Reducing the Time and Cost Overruns for Aerospace Development Programs Using Precedence Networks Patterns

by Scott D. Button

B.S. in Mechanical Engineering, May 1986, University of Alaska, Fairbanks
M.S. in Mechanical Engineering, March 1995, University of Washington
M.S. in Engineering Management, May 2013, Washington State University

A Praxis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Engineering

January 10, 2019

Praxis directed by

Amir Etemadi
Assistant Professor of Engineering and Applied Science

Ebrahim Malalla
Visiting Associate Professor of Engineering and Applied Science

The School of Engineering and Applied Science of The George Washington University certifies that Scott David Button has passed the Final Examination for the degree of Doctor of Engineering as of November 08, 2018. This is the final and approved form of the Praxis.

Reducing the Time and Cost Overruns for Aerospace Development Programs Using Precedence Networks Patterns

Scott D. Button

Praxis Research Committee:

Amir Etemadi, Assistant Professor of Engineering and Applied Science, Praxis Director

Timothy Blackburn, Professional Lecturer of Engineering Management and Systems Engineering, Committee Member

Ebrahim Malalla, Visiting Associate Professor of Engineering and Applied Science, Committee Member

© Copyright 2019 by Scott D. Button
All rights reserved

Dedication

To Thuy, Don and Carol.

Acknowledgements

There are many colleagues and cohorts who helped me in this journey, and I am eternally grateful for their assistance. Dr. David L. Grose, for mentoring and guidance on this topic. Dr. Kumar Bhatia for his leadership and guidance on how to engage on strategy and change; Steven C. Holt, a generous collaborator in my technical career. Thomas E. Sherer, for suggestions and guidance around classification, clustering, the suggestion to look at the pure network structure, for speeding up the execution time of the triadic closure prediction algorithm, and years of mentoring for object-oriented design and Java structures. Tek D. Kim for assistance with Java, Linux, and business process export formats. Dr. Ricardo Fricks' discussions about machine learning, the suggestion to look at social network analysis, and how to use arXiv; Vipul Mascarenhas and Tom Vu for guidance on my approach at the Boeing Technical Excellence conference in 2018. Marjorie Blanco who suggested tuning the community algorithms and guided this work towards a recommendation system, and Thanh-Thuy T. Huynh, who provided detailed feedback on this praxis draft.

From my cohort, John Kamp for hosting study sessions, his many contributions to my understanding of the material and insight into my research, and his careful review of my draft praxis. Thanks to Taher Gomaa for his critiques of my analysis methods. From GW, I'm grateful to Dr. Etemadi, who guided this work at key points, and provided both general and specific detailed feedback on the written Praxis. I'm very grateful to Jason Hurlburt and the staff that keeps the Colonial One High Performance Computing Center, running. Finally, I am grateful for the numerous contributors to R bloggers, stack overflow, Quizlet and Google.

Abstract of Praxis

Reducing the Time and Cost Overruns for Aerospace Development Programs Using Precedence Networks Patterns

Many commercial and defense product development programs are large-scale, complex, and contain significant innovation. Such programs often suffer cost and schedule over-runs.

The goal of this work is to increase the production rate of well-ordered precedence networks for complex, large-scale aerospace product development. Earlier work in correspondence pattern automation enabled the automatic selection of inputs based on name matching. This praxis pursued broadening the scope to evaluate patterns of nodes and inputs. This research will accelerate the identification and reuse of patterns resident in project precedence networks, enabling a reduction in time and cost overruns for aerospace Product Development.

This work has applied link prediction techniques developed for social network analysis to a recommendation system which augments human selection of inputs during precedence network development. This work also introduces the Triad Significance Profile for a project management precedence network.

Using the link prediction techniques developed in social network analysis, and using a triadic closure algorithm developed in this praxis, it is shown that inputs may be predicted for the studied product development network, with reasonable prediction accuracy.

Table of Contents

Dedication ..... **iv**

Acknowledgements ..... **v**

Abstract of Praxis ..... **vi**

List of Figures ..... **x**

List of Tables ..... **xiii**

List of Acronyms ..... **xv**

Chapter 1 Introduction 1

1.1 Background 1

1.2 Problem Statement 6

1.3 Thesis Statement 6

1.4 Research Objectives 7

1.5 Research Questions and Hypotheses 7

1.6 Research Significance 8

1.7 Organization of Praxis 9

Chapter 2 Literature Review 10

2.1 Introduction 10

2.2 Search Process 10

2.3 Cost/Schedule Overruns 11

2.4 Planning/Ordering 12

2.5 Execution 13

2.6 Graph Theory 14

2.6.1 Introduction 14

2.6.2 Subgraph Isomorphism 14
2.7 Association Rule Mining 16
2.7.1 Frequent Subgraph Mining 16
2.7.2 Similar Graph Searches 19
2.8 Measurements of "Interestingness" 21
2.9 Motifs 21
2.9.1 Motifs in Engineered Systems 26
2.10 Link Prediction 26
2.11 Labeling Graphs for Mining 30
2.12 Work Breakdown Structure Automation 30
2.13 Summary 31
Chapter 3 Methods 33
3.1 Introduction 33
3.2 Research Approach 35
3.3 Node Classification 39
3.4 Link Prediction with Triadic Closure, Toy Example 41
3.5 Time and Cost 44
3.6 Summary 45
Chapter 4 Results 46
4.1 Introduction 46
4.2 Successful Experiments 52
4.3 Failed Experiments 55
4.4 Analysis 55

4.5 Patterns in project management networks: A search for meaning... 56
4.6 Social Network Analysis 60
4.7 Link Prediction 66
4.7.1 Link Prediction Results 68
4.7.2 Binary Classification of Link Prediction Results 70
4.7.3 Additional Link Predictors 74
4.8 Precedence Network Development Rate 75
4.9 Time and Cost 79
4.10 Summary 82
Chapter 5 Discussion/Conclusion 84
5.1 Summary of Contribution 84
5.2 Results Discussion 85
5.3 Potential Future Work 87
Bibliography 89
Appendix A Toy Example Remaining Mining Results 106
Appendix B Size 17 Frequent Subgraphs 107
Appendix C Frequent Subgraph Clusters 108
Appendix D Triadic Closure Algorithms 109
Appendix E Input Recommender System Design 115
Appendix F Component Diagram 116

List of Figures

Figure 1-1 (A) Toy graph representing a portion of a project management precedence network. (B) Subgraph isomorph of (A). 4

Figure 2-1: Literature Search Storyboard 10

Figure 2-2: An example of a subgraph isomorph in blue (left) and its matches in a larger graph (right). 15

Figure 2-3: Subgraph ratio profiles for nondirected networks. From (Milo et al., 2004). Reprinted with permission from AAAS. 24

