0.1	0.1415	0.8804	0.9196	19.5413
	DecRec	r=1000	**0.4626**	**0.5111**	**0.6805**	23.8410
	CBF	—	0.2201	0.6644	0.7741	**4.4800**
	LCE	k=500, λ_g=0.5, ε=0.001, T_m=500, β=0.05	0.2327	0.6713	0.7786	1067.11
Epinions	LCE-NL	k=500, λ_g=0.5, ε=0.001, T_m=500, β=0	0.2319	0.6712	0.7785	11969.9
Cold-start user	KMF	σ_r=0.4, D=10, η=0.003, γ=0.1	0.2084	0.8522	0.8882	1196.32
	DecRec	r=1063	**0.2343**	**0.6618**	**0.7716**	144.660
	RS	—	0.1022	1.1981	0.9782	**0.0131**
MovieLens 100K	KMF	σ_r=0.4, D=10, η=0.003, γ=0.1	0.2423	0.9823	0.8730	11.540
Cold-start	ELCE	k=500, λ_g=0.5, ε=0.001, T_m=500, β=0	0.2681	0.8934	0.7626	16.4683
user & item	DecRec	r=100	**0.2641**	**0.8672**	**0.7230**	4.8729
	RS	—	0.0652	1.3820	0.9326	**0.0682**
MovieLens 1M	KMF	σ_r=0.4, D=10, η=0.003, γ=0.1	0.1834	0.9730	0.8442	63.732
Cold-start	ELCE	k=500, λ_g=0.5, ε=0.001, T_m=500, β=0	0.2662	0.8849	0.7684	166.201
user & item	DecRec	r=100	**0.2783**	**0.8524**	**0.7162**	10.578

Table 8.4 Results on all of the cold-start scenarios for real datasets

8.3.7 Cold-start Users

To simulate cold-start user problems, the users were divided into two disjoint training and test subsets. Here, 80% of the users were considered existing users for training and the remaining 20% were cold-start users for testing. To show the relative results of DecRec, it was compared with several competitive algorithms: CBF, LCE, LCE-NL and KPMF.

Because Epinions has an explicitly given trust network among the users, it has very useful side information. This set is used to experiment on the cold-start user scenario. Table 8.4 shows the averaged (5-fold cross validation) performance results of the mentioned algorithms. DecRec achieved the best performance of NDCG, RMSE and MAE on the Epinions dataset.

These evaluations of DecRec and the baseline algorithms are close, but the consistency in outperforming other competitive methods on both cold-start user and item scenarios confirms the stability and performance advantage of DecRec over state-of-the-art algorithms.

8.3.8 Cold-Start Users and Items

To simulate handling both cold-start users and items, 20% of the users and 20% of the items where randomly chosen to be cold-start users and items, respectively. All experiments were performed on the two MovieLens variants because they have rich side information about both the users and the items. Both the ratings for cold-start users and items were then predicted. This is a very challenging scenario because of all of the completely empty rows and columns in the rating matrix. Very few baseline algorithms exists that can even attempt a solution. DecRec was compared only with only RS, KMF, ELCE because of this restriction.

Table 8.4 shows the results of applying RS, KMF, ELCE and DecRec algorithms on MovieLens 100K and 1M. On both datasets, DecRec outperformed other baselines. The nearest competitor was ELCE– a collaborative factorization method.

8.4 Conclusion

DecRec explicitly exploits the similarity information about users and items to alleviate cold-start problems. In particular, DecRec decouples the completion from the knowledge transduction, thus preventing some error propagation as described. Experimental results on real datasets clearly indicated that DecRec outperforms modern benchmark algorithms in all of the permutations of the cold-start problems, and even performs well when no cold-start users/items are present. Along with the dearth of algorithms that can be applied to cold-start problems, this positions DecRec as one of the best recommendation algorithms for cold-start scenarios.

Chapter 9

Conclusions

This work presented a new framework to study invariant evolution with the standard and random $\mathcal{P}$-impact process. Several invariants were explored and their impact in trees and empty graphs was shown. The process of determining maximum distance-impact edges was shown to be non-trivial through a series of counterexamples and a proof that these edges must not be incident to two leaves in trees. Further experimentation seems to indicate that some edges provide more accuracy when they are added before matrix factorization is applied.

An effective algorithm for network completion with auxiliary similarity information about nodes was also developed. Its comparison to the state-of-the-art baselines on synthetic and real datasets reveals that proposed method exhibits improved performance in terms of recovering the full network. This advantage is brought by the process in which we decoupled the completion of observed submatrix from transduction of knowledge by exploiting similarity information.

Finally, an algorithm that explicitly exploits the similarity information about users and items to alleviate cold-start problems for recommendation was given. Similar to the network completion algorithm, it completes sub-matrix of the rating matrix and transducts knowledge from recovered sub-matrix along with the side information of the users and items.

This algorithm also performed well in experimentation with existing users/items and in all three cold-start scenarios.

APPENDIX

APPENDIX

This appendix contains the results of the additional experimentation in Chapter 6. Five order 25 random graphs of each class were generated. From each of these, nine other graphs were created. In each of these nine graphs, between 10-90% (step size of 10%) of the entries in the adjacency matrix corresponding to potential edges were eliminated. Note that for non-directed graphs the adjacency matrix is symmetric, so in practice only the upper-triangle adjacency matrix was considered. From each of these adjacency matrices two processes occurred. First, the adjacency matrix was completed as-is via matrix factorization. Second, one entry at a time was revealed and replaced with its true value (0 or 1). At this point, the adjacency matrix was completed (again via matrix factorization). In the first case, the basal error rates were calculated (i.e. error without revealing any edges). In the second case, each time an entry was revealed the error of completion was reported as in instance. For the ease of language, in the original graph both absent and present edges will be referred to as 'edges'. The difference being, absent 'edges' will have a true value of 0 in the adjacency matrix while present edges will have a value of 1.

The predictive power of individual edges are found by computing the completion error that their addition yields for the resultant adjacency matrix. The basal error rate is also reported. For all random graphs of the form $G(25, p)$ for all $0.1 \leqslant p \leqslant 0.9, p \neq 0.5$ step size 0.1 refer to Figures A.1-15. The tables comparing the basal RMSE and MAE and the percent of edges who's addition degrades the matrix factorization results below the basal rate are given in Tables A.1-8 for $G(25, p)$ with $0.1 \leqslant p \leqslant 0.9, p \neq 0.5$ step size 0.1, respectively. For $G(25, 0.5)$, refer to Figures 6.1, 6.2, and 6.1 in Chapter 5 for the RMSE, MAE, and basal comparison results, respectively.