Figure 2-4: Triadic Closure Illustration (1) an open triad, with two directed edges indication information flow into the top node. (2) A closed triad with an edge directed to the left, and (3) a closed triad with an edge directed to the right. 30

Figure 3-1: Work Breakdown Structure to prepare for a moon launch 34

Figure 3-2: Level 1 precedence network 34

Figure 3-3: Level 2 precedence network 35

Figure 3-4: Precedence Network Pattern Research Process 36

Figure 3-5: Toy example target graph 42

Figure 3-6: Partial results of mining toy example. 43

Figure 3-7: Open triad, color coded per color assignments 44

Figure 4-1: Connection topology spectrum. Reprinted with permission from Nature: Watts, D. J., & Strogatz, S. H. Collective dynamics of 'small-world' networks, 1998. 46

Figure 4-2: Degree Distribution for level two of the airplane product development

network 48
Figure 4-3: Degree distribution for level three of the airplane product development network 49
Figure 4-4: Degree distribution for level four of the airplane product development network 50
Figure 4-5: Degree distribution for level five of the airplane product development network 51
Figure 4-6: BiFan (#204) and Bi-Parallel (#2182), two of fifteen motifs found in the Level 4 Airplane Product development Network. 57
Figure 4-7 The Triad significance profile (TSP) of Networks from Various Disciplines: A: Button Praxis; B-F from (Milo et al., 2004). B-F Reprinted with permission from AAS 58
Figure 4-8: Vertex color to hyponym assignment for level three airplane product development network 61
Figure 4-9: Network characteristics and FANMOD settings used to mine level three airplane product development network 62
Figure 4-10: Colored triads found in level three airplane product development network 63
Figure 4-11: Motifs Across Communities 64
Figure 4-12: Recall by fraction for airplane network at level 2-5 69
Figure 4-13: Binary classification for airplane level 3 triadic closure network predictions 71
Figure 4-14: Binary classification for airplane level 4 triadic closure network

predictions 72
Figure 4-15: Binary classification for airplane level 5 triadic closure network predictions 73
Figure 4-16: Other predictors for recall by fraction for airplane network level 2-5 75
Figure 4-17: Five years of network edge edits 77
Figure 4-18: Edits to level 4 airplane network for first half of 2018 78
Figure 4-19: One of 40 runs of a Monte-Carlo Simulation comparing schedule-driven to order-driven execution 80
Figure 4-20: Results of 40 runs for each of a Monte-Carlo simulation comparing schedule-driven to order-driven execution. 81
Figure 4-21 Well-Ordering Payoff on Preliminary Design 82
Figure 5-1: Input Recommendation System 85
Figure E-1: Use Case Diagram 115
Figure E-2: Review Recommended Inputs Activity Diagram 115
Figure E-3: Select Recommended Inputs Activity Diagram 115

List of Tables

Table 2-1: Graph Matching Time Periods by Application Area Adapted From (Conte, Foggia, Sansone, & Vento, 2004) 17

Table 2-2: Taxonomy of graph matching algorithms adapted from (Conte et al., 2004) 17

Table 2-3: Pattern Recognition Techniques Adapted From (Foggia, Percannella, & Vento, 2014). 18

Table 2-4: Network motifs described by Milo et. al. From (Milo et al., 2002). Reprinted with permission from AAAS. 23

Table 2-5: Node Similarity Metrics 28

Table 3-1: Comparison of Analysis Techniques 38

Table 3-2: Hypernym-Hyponym Confidence Relations 40

Table 3-3: Proposed and Selected Hyponyms 41

Table 3-4: Label Assignment for Tox Example. 42

Table 4-1: Network Characteristics and Mining Results. 55

Table 4-2: Statistics for Motifs Found Across Communities 65

Table 4-3: Networks ranked by predictability 66

Table 4-4: Run times for Triadic Closure Close All 68

Table 4-5: Time (seconds) to create precedence in a thread, old method vs. triadic closure method 76

Table 5-1: Research Questions and Hypotheses 85

Table 5-2: Research Hypothesis Table 86

List of Symbols

τ	Frequency Threshold
G	Graph
V	Vertices
E	Edges
L	Label Assignment Function
S	subgraph
$\subseteq$	subset
$\in$	element of
$\cup$	union
f	function
$\rightarrow$	function arrow (to)
$=$	equals
$\Gamma(x)$	Set of neighbors of x
$\lvert \Gamma(x) \rvert$	Cardinality of set x
$\sum$	sum
γ(s)	local community degree of s

List of Acronyms

AA	Adamic/Adar
AACD	APROMORE Approximate Clone Detection
AAS	American Association for the Advancement of Science
ACC	Average Clustering Coefficient
AGraMi	Approximate Graph Mining
AKA	Also Known As
APD	Airplane Product development
APROMORE	Advanced Process Model Repository
ASP	Average Shortest Path
BA	Barabasi and Albert
BPM	Business Process Modeling
CAR	Cannistraci-Alanis-Ravasi
CN	Common Neighbors
CentOS	Community Enterprise Operating System
DAG	Directed Acyclic Graph
DNA	Deoxyribonucleic acid
FANMOD	Fast Network Motif Detection
FSM	Frequent Subgraph Mining
GAO	Government Accountability Office
GB	Gigabytes
GCC	Global Clustering Coefficient
GHz	Gigahertz

GW	George Washington
GXL	Graph Exchange Language
GraMi	Graph Mining
HP	Hewlett-Packard
HPLP	High Performance Link Prediction
ID	Identifier
LOD	Linked Open Data
NP	Non-Deterministic Polynomial Time
NPD	New Product development
OARPLAN	Objects, Actions and Resources Plan
OS	Operating System
PA	Preferential Attachment
PC	Personal Computer
PCA	Principal Component Analysis
PWr	Wrocław University of Technology
RAM	Random Access Memory
REAFUM	Representative Approximate Frequent Subgraph Mining
ROC	Receiver-Operator Curve
RoSE	Reoccurring Structures Detection
RQ	Research Question
RSS	Rich Site Summary
SD	Standard Deviation
SESE	Single-Entry Single-Exit

SPARQL	SPARQL Protocol and RDF Query Language (recursive acronym)
TCCA	Triad Closure Close All
TCPC	Triad Closure Percent Close
TSP	Triad significance profile
UC	University of California
UML	Unified Modeling Language

Chapter 1 Introduction

1.1 Background

Many commercial and defense product development programs are large-scale, complex, and contain significant innovation. Such programs often suffer from cost and schedule over-runs. A Government Accountability Office (GAO) study of weapons programs showed $484 billion in total acquisition cost growth since initial estimates (Sullivan, 2017). Another GAO study of 778 Information Technology projects found that 413 of these projects (expending $25.2 billion) were poorly planned, poorly performed, or both (Powner, 2008).