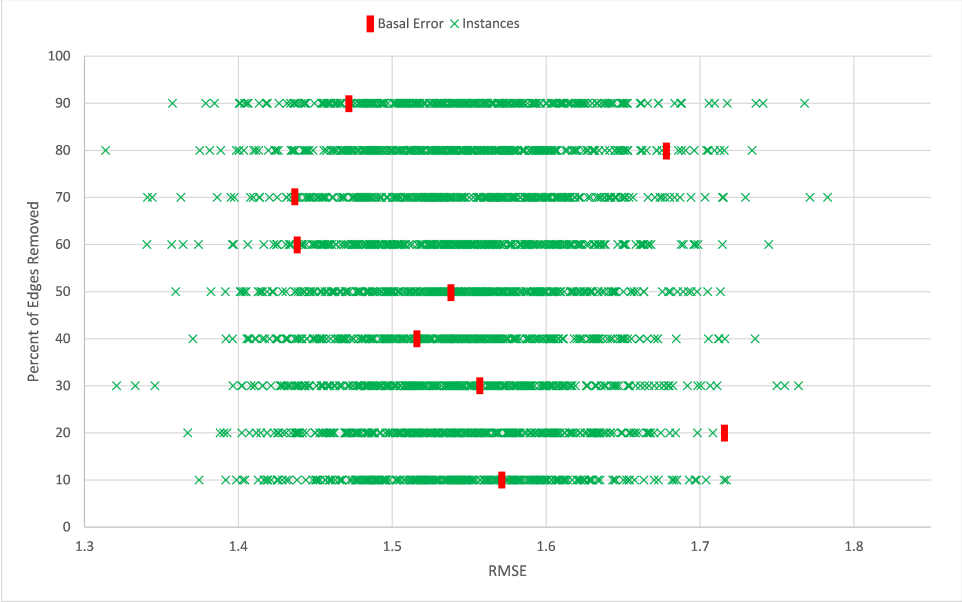

Figure A.1 RMSE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.1$)

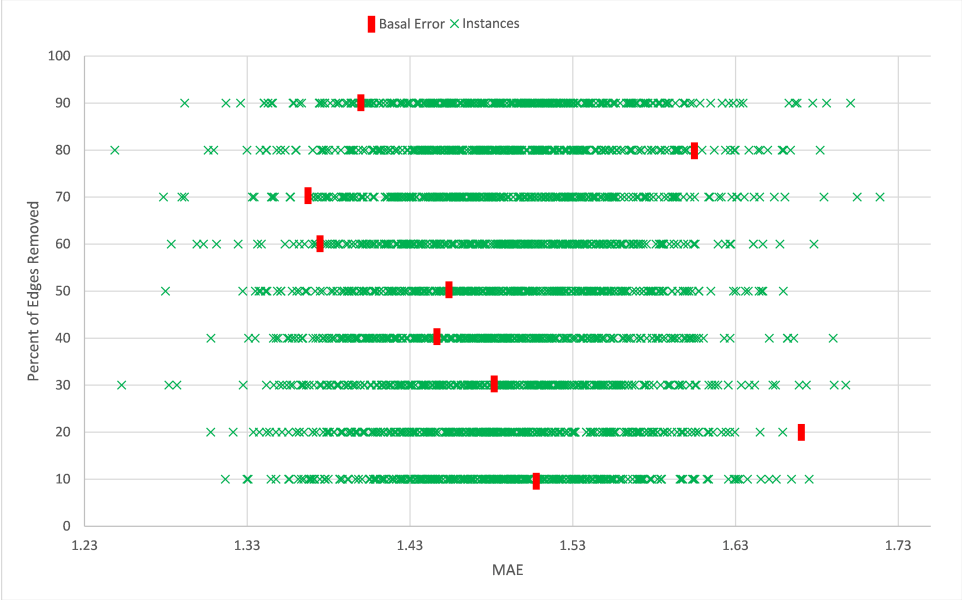

Figure A.2 MAE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.1$)

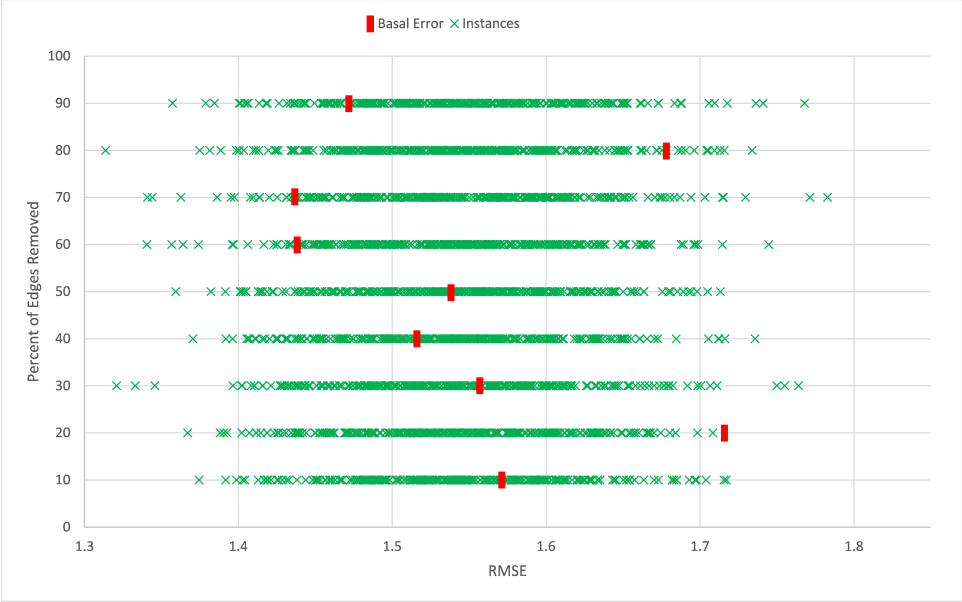

Figure A.3 RMSE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.2$)

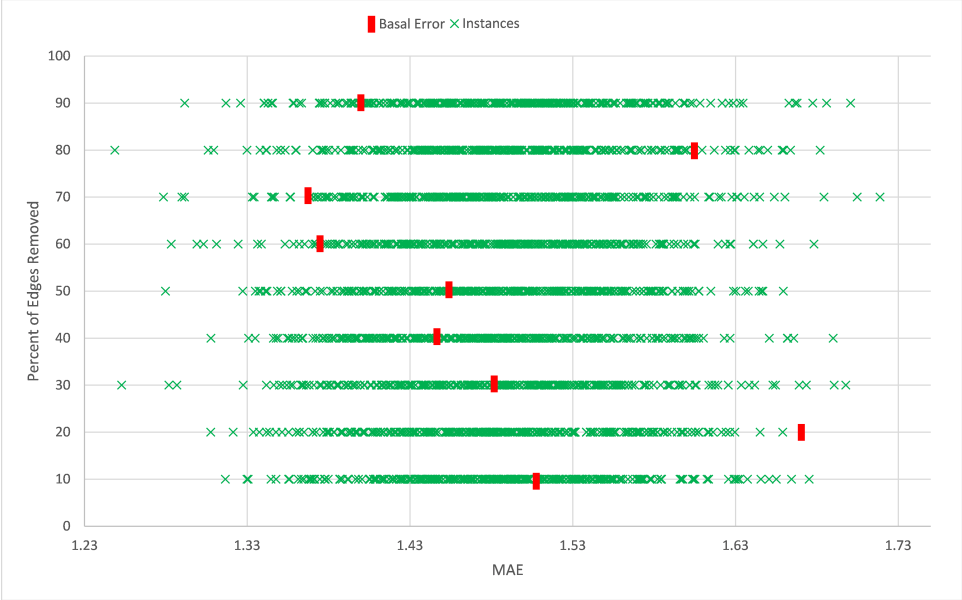

Figure A.4 MAE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.2$)