Project planning is a fundamental discipline for project management. A study of 2,820 large, complex programs found that the use of project management control and execution systems was the top success driver (Crosby, 2012). Project management and control systems rely on valid project planning. One output from project planning is a precedence network, which provides the ordering of activities necessary to develop products. Formally, "*The logical precedence network of a project is a mapping network which consists of the project activities as its nodes and represents their logical priority or sequence by its edges, which should be met in the execution of project activities.*" (Golpayegani & Parvaresh, 2011). The precedence network may be arranged according to the well-ordering principle in mathematics (Grose, Sherer, & Button, 2014). This means the precedence network can be organized into subsets, where each subset is placed in order according to its precedence relationships. The set of activities which has no inputs

is the first subset in the order. The set of activities which are preceded by the activities with no inputs is the second subset in the order, and this process is repeated until all subsets are ordered. A well-ordered precedence network may be used as an input to develop a program schedule in the time domain (vs. the order domain). Disregarding the precedence order of activities may result in rework of those activities. Rework during product development can lead to cost and schedule over-runs; in one study, rework interacted with other reinforcing loops to cause "enormous delays" causing an increase in the planned duration by nearly five-fold (Jalili & Ford, 2016).

Project management is one of the primary management disciplines applied to product development. Business Process Modeling (BPM) is similar to project management, but in BPM the focus is on recurring activities rather than one-time activities. New product development is composed from processes which are repeated for every product developed, with some variation and customization. Where similar product development is conducted, similar project schedules are often created from a common project template.

The activities and precedence relationship between activities is a network and may be represented as a graph, with the activities represented as nodes, and precedence represented as edges. An example of a project management network is shown in Figure 1-1. The round shapes labeled 73, 15, or 61 represent activities in the network, where the numbers are identifiers for the activities. The arrows represent precedence relations in the network, where an arrow is directed from predecessor to successor.

Within a network, a collection of nodes and precedence could constitute a thread pattern. A thread pattern is the life-cycle of maturity of information for one or a few

functional groups. In air vehicles, an example of a thread pattern is the network of nodes and precedence which represents how the flight controls team interacts with the stability and controls team to mature information from concept to deliverable (Button, Brown, Kim, & Sherer, 2018).

The analog to thread patterns in business process modeling are process fragments. Process fragments have a plurality of definitions and interpretations. Most generally, a process fragment is a sub-graph of a process graph (Schumm et al., 2011).

Thread patterns potentially could be reused to speed up the rate of precedence network development. A thread pattern may be identified as a repeating subgraph, known as a subgraph isomorph. A subgraph isomorph is an extracted subset of nodes and inputs from a larger graph, which can be matched up to nodes and inputs in the large graph, regardless of how they are laid out for visualization. Figure 1-1 (B) is subgraph isomorphic to Figure 1-1 (A). Strictly speaking, a graph is an abstract representation of the actual network, and when represented as a graph the nodes are called vertices, and the relations between vertices are called arcs. This is an academic distinction, and for this praxis the terminology of nodes and edges will be used.

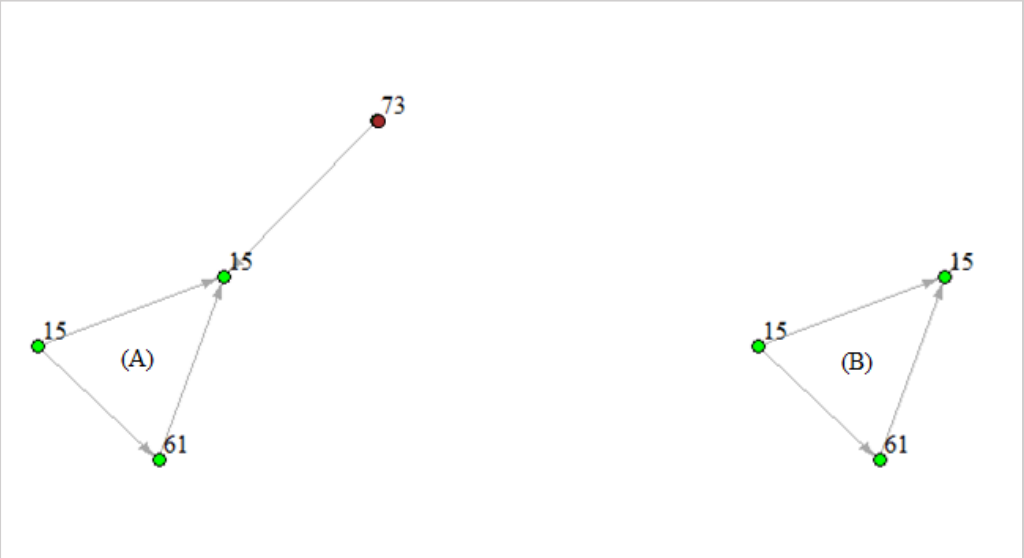


Figure 1-1 (A) Toy graph representing a portion of a project management precedence network. (B) Subgraph isomorph of (A).

Frequent Subgraph Mining is finding all patterns in a graph or database where the count of occurrences is over a given threshold. Mining of patterns for re-use is a theme in Business Process Modeling literature (Dijkman et al., 2013; Ekanayake, Dumas, Garcia-Banuelos, & La Rosa, 2013; Koschmider, Fellmann, Schoknecht, & Oberweis, 2014; La Rosa et al., 2015; La Rosa et al., 2011; Skouradaki, Andrikopoulos, Kopp, & Leymann, 2016; Ying Li et al., 2014; Zaaboub Haddar, Makni, & Ben Abdallah, 2014) but not in Project Management literature (Bell, 1989; Darwiche, Levitt, & Hayes-Roth, 1988; Emblemsvåg, 2014; King & Lukas, 1973; Rupani, 2011).

Techniques developed for social network analysis can be useful for analysis of other networks, including a project management precedence network. Typical social network analysis tasks can include finding:

- The structure of a social network

- Attributes of the network such as radius, diameter, centrality, shortest paths, etc.
- Communities in the social network

Social Networks are networks of relationships or interactions, where the nodes are people or some other actor, and the edges are their relationships or interactions. Online examples of social network sites include Facebook, LinkedIn, Twitter, Instagram and Google+. Traditional sources of social network interactions include phone calls, e-mails, citation networks and author collaborations. These social network interactions can be represented as a graph. The nodes are the people, and the edges (called links in social networks) are the relationships (phone calls, e-mails, citations or collaborations) between them. Social Network Analysis is the analysis of social networks using techniques of network theory, graph theory, sociology, statistics, and information science (Akhtar, 2014).