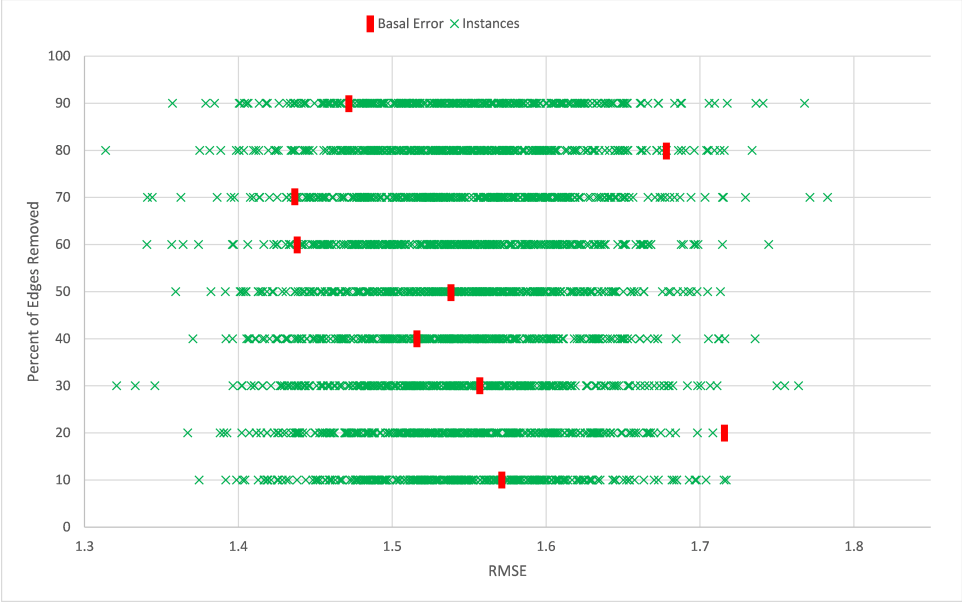

Figure A.5 RMSE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.3$)

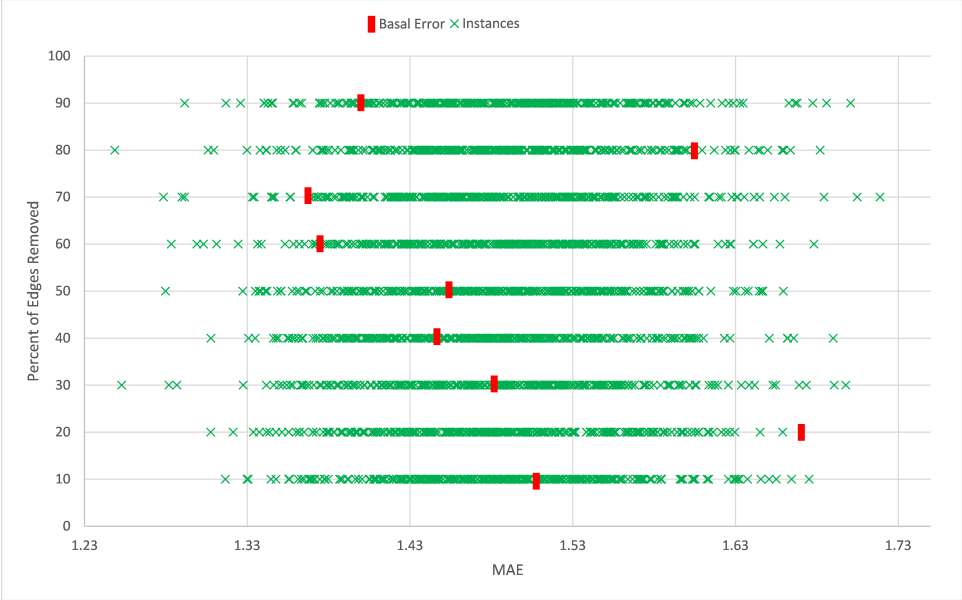

Figure A.6 MAE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.3$)

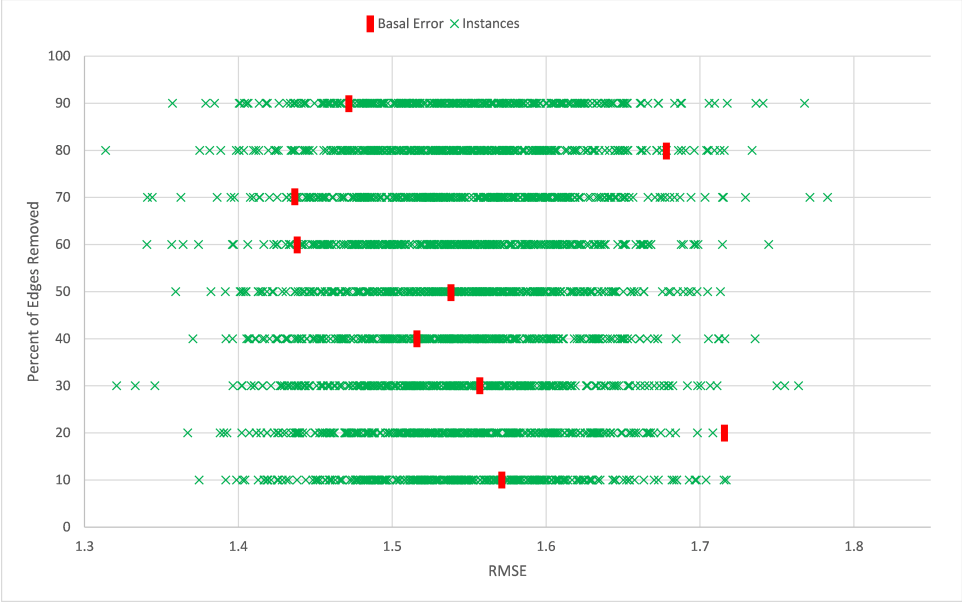

Figure A.7 RMSE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.4$)

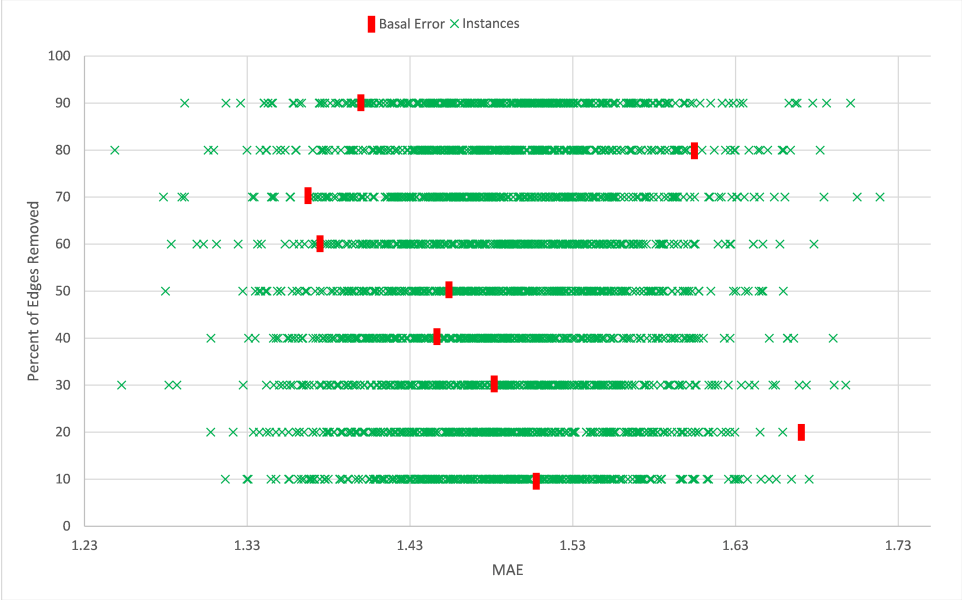

Figure A.8 MAE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.4$)

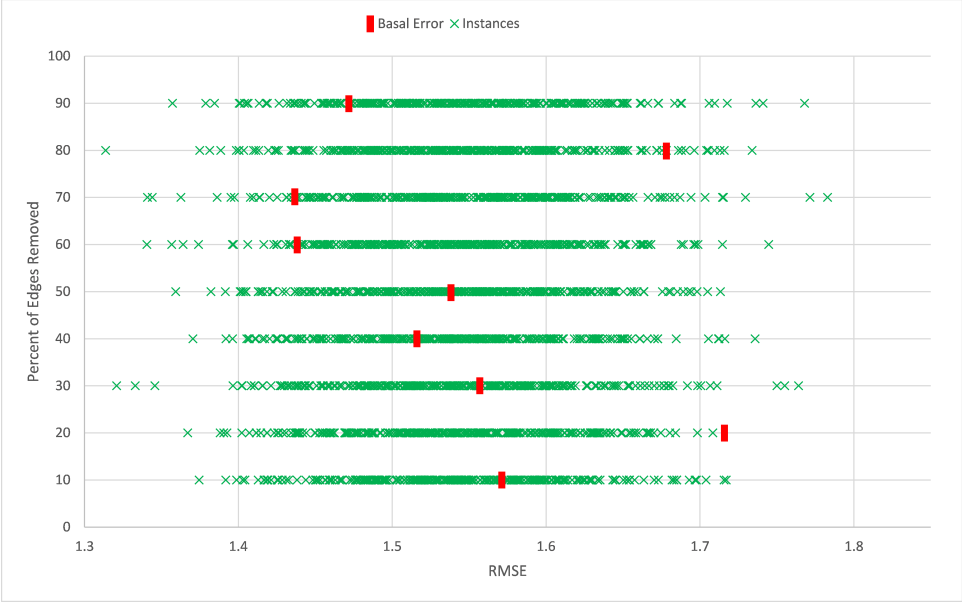

Figure A.9 RMSE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.6$)

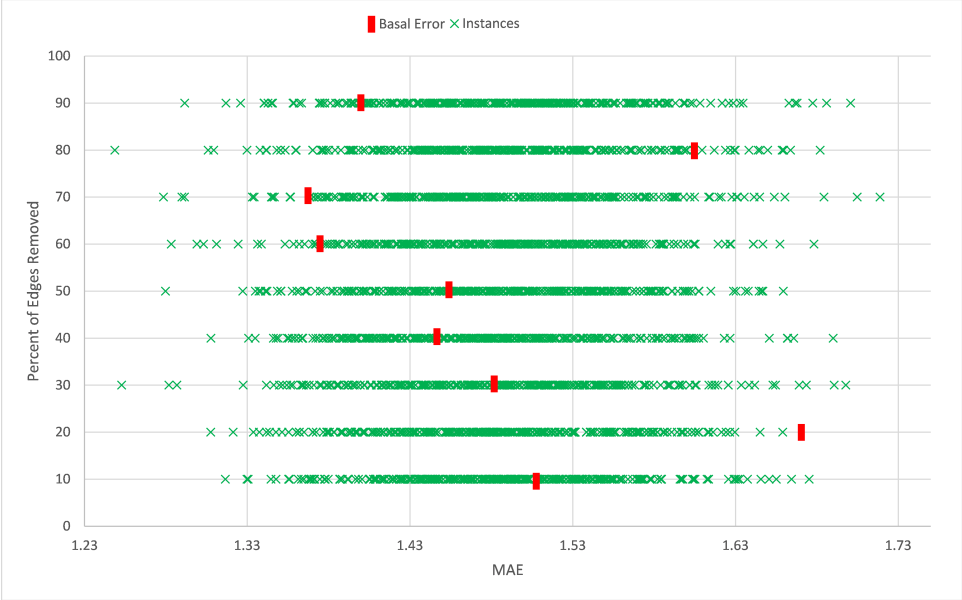

Figure A.10 MAE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.6$)

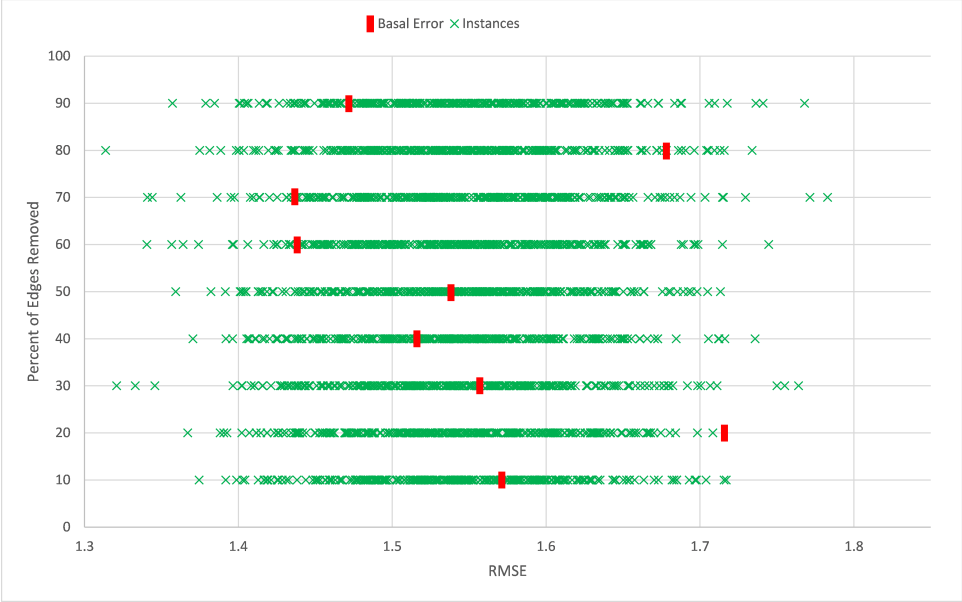

Figure A.11 RMSE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.7$)