Tools and techniques developed for finding and understanding patterns in biological networks can be useful for finding and understanding patterns in a project management precedence network. An example of such techniques is bioinformatics. Bioinformatics is the use of software tools to understand biological data. It is an interdisciplinary field which combines biology, computer science, information engineering, math and statistics for analysis and interpretation of biological data. Bioinformatics is used to analyze and interpret amino acid sequences, locate genes in sequences, and determine genes that encode proteins.

This Praxis combines mining techniques developed for bioinformatics with social network analysis techniques to develop a system to recommend inputs, which augments human selection of inputs during precedence network development.

1.2 Problem Statement

Out of order project execution in aerospace product development contributed to $484 billion in total acquisition cost growth since initial estimates (Sullivan, 2017). To reduce out of order project execution, well-ordered precedence networks for complex, large-scale aerospace product development are necessary. One obstacle to developing a well-ordered precedence network is the time and modeling expertise necessary to develop a robust and comprehensive precedence network. In a study of the utility of business process modeling in 60 banks, the effort exceeded the value by 36% (Becker, Weiß, & Winkelmann, 2010). Although it would be desirable to increase the rate of network production by the application of more personnel, personnel with the aptitude for this type of modeling are rare and expensive compared to the computing equipment they use. Consequently, the use of automation or augmentation to increase this rate is a desirable solution.

This summarized problem statement is:

Out of order project execution in aerospace product development contributed to $484 billion in total acquisition cost growth since initial estimates.

1.3 Thesis Statement

There are numerous directions for automation and augmentation to increase the rate of precedence network development. Machine learning techniques including neural networks and data mining approaches are discussed in the literature (Faghihi, Nejat,

Reinschmidt, & Kang, 2015; Golpayegani & Parvaresh, 2011; Yusof & Latif, 2014). Knowledge-based systems are also discussed (Darwiche et al., 1988; Faghihi et al., 2015; Yusof & Latif, 2014). Earlier work in correspondence pattern automation enabled the automatic selection of inputs based on name matching (Grose, Sherer, & Button, 2016). This praxis broadened the scope to evaluate patterns of nodes and inputs. The thesis statement formulated for this praxis is:

The identification and reuse of patterns resident in project precedence networks will reduce time and cost overruns for aerospace product development.

1.4 Research Objectives

Through literature search and experiment, the definition of a pattern was clarified to be frequent subgraphs. A subgraph isomorph is an extracted subset of nodes and inputs from a larger graph, which can be matched up to nodes and inputs in the large graph, regardless of how they are laid out for visualization. A frequent subgraph is a set of nodes and inputs that are repeated within a large graph above some frequency threshold τ. The primary objectives of this research are to deliver:

O1: Evidence that there are patterns (frequent isomorphic subgraphs) within well-ordered precedence networks.

O2: A method to identify these patterns.

O3: A method to harness these patterns to expand precedence networks to include content in areas not previously modeled in these precedence networks.

1.5 Research Questions and Hypotheses

This praxis was guided by the following research questions (**RQ**):

RQ1: Are there patterns of tasks and inputs in an example of a well-ordered aerospace precedence network?

RQ2: What techniques can be used, machine learning, neural networks, data mining, etc., to identify patterns in networks?

RQ3: What are the appropriate categories for precedence network patterns?

RQ4: Does the use of patterns as enablers lead to cost and schedule reductions?

The research hypotheses (**H**) which were formulated and tested in this research are:

H1: There are patterns of tasks and inputs in well-ordered product development precedence networks.

H2: If we identify and re-use patterns, then the time required to produce well-ordered precedence networks will decrease.

H3: Time and cost to develop aerospace products can be reduced over time by a discipline of executing well-ordered precedence networks.

1.6 Research Significance

To our knowledge, there is limited literature on the mining of project management precedence networks for reuse. Although there is ample literature on mining business process models, the reuse of fragments is often problematic and rarely supported by tools (Koschmider et al., 2014).

Searches for network motifs is a specialized form of network mining. Network motifs are subgraph isomorphs which are statistically over-represented compared to their

frequency in a random graph (Milo et al., 2002). Statistically over-represented means the probability P of the subgraph isomorph appearing in a randomized network an equal or greater number of times than in the studied network is lower than some cutoff value, P=0.05 used in this work.

No literature on the application of motifs to project management precedence networks has been found. To the best of our knowledge, this praxis is the pioneering effort for the new triad significance profile for a project management precedence network, enabling a new superfamily (Milo et al., 2004) of project management networks characterized by their triad significance profiles.

This work applied link prediction techniques developed for social network analysis to a recommendation system which augments human selection of inputs during precedence network development. To our knowledge, link prediction techniques from social network analysis applied to project management precedence networks is innovative and groundbreaking.

1.7 Organization of Praxis

The remainder of the Praxis is organized as follows:

Chapter Two contains a literature review, with emphasis on frequent subgraph mining, motifs, and link prediction.

Chapter Three contains methods and describes both successful and unsuccessful experiments.

Chapter Four contains the experimental results and analysis.

Chapter Five contains the discussion, conclusions, research contribution and potential future work.

Chapter 2 Literature Review

2.1 Introduction

This literature search was organized into five main areas: Cost/Schedule Overruns, Planning/Ordering, Execution, Graph Theory, and Work Breakdown Structure (WBS) Automation, as shown in Figure 2-1.

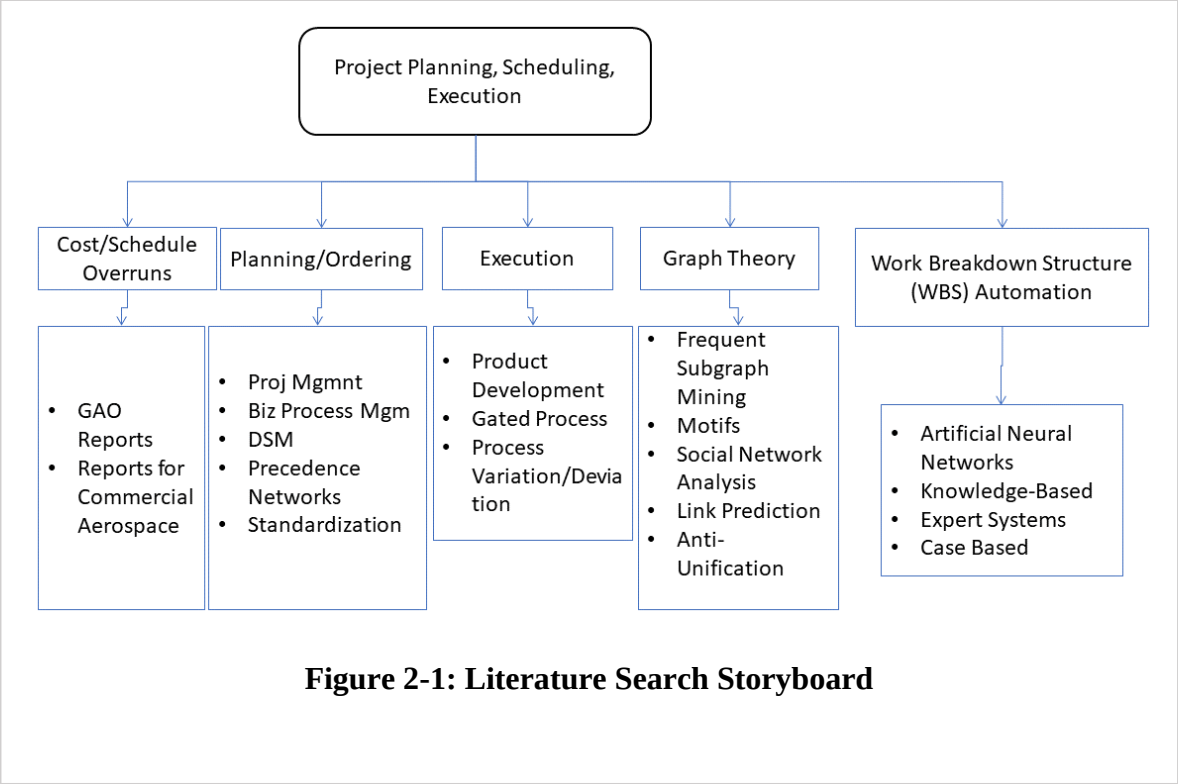


Figure 2-1: Literature Search Storyboard

2.2 Search Process

The bulk of the literature search was conducted in March, 2018. In order to establish the novelty of this work, a structured search approach was used (Cook, Greengold, Ellrodt, & Weingarten, 1997; Schoknecht, Thaler, Fettke, Oberweis, & Laue, 2017). The structured search used the terms “precedence network patterns” in George Washington (GW) Article Plus. The search was restricted to the discipline of engineering and provided 2,151 results. The first 500 results were reviewed, and 12 of the results

were selected as relevant to this research. As the results were ranked by relevance, the most relevant references were found in the first 300 results. Per the GW Libraries' "Citation Chasing & Cited Reference Searching" topic (Brown, 2018), Google Scholar was used for forward citation searching. Five forward citations were identified as relevant. In addition to the structured search, a literature search was conducted for each of the topics identified in Figure 2-1. Further literature searches were conducted ad-hoc for problems and challenges faced in this research. In total over 300 references were reviewed, and over 100 citations were incorporated in this praxis.

2.3 Cost/Schedule Overruns

A Government Accountability Office (GAO) study of weapons programs showed $484 billion in total acquisition cost growth since initial estimates (Sullivan, 2017). One cause given for the overruns is concurrent software and hardware development during production. Two other causes given are starting system development with immature technologies, and incomplete systems engineering reviews. The GAO's solution is "knowledge-based practices", an approach that consists of technology readiness levels, manufacturing readiness levels and an assessment of the readiness levels as each phase of the program is completed (Mackin, 2009; Sullivan, 2010).

Another GAO study of 778 Information Technology (IT) projects found that 413 of these projects (expending $25.2 billion) were poorly planned, poorly performed, or both. This study finds that planning, management, and oversight need to be strengthened, and recommends an IT investment management framework. (Powner, 2008).The investment management framework has five maturity stages and describes the critical processes for each stage (Powner, 2004).

2.4 Planning/Ordering

Project planning is a fundamental discipline for project management. A study of 2,820 large, complex programs found that the use of project management control and execution systems was the top success driver (Crosby, 2012). Project management and control systems rely on valid project planning. One output from project planning is a precedence network, which provides the ordering of activities necessary to develop products. Formally, "*The logical precedence network of a project is a mapping network which consists of the project activities as its nodes and represents their logical priority or sequence by its edges, which should be met in the execution of project activities.*" (Golpayegani & Parvaresh, 2011).

The precedence network may be arranged according to the well-ordering principle in mathematics (Grose et al., 2014). This means the precedence network can be organized into subsets or "bands", where each band is placed in order according to its precedence relationships. A "band" is a subset of activities which can be conducted concurrently (Grose, 1994). The set of activities which has no inputs is the first band in the order. The set of activities which are preceded by the activities with no inputs is the second band in the order, and this process is repeated until all subsets are ordered.

A well-ordered precedence network may be used as an input to develop a program schedule in the time domain (vs. the order domain). Disregarding the precedence order of activities will often result in rework of those activities. Rework during product development can lead to cost and/or schedule over-runs; in one study, rework interacted with other reinforcing loops to cause "enormous delays" causing an increase in the planned duration by nearly five-fold (Jalili & Ford, 2016).

Project management is one of the primary management disciplines applied to product development. Business Process Modeling (BPM) is similar to project management, but in BPM the focus is on recurring activities rather than one-time activities. New product development is composed from processes which are repeated for every product developed, with some variation and customization. Where similar product development is conducted, similar project schedules are often created from a common project template.

2.5 Execution

Project execution is conducting the activities which lead to the project's final deliverables. During execution, governance, policies, and practices can affect the project's outcomes in terms of quality, cost and schedule. Governance is the establishment, measurement and monitoring of policies intended to maximize the benefits of the project to its stakeholders. In one study of new product development, when upper managers exerted too much governance, project performance was degraded. On the contrary, improved project performance was achieved when team members and upper management co-developed the operational controls (Bonner, Ruekert, & Walker, 2002).

The stage-gate process is a widely used governance approach. It consists of a series of stages, where project activities are completed, followed by gates, where go forward/kill decisions are made. A drawback is that there are many implementation pitfalls, such as putting activities in sequence that could be done in parallel, and over-regulating the project with rules, regulations, and mandatory procedures (Cooper, 2017).

A distinction made in the execution of a project is related to process customization, variation and deviation. Process variation is related to variety in what

activities are executed, the sequence in which the activities are completed, and what tools are used. Process customization refers to a form of process variation where the project plan is adjusted prior to execution, to accommodate a particular project's needs. Process deviation is a form of process customization which has to do with changing the planned activities during execution. In a detailed empirical study at one company, process customization and process deviation had a net negative effect on project performance. (Rupani, 2011).

2.6 Graph Theory

2.6.1 Introduction

This section will introduce graph theory, then discuss topics which use graph theory. These topics include:

- Subgraph isomorphism, which defines exact graph matching.
- Graph mining, which searches for frequent subgraphs above some support threshold. Both exact matches and inexact matches are discussed.
- Motifs, which are statistically over-represented frequent subgraphs compared to a random graph.
- Link prediction. This is an approach from social network analysis that uses statistical techniques to predict where edges will form in a future evolution of a social network.