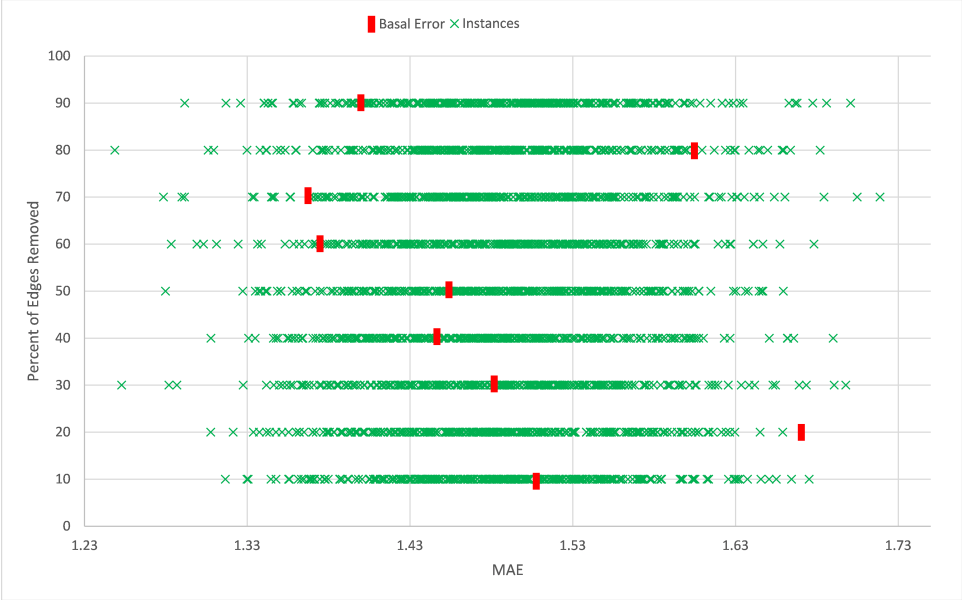

Figure A.12 MAE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.7$)

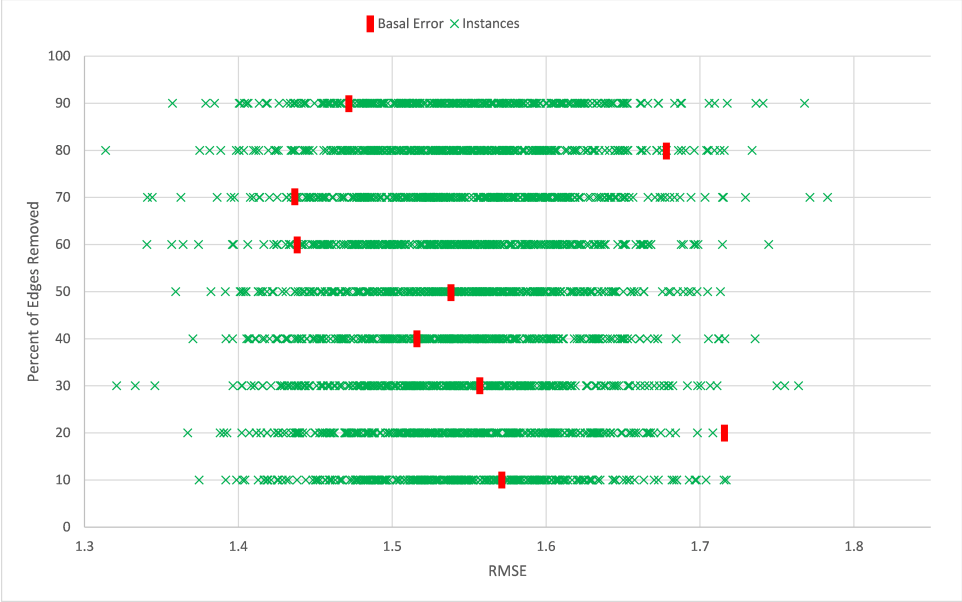

Figure A.13 RMSE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.8$)

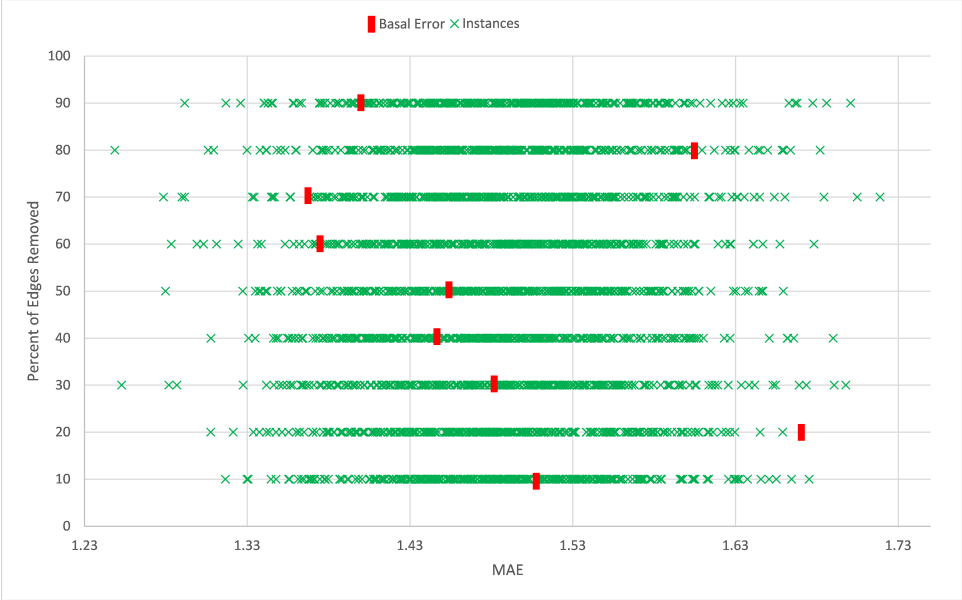

Figure A.14 MAE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.8$)