2.6.2 Subgraph Isomorphism

A precedence network may be represented as a Directed Acyclic Graph (DAG). A directed graph has edges directed from a source vertex to a sink vertex, most commonly displayed with an arrow head pointing in to the sink vertex. There are also undirected

graphs, where the edges have no direction. A directed acyclic graph is feed-forward, meaning there is no way to start at any node, follow the directed edges and eventually loop back to the starting node.

The basis for showing graphs as isomorphic is graph theory. A graph $G = (V, E, L)$ consists of vertices V, edges E, and a function L which assigns labels to the vertices. A graph $S = (V_s, E_s, L_s)$ is a subgraph of graph G if and only if $V_s \subseteq V, E_s \subseteq E$ and $L_s(v) = L(v)$ for all $v \in V_s \cup E_s$. A subgraph isomorphism is an injective function $f : V_s \to V$ which satisfies (a) $(L_s(v) = L(f(v))$ for all nodes $v \in V_s$, and (b) $(f(u), f(v)) \in E$ (Elseidy, Kalnis, Abdelhamid, & Skiadopoulos, 2014; Saltz et al., 2014).

An example of subgraph isomorphism is provided in Figure 2-2. The subgraph isomorph is shown extracted from the target graph and highlighted in blue in the pattern graph at the left, and is also blue in the larger target graph at the right. Note the blue subgraph isomorph is shown twice in the larger target graph. Note also that the isomorph at the left of the larger target graph is flipped around a horizontal axis by 180 degrees, yet is isomorphic to the blue pattern at the right of the larger graph.

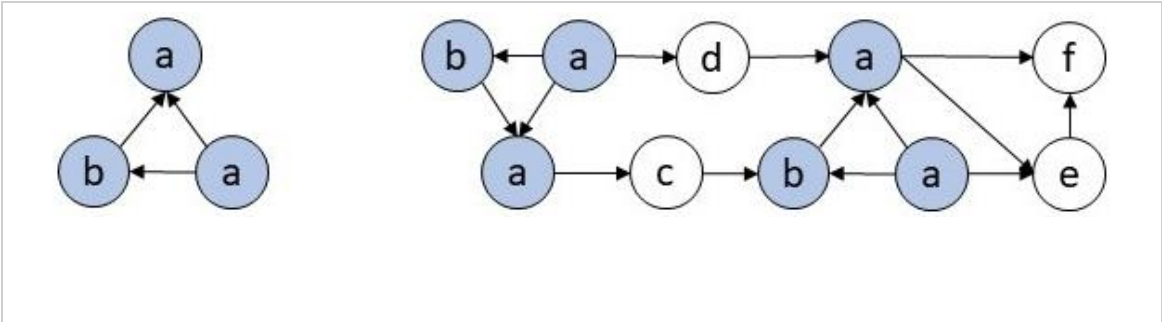


Figure 2-2: An example of a subgraph isomorph in blue (left) and its matches in a larger graph (right).

2.7 Association Rule Mining

The seminal literature for frequent subgraph mining is association rule mining. Rule mining considers the problem of the set of items (itemset) purchased in a transaction and stored in a database of transactions. Given an itemset and a purchase, can other purchases be predicted? In a 1993 paper, Agrawal et al. introduced the initial solution to the problem of mining the transaction database for rules associating sets of items which can be used to predict purchases (Agrawal, Imieliński, & Swami, 1993). In a 2006 overview paper, Kotsiantis and Kanellopoulos describe association rule mining as being decomposable into two problems: 1) find item sets whose frequency exceeds a support threshold, and 2) produce association rules for those frequent item sets above a specified confidence threshold (Kotsiantis & Kanellopoulos, 2006). In a 2012 paper, Borgelt describes an "annoying problem" as the frequent item sets output exceeding the size of the mined transaction database. Restrictions such as closed or maximal item sets are used to prune the results (Borgelt, 2012).

2.7.1 Frequent Subgraph Mining

The adoption of association rule mining to frequent subgraph mining was initiated in 1994 by Cook and Holder with their introduction of the SUBDUE algorithm. Their work is also built on a foundation of substructure search that dates back to graph theory work in 1971, and computer vision work in 1975 (Winston, 1975; Zahn, 1971).

A 2004 overview paper by Conte, Foggia, Sansone and Vento covers thirty years in the history of graph matching for pattern recognition. Table 2-1 shows the number of papers for each time period, broken down into applied areas.

Table 2-1: Graph Matching Time Periods by Application Area Adapted From (Conte, Foggia, Sansone, & Vento, 2004)

Period	2D and 3D Image Analysis	Document Processing	Biometric Identification	Image Databases	Video Analysis	Biomedical and Biological
Up to 1990	3 (0)	1 (1)				
1991-1994	3 (0)	2 (2)	1 (1)			1 (1)
1995-1998	8 (3)	9 (5)	6 (6)	5 (5)		1 (0)
1999-2002	19 (6)	8 (4)	8 (8)	8 (7)	6 (6)	2 (2)

Note. The values in parenthesis indicate the paper is presenting an application.

The taxonomy of topic areas is organized hierarchically into Exact Matching, Inexact Matching, and other matching problems, and then further decomposed as shown in Table 2-2.

Table 2-2: Taxonomy of graph matching algorithms adapted from (Conte et al., 2004)

Category	Topic	Sub-Topic
Matching	Exact Matching	Tree Search
		Other Techniques
		Special Kind of Graphs
	Inexact Matching	Tree Search
		Continuous Optimization
		Spectral Methods
		Other Techniques
	Other Matching Problems	

Graph and subgraph isomorphism are considered forms of exact graph matching. The isomorphism algorithms are classified as computing complexity non-deterministic polynomial time (NP)-complete, except for graph isomorphism. In 2004, Conte et. al stated it had not yet been demonstrated whether graph isomorphism was NP or not (Conte

et al., 2004). In an exciting recent development, Babai has found graph isomorphism to be solvable in quasi-polynomial time, a claim which was retracted then reinstated after finding and fixing an error (Babai, 2016; Klarreich, 2014).

The more recent developments in graph matching are exposed by Foggia, Percannella and Vento in their 2014 paper. The pattern recognition techniques they discuss are summarized in Table 2-3.

Table 2-3: Pattern Recognition Techniques Adapted From (Foggia, Percannella, & Vento, 2014).

Category	Topic
Graph Matching	Exact Matching
	Inexact Matching
Graph Embedding	Isometric Embeddings
	Spectral Embeddings
	Prototype-Based Embeddings
Graph Kernels	
Graph Clustering	Clustering of Graphs
	Graph-Based Learning
Misc. Problems	

The preponderance of graph matching algorithms uses a tree search, where an empty mapping function is filled a pair of nodes at a time, preserving the option to backtrack. Heuristics are used to prune the search space (Foggia et al., 2014). A significant development on his own well known 1976 pruned successor tree search algorithm (Ullmann, 1976) was introduced by Ullmann in 2010. This development introduced the idea of treating the search as a constraint satisfaction problem (Ullmann, 2010).

The graph mining problem can be bisected into the categories of transaction mining and the mining of a single large graph. Transaction mining treats the problem of

many small graphs in a graph database. Mining a single large graph tends to be a more difficult problem, as the complexity increases exponentially with the size of the graph (Elseidy et al., 2014).

2.7.2 Similar Graph Searches

In the Business Process Modeling literature, "clones" are isomorphic graphs with identical node labels (Ekanayake et al., 2013; La Rosa et al., 2015; La Rosa et al., 2011; Schoknecht et al., 2017; Skouradaki et al., 2016). Measuring and searching for similarity is a harder problem than the identification of graph isomorphism (clones) and is mentioned as a complexity NP-hard problem in the literature. Four dimensions of similarity measurement are Natural Language, Graph Structure, Behavior, and Human Estimation (Schoknecht et al., 2017).

Several business process model similarity measures have been applied to business process models. These similarity measures look at four types of scale to quantify the similarity between two graphs.

There is an implementation of similarity searches for business process models in the Advanced Process Model Repository (APROMORE) platform, which was developed through a collaboration of universities. APROMORE has a plug-in ecosystem which includes "Detect Clones" and "Search Similar Models".

Frequent Subgraph Mining (FSM) is a technique used to extract all frequent subgraphs with an occurrence count over a specific threshold. The APROMORE algorithm, which will be referred to as APROMORE Approximate Clone Detection (AACD) decomposes process models into fragments, then clusters the fragments using

graph edit distance, which is a function of structural and node label similarity (La Rosa et al., 2015).

This approach seems promising for the precedence network pattern problem. However, the technique for decomposition of networks into fragments exploits a feature of business process models which is not appropriate for the project precedence networks. The decomposition depends on the identification of Single-Entry Single-Exit (SESE) process fragments (La Rosa et al., 2015). These SESE process fragments are rarely found in the project management precedence networks used for this praxis. Another difference between the AACD approach and the approach needed for this work has to do with the input graphs. The AACD approach is searching many business process models for similar fragments, where this praxis needed to search one large precedence network for similar fragments.

An alternative algorithm discussed in a later paper is a recurring structure detection algorithm called "RoSE" (Skouradaki et al., 2016). An advantage claimed for the RoSE algorithm is that a subgraph or fragment is not needed as in input for the similarity search. However, as this algorithm was designed for anonymized process models the textual semantics of the node labels were not considered. For this praxis, the node labels are important, as they describe the activities needed for product development. As with AACD, this algorithm depends on finding a single-entry point, used as a stem for the maximal common reoccurring structure.

2.8 Measurements of "Interestingness"

In biological sequences such as Deoxyribonucleic Acid (DNA) sequences, number of occurrences is the major metric of whether a pattern is interesting (Rashid, Karim, Jeong, & Choi, 2012). Statistical approaches such as statistically significant departure from what a random sample would yield (Arora, Sachan, & Bhattacharya, 2014), and Principal Component Analysis (Saigo & Tsuda, 2008) have been applied in the frequent subgraph mining literature. Work in computational biology used departure from expected frequency of a pattern as a measure of interest for patterns. Rashid et. al. argue that interesting patterns are found by measuring difference between expectations of a pattern's frequency and the actual frequency of occurrence (Rashid et al., 2012). In a later work, Demesmaeker et. al. used a similar definition for a measure of interesting patterns in a social network (Demesmaeker, Ghrab, Nijssen, & Skhiri, 2017).

In phylogeny, an interesting pattern is related to the "cousin distance" (Shasha, Tsong-Li Wang, & Zhang). In biochemistry, an interesting pattern is the largest structure with specific properties (I. Fischer & Meinl, 2004). In Frequent Itemset Mining, a pattern is interesting if it is easily understood, valid with a degree of certainty, potentially useful, and novel (not formerly known or used). Patterns can represent knowledge. (Han, Kamber, & Pei, 2000). After consideration of the above definitions, the frequent itemset mining definition of interesting was ultimately used as a guide to search for patterns in this praxis.

2.9 Motifs

Motifs are statistically over-represented patterns in networks. These patterns occur far more frequently than are found in randomly generated networks. The randomly

generated networks used for comparison are the "null model" for statistical testing. The null models are constructed so that each node in the random network has the same number of incoming and outgoing edges (degree) as the matching node in the actual network. As early as 1989, motifs were described in nucleic acid and protein sequences (Staden, 1989). Motifs and anti-motifs (statistically under-represented patterns) were broadly introduced by Ron Milo et al. in a 2002 article in Science (Milo et al., 2002). Milo reports on motifs found in transcription interactions between regulatory proteins and genes in e-coli and yeast, in synaptic connections between neurons in nematodes, in predator-prey interactions in food webs, between logic gates in electronic logic circuits, and in networks made from hyperlinks between web pages. Examples of motifs are given in Table 2-4. The definition of the values shown in the table are as follows:

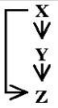
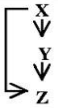
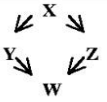
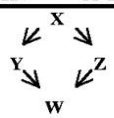
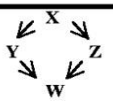
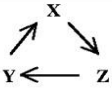
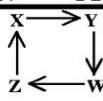
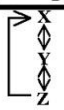
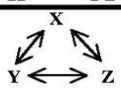
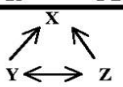
N_{real}: Number of appearances of the motif in the real network.

$N_{rand} \pm SD$: Number of appearances of the motif in randomized networks, with standard deviation.

Z-Score: Number of standard deviations the motif count is from the mean motif count found in the random networks, where:

$$z=\frac{(N_{real}-N_{rand})}{SD}. \text{ (Equation 1)}$$

Table 2-4: Network motifs described by Milo et. al. From (Milo et al., 2002). Reprinted with permission from AAAS.