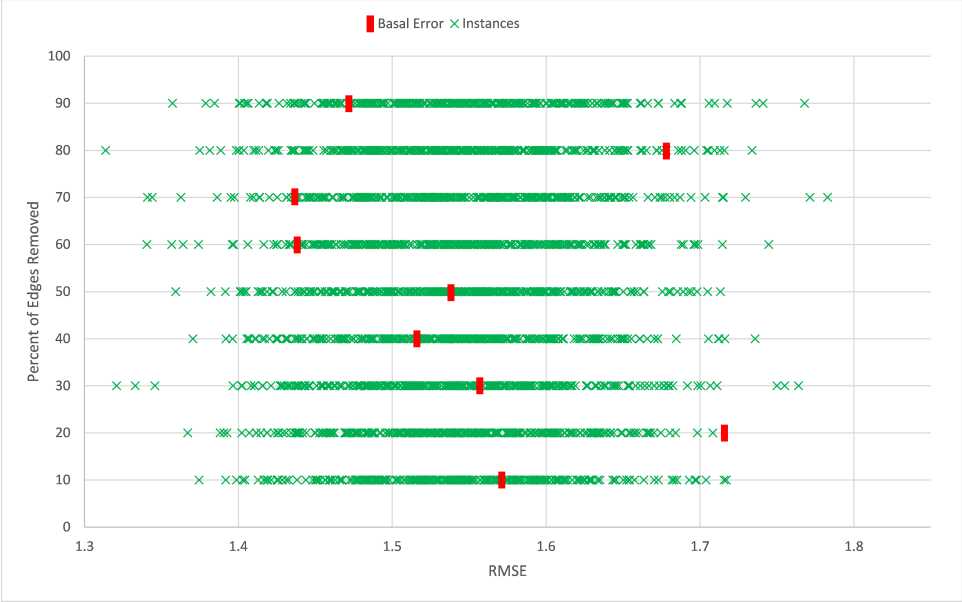

Figure A.15 RMSE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.9$)

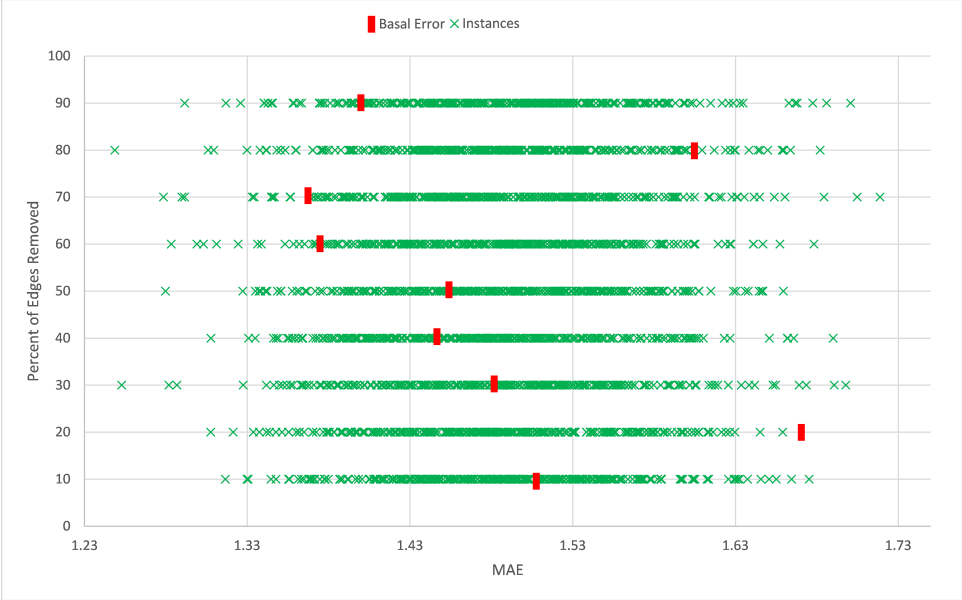

Figure A.16 MAE on single edge addition for varied percentages of removed edges for random graphs ($p = 0.9$)

Edges Removed (%)	Basal RMSE	Worse Edges (%)	Basal MAE	Worse Edges (%)
10%	1.5763	78.33	1.5146	79.66
20%	1.7159	100.0	1.6573	100.0
30%	1.5616	62.0	1.4868	54.0
40%	1.5273	45.33	1.4555	41.66
50%	1.5427	54.33	1.4849	57.66
60%	1.4446	12.66	1.3757	11.33
70%	1.4411	9.666	1.3682	8.666
80%	1.6711	98.33	1.6178	99.0
90%	1.4727	14.33	1.4069	14.33

Table A.1 The basal completion error and the percent of edges that produce higher error for all percentages of edges removed for Erdős-Rényi random graphs of the form $G(25, 0.1)$

Edges Removed (%)	Basal RMSE	Worse Edges (%)	Basal MAE	Worse Edges (%)
10%	1.4106	24.66	1.3127	22.66
20%	1.3668	10.33	1.2707	10.0
30%	1.5186	81.0	1.4315	83.33
40%	1.4561	53.0	1.3610	50.66
50%	1.4525	46.66	1.3597	46.66
60%	1.4615	56.00	1.3687	55.33
70%	1.4758	60.0	1.3676	50.33
80%	1.4231	30.0	1.3252	26.66
90%	1.5231	80.0	1.4284	77.0

Table A.2 The basal completion error and the percent of edges that produce higher error for all percentages of edges removed for Erdős-Rényi random graphs of the form $G(25, 0.2)$

Edges Removed (%)	Basal RMSE	Worse Edges (%)	Basal MAE	Worse Edges (%)
10%	1.5350	91.66	1.4448	92.33
20%	1.5463	93.66	1.4490	92.33
30%	1.5396	88.0	1.4323	86.33
40%	1.5079	78.0	1.4160	78.33
50%	1.5263	86.33	1.4275	83.0
60%	1.4074	28.33	1.3015	25.0
70%	1.4692	61.0	1.3833	64.66
80%	1.6101	98.0	1.5218	97.33
90%	1.4333	39.33	1.3404	40.0

Table A.3 The basal completion error and the percent of edges that produce higher error for all percentages of edges removed for Erdős-Rényi random graphs of the form $G(25, 0.3)$

Edges Removed (%)	Basal RMSE	Worse Edges (%)	Basal MAE	Worse Edges (%)
10%	1.2241	10.33	1.0778	8.666
20%	1.4249	97.33	1.2985	97.0
30%	1.2825	35.0	1.1382	32.33
40%	1.3024	44.33	1.1746	53.33
50%	1.3004	39.33	1.1629	39.0
60%	1.2436	14.66	1.1080	15.66
70%	1.2742	26.66	1.1283	25.0
80%	1.3970	87.66	1.2568	84.66
90%	1.2930	38.33	1.1553	36.66

Table A.4 The basal completion error and the percent of edges that produce higher error for all percentages of edges removed for Erdős-Rényi random graphs of the form $G(25, 0.4)$

Edges Removed (%)	Basal RMSE	Worse Edges (%)	Basal MAE	Worse Edges (%)
10%	1.2341	76.66	1.0773	73.0
20%	1.2247	67.33	1.0782	70.66
30%	1.2711	82.33	1.1196	81.33
40%	1.2001	43.0	1.0291	35.0
50%	1.2378	72.33	1.0682	63.0
60%	1.2636	75.0	1.1118	75.33
70%	1.1872	28.99	1.0482	40.0
80%	1.1858	32.66	1.0333	34.33
90%	1.2866	88.0	1.1541	92.33

Table A.5 The basal completion error and the percent of edges that produce higher error for all percentages of edges removed for Erdős-Rényi random graphs of the form $G(25, 0.6)$

Edges Removed (%)	Basal RMSE	Worse Edges (%)	Basal MAE	Worse Edges (%)
10%	1.0719	26.66	0.8949	23.33
20%	1.1973	93.33	1.0316	91.66
30%	1.1397	66.33	0.9922	76.0
40%	0.9732	0.666	0.7849	0.666
50%	1.0772	27.33	0.9092	28.00
60%	1.0806	28.33	0.9108	27.66
70%	1.0205	6.666	0.8312	5.0
80%	1.0496	12.33	0.8719	11.66
90%	1.1522	67.66	0.9973	71.0

Table A.6 The basal completion error and the percent of edges that produce higher error for all percentages of edges removed for Erdős-Rényi random graphs of the form $G(25, 0.7)$

Edges Removed (%)	Basal RMSE	Worse Edges (%)	Basal MAE	Worse Edges (%)
10%	0.9445	40.33	0.7659	32.66
20%	0.8914	8.0	0.7412	17.66
30%	0.9826	62.66	0.8054	55.00
40%	1.0009	66.33	0.8357	65.33
50%	0.8383	1.666	0.6597	2.333
60%	0.9236	20.66	0.7791	34.0
70%	1.0246	76.66	0.8728	78.33
80%	0.9884	58.66	0.8203	54.0
90%	1.0583	88.33	0.9155	90.0

Table A.7 The basal completion error and the percent of edges that produce higher error for all percentages of edges removed for Erdős-Rényi random graphs of the form $G(25, 0.8)$

Edges Removed (%)	Basal RMSE	Worse Edges (%)	Basal MAE	Worse Edges (%)
10%	0.8708	49.0	0.6840	31.0
20%	0.8533	34.0	0.6948	32.33
30%	0.9176	76.33	0.7621	73.0
40%	0.9716	94.0	0.8229	93.66
50%	0.8869	56.00	0.7358	53.66
60%	0.8449	27.0	0.7013	34.33
70%	0.8918	56.66	0.7316	52.33
80%	0.8154	10.0	0.6538	11.33
90%	0.8633	30.66	0.7021	28.99

Table A.8 The basal completion error and the percent of edges that produce higher error for all percentages of edges removed for Erdős-Rényi random graphs of the form $G(25, 0.9)$

REFERENCES

REFERENCES

[1] Jacob Abernethy, Francis Bach, Theodoros Evgeniou, and Jean-Philippe Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *The Journal of Machine Learning Research*, 10:803–826, 2009.

[2] A Annibale and ACC Coolen. What you see is not what you get: how sampling affects macroscopic features of biological networks. *Interface Focus*, 1(6):836–856, 2011.

[3] Sitaram Asur and Bernardo A Huberman. Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 492–499. IEEE, 2010.

[4] Eytan Bakshy, Jake M Hofman, Winter A Mason, and Duncan J Watts. Everyone's an influencer: quantifying influence on twitter. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 65–74. ACM, 2011.

[5] Iman Barjasteh, Rana Forsati, Abdol-Hossein Esfahanian, and Hayder Radha. Cold-start item and user recommendation with decoupled completion and transduction. In *RecSys*, pages 91–98, 2015.

[6] Iman Barjasteh, Rana Forsati, Dennis Ross, Abdol-Hossein Esfahanian, and Hayder Radha. Cold-start recommendation with provable guarantees: A decoupled approach.

[7] Justin Basilico and Thomas Hofmann. Unifying collaborative and content-based filtering. In *ICML*, page 9. ACM, 2004.

[8] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[9] Daniel Billsus and Michael J Pazzani. User modeling for adaptive news access. *User modeling and user-adapted interaction*, 10(2-3):147–180, 2000.

[10] Stephen P Borgatti and Martin G Everett. A graph-theoretic perspective on centrality. *Social networks*, 28(4):466–484, 2006.

[11] Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[12] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[13] Guo-Ray Cai and Yu-Geng Sun. The minimum augmentation of any graph to a k-edge-connected graph. *Networks*, 19(1):151–172, 1989.

[14] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[15] Paul A. Catlin. A reduction method to find spanning eulerian subgraphs. *Journal of Graph Theory*, 12(1):29–44, 1988.

[16] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, volume 60. Citeseer, 1999.

[17] Gabriella Contardo, Ludovic Denoyer, and Thierry Artieres. Representation learning for cold-start recommendation. *arXiv preprint arXiv:1412.7156*, 2014.

[18] Aron Culotta. Towards detecting influenza epidemics by analyzing twitter messages. In *Proceedings of the first workshop on social media analytics*, pages 115–122. ACM, 2010.

[19] Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. The yahoo! music dataset and kdd-cup'11. In *KDD Cup*, pages 8–18, 2012.

[20] Asmaa Elbadrawy and George Karypis. Feature-based similarity models for top-n recommendation of new items. 2014.

[21] Paul Erdős. On some extremal problems in graph theory. *Israel Journal of Mathematics*, 3(2):113–116, 1965.

[22] Paul Erdős, András Hajnal, and John W Moon. A problem in graph theory. *American Mathematical Monthly*, pages 1107–1110, 1964.

[23] Abdol-Hossein Esfahanian and S Louis Hakimi. On computing a conditional edge-connectivity of a graph. *Information Processing Letters*, 27(4):195–199, 1988.

[24] Yi Fang and Luo Si. Matrix co-factorization for recommendation with rich side information and implicit feedback. In *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, pages 65–69. ACM, 2011.

[25] András Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM Journal on Discrete Mathematics*, 5(1):25–53, 1992.

[26] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[27] Harold N Gabow and Eugene W Myers. Finding all spanning trees of directed and undirected graphs. *SIAM Journal on Computing*, 7(3):280–287, 1978.

[28] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM*. IEEE, 2010.

[29] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. MyMediaLite: A free recommender system library. In *ACM, Recommender Systems, 2011*.

[30] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM*, 2005.

[31] Quanquan Gu and Jie Zhou. Learning the shared subspace for multi-task clustering and transductive transfer classification. In *ICDM'09*, pages 159–168. IEEE, 2009.

[32] Roger Guimerà and Marta Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences*, 106(52):22073–22078, 2009.

[33] Asela Gunawardana and Christopher Meek. A unified approach to building hybrid recommender systems. In *Proceedings of the third ACM, Recommender systems*, 2009.

[34] Sunil Kumar Gupta, Dinh Phung, Brett Adams, Truyen Tran, and Svetha Venkatesh. Nonnegative shared subspace learning and its application to social media retrieval. In *ACM SIGKDD*, pages 1169–1178. ACM, 2010.

[35] Sunil Kumar Gupta, Dinh Phung, Brett Adams, and Svetha Venkatesh. Regularized nonnegative shared subspace learning. *Data mining and knowledge discovery*, 26(1):57–97, 2013.

[36] Steve Hanneke and Eric P Xing. Network completion and survey sampling. In *AISTAT*, pages 209–215, 2009.

[37] Thorsten Hennig-Thurau, Caroline Wiertz, and Fabian Feldhaus. Exploring the twitter effect: an investigation of the impact of microblogging word of mouth on consumers early adoption of new products. *Available at SSRN*, 2016548, 2012.

[38] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[39] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142. ACM, 2010.

[40] Bhargav Kanagal, Amr Ahmed, Sandeep Pandey, Vanja Josifovski, Jeff Yuan, and Lluis Garcia-Pueyo. Supercharging recommender systems using taxonomies for learning user purchase behavior. *Proceedings of the VLDB Endowment*, 5(10):956–967, 2012.

[41] Myunghwan Kim and Jure Leskovec. The network completion problem: Inferring missing nodes and edges in networks. In *SDM*, pages 47–58. SIAM, 2011.

[42] Noam Koenigstein, Gideon Dror, and Yehuda Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. pages 165–172. ACM, 2011.

[43] Jnos Komls and Mikls Simonovits. Szemerdi's regularity lemma and its applications in graph theory, 1996.

[44] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[45] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):1, 2010.

[46] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[47] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *SDM*, volume 5, pages 1–5. SIAM, 2005.

[48] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[49] Guang Ling, Michael R Lyu, and Irwin King. Ratings meet reviews, a combined approach to recommend. In *ACM Conference on Recommender Systems*, pages 105–112. ACM, 2014.

[50] Juntao Liu, Caihua Wu, and Wenyu Liu. Bayesian probabilistic matrix factorization with social relations and item contents for recommendation. *Decision Support Systems*, 2013.

[51] Nathan N Liu, Xiangrui Meng, Chao Liu, and Qiang Yang. Wisdom of the better few: cold start recommendation via representative based rating elicitation. In *ACM Conference on Recommender Systems*, pages 37–44. ACM, 2011.

[52] Mingsheng Long, Jianmin Wang, Guiguang Ding, Wei Cheng, Xiang Zhang, and Wei Wang. Dual transfer learning. In *SDM*, pages 540–551. SIAM, 2012.

[53] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940. ACM, 2008.

[54] Wolfgang Mader. A reduction method for edge-connectivity in graphs. *Annals of Discrete Mathematics*, 3:145–164, 1978.

[55] Farzan Masrour, Iman Barjesteh, Rana Forsati, Abdol-Hossein Esfahanian, and Hayder Radha. Network completion with node similarity: A matrix completion approach with provable guarantees. pages 302–307. ACM, 2015.

[56] David W Matula. Determining edge connectivity in 0 (nm). In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 249–251. IEEE, 1987.

[57] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *AAAI/IAAI*, pages 187–192, 2002.

[58] Aditya Krishna Menon, Krishna-Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. Response prediction using collaborative filtering with hierarchies and side-information. In *ACM SIGKDD*, pages 141–149. ACM, 2011.

[59] Aditya Krishna Menon and Charles Elkan. A log-linear model with latent features for dyadic prediction. In *ICDM*, pages 364–373. IEEE, 2010.

[60] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Machine Learning and Knowledge Discovery in Databases*, pages 437–452. Springer, 2011.

[61] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[62] Juhani Nieminen. On the centrality in a graph. *Scandinavian Journal of Psychology*, 15(1):332–336, 1974.

[63] Juhani Nieminen. Distance center and centroid of a median graph. *Journal of the Franklin Institute*, 323(1):89–94, 1987.

[64] Uroš Ocepek, Jože Rugelj, and Zoran Bosnić. Improving matrix factorization recommendations for examples in cold start. *Expert Systems with Applications*, 2015.

[65] Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. Transfer learning in collaborative filtering for sparsity reduction. In *AAAI*, volume 10, pages 230–235, 2010.

[66] Manos Papagelis, Gautam Das, and Nick Koudas. Sampling online social networks. *Knowledge and Data Engineering, IEEE Transactions on*, 25(3):662–676, 2013.

[67] Seung-Taek Park and Wei Chu. Pairwise preference regression for cold-start recommendation. In *RecSys*, pages 21–28, 2009.

[68] Seung-Taek Park, David Pennock, Omid Madani, Nathan Good, and Dennis DeCoste. Naïve filterbots for robust cold-start recommendations. pages 699–705, 2006.

[69] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[70] Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[71] Alexandrin Popescul, David M Pennock, and Steve Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *UAI*, pages 437–444, 2001.

[72] Ian Porteous, Arthur U Asuncion, and Max Welling. Bayesian matrix factorization with side information and dirichlet process mixtures. In *AAAI*, 2010.

[73] Benjamin Recht. A simpler approach to matrix completion. *The Journal of Machine Learning Research*, 12:3413–3430, 2011.

[74] Steffen Rendle. Factorization machines. In *ICDM*, pages 995–1000. IEEE, 2010.

[75] Steffen Rendle and Lars Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 251–258. ACM, 2008.

[76] Jasson DM Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719. ACM, 2005.

[77] Dennis Ross, Bruce E Sagan, Ronald Nussbaum, and Abdol-Hossein Esfahanian. On constructing regular distance-preserving graphs. *arXiv preprint arXiv:1405.1713*, 2014.

[78] S Roweis. Nips dataset (2002). *http://www. cs. nyu. edu/˜ roweis*.

[79] Martin Saveski and Amin Mantrach. Item cold-start recommendations: learning local collective embeddings. In *RecSys*, pages 89–96. ACM, 2014.

[80] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, pages 253–260. ACM, 2002.

[81] Hanhuai Shan and Arindam Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *ICDM*, pages 1025–1030. IEEE, 2010.

[82] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):3, 2014.

[83] Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336, 2004.

[84] Michele Trevisiol, Luca Maria Aiello, Rossano Schifanella, and Alejandro Jaimes. Cold-start news recommendation with domain-dependent browse graph. In *ACM Conference on Recommender Systems*, volume 14, 2014.

[85] Omar Wasow, Alex Baron, Marlon Gerra, Katharine Lauderdale, and Han Zhang. 1 can tweets kill a movie? an empirical evaluation of the bruno effect. *Available at SSRN*, 2010.

[86] Wouter Weerkamp and Maarten De Rijke. Activity prediction: A twitter-based exploration. In *SIGIR Workshop on Time-aware Information Access*, 2012.

[87] Xi Zhang, Jian Cheng, Shuang Qiu, Guibo Zhu, and Hanqing Lu. Dualds: A dual discriminative rating elicitation framework for cold start recommendation. *Knowledge-Based Systems*, 73:161–172, 2015.

[88] Yuchen Zhang, Amr Ahmed, Vanja Josifovski, and Alexander Smola. Taxonomy discovery for personalized recommendation. pages 243–252. ACM, 2014.

[89] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *ACM SIGIR*, pages 315–324. ACM, 2011.

[90] Tinghui Zhou, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *SDM*, pages 403–414, 2012.

[91] Tinghui Zhou, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. volume 12, pages 403–414. SIAM, 2012.