Network	Nodes	Edges	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	Z score
Gene regulation (transcription)					**Feed-forward loop**			**Bi-fan**			
E. coli	424	519	40	7 ± 3	10	203	47 ± 12	13			
*S. cerevisiae**	685	1,052	70	11 ± 4	14	1812	300 ± 40	41			
Neurons					**Feed-forward loop**			**Bi-fan**			**Bi-parallel**
C. elegans†	252	509	125	90 ± 10	3.7	127	55 ± 13	5.3	227	35 ± 10	20
Food webs					**Three chain**			**Bi-parallel**			
Little Rock	92	984	3219	3120 ± 50	2.1	7295	2220 ± 210	25			
Ythan	83	391	1182	1020 ± 20	7.2	1357	230 ± 50	23			
St. Martin	42	205	469	450 ± 10	NS	382	130 ± 20	12			
Chesapeake	31	67	80	82 ± 4	NS	26	5 ± 2	8			
Coachella	29	243	279	235 ± 12	3.6	181	80 ± 20	5			
Skipwith	25	189	184	150 ± 7	5.5	397	80 ± 25	13			
B. Brook	25	104	181	130 ± 7	7.4	267	30 ± 7	32			
Electronic circuits (forward logic chips)					**Feed-forward loop**			**Bi-fan**			**Bi-parallel**
s15850	10,383	14,240	424	2 ± 2	285	1040	1 ± 1	1200	480	2 ± 1	335
s38584	20,717	34,204	413	10 ± 3	120	1739	6 ± 2	800	711	9 ± 2	320
s38417	23,843	33,661	612	3 ± 2	400	2404	1 ± 1	2550	531	2 ± 2	340
s9234	5,844	8,197	211	2 ± 1	140	754	1 ± 1	1050	209	1 ± 1	200
s13207	8,651	11,831	403	2 ± 1	225	4445	1 ± 1	4950	264	2 ± 1	200
Electronic circuits (digital fractional multipliers)					**Three-node feedback loop**			**Bi-fan**			**Four-node feedback loop**
s208	122	189	10	1 ± 1	9	4	1 ± 1	3.8	5	1 ± 1	5
s420	252	399	20	1 ± 1	18	10	1 ± 1	10	11	1 ± 1	11
s838‡	512	819	40	1 ± 1	38	22	1 ± 1	20	23	1 ± 1	25
World Wide Web					**Feedback with two mutual dyads**			**Fully connected triad**			**Uplinked mutual dyad**
nd.edu§	325,729	1.46e6	1.1e5	2e3 ± 1e2	800	6.8e6	5e4±4e2	15,000	1.2e6	1e4 ± 2e2	5000

In a later development, Milo et. al. introduced significance profiles, where a count of each type of subgraph is compared to counts of these subgraphs in random networks. The term "Motif" is reserved for those subgraphs which are statistically over-represented compared to a random graph. Normalized Z-scores for the subgraphs are plotted along

an axis of subgraph class, and "super families" emerge which have similar significance profiles (Milo et al., 2004). An example is shown as Figure 2-3.

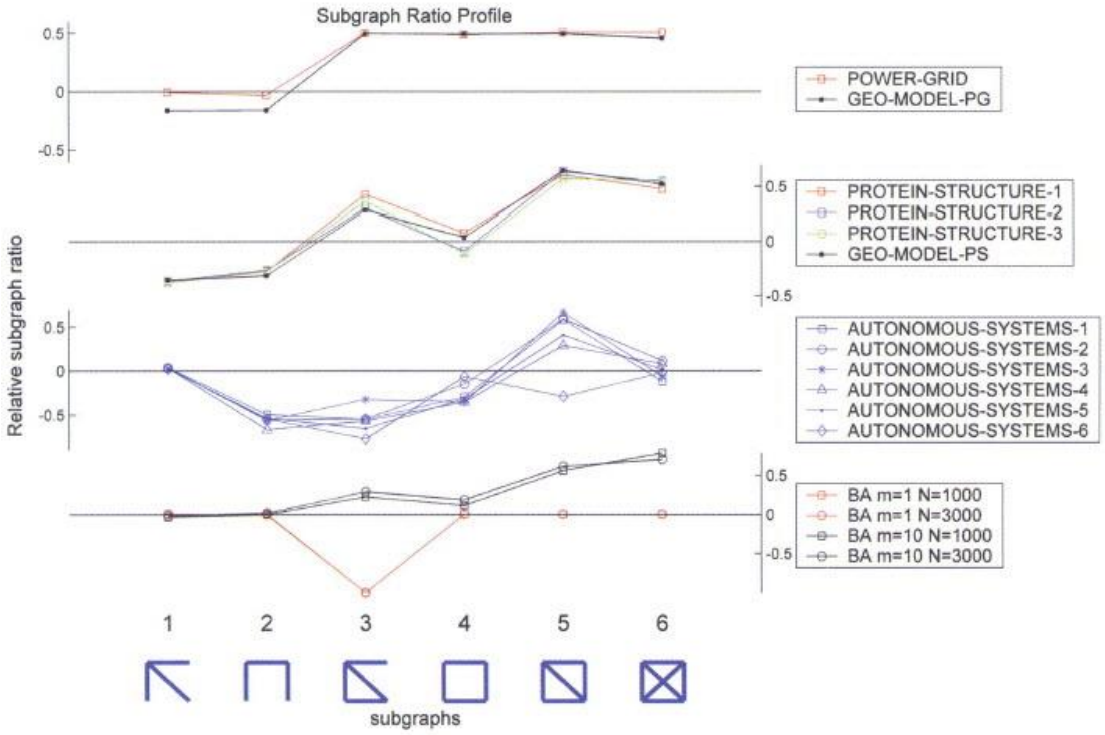


Figure 2-3: Subgraph ratio profiles for nondirected networks. From (Milo et al., 2004). Reprinted with permission from AAAS.

The networks profiled include the electrical power grid of the western United States, synthetic networks using a geometric model; protein structures, and another synthetic geometric model; the internet at the autonomous systems level, and synthetic networks developed using the preferential attachment model of Barabasi and Albert (BA).

A thoughtful critique of the randomly generated networks used as the null model in Milo's work was submitted by Artzy-Randrup et. al. published as a "Technical Comment". She and her co-authors show that in a toy network, where the probability of

an edge between two nodes increases as the distance between the nodes reduces, motifs such as feed-forward, bi-fan, and bi-parallel are detected (Artzy-Randrup, Fleishman, Ben-Tal, & Stone, 2004). Milo et al. responded to the criticism by showing the significance profiles for real networks are different than those for random networks (Milo, Itzkovitz, Kashtan, Levitt, & Alon, 2004). Konagurthu and Lesk showed that the topological structure of the network gives rise to motifs, also challenging the null model selected by Milo et al. (Konagurthu & Lesk, 2008).

Motifs are fascinating because the function of many motifs are known. In biological systems, the function of a feed-forward loop is to provide sign-sensitive delay and persistence detection, depending on the input logic. With AND input logic, delay generated by the feed forward loop can filter out brief signal pulses. With OR input logic, production continues through a transient drop of the input signal (Alon, 2007; Kashtan, Itzkovitz, Milo, & Alon, 2004).

These functions of delay and persistence detection are likely important to the information processing in a precedence network. In a precedence network for product development, the nodes are activities which are conducted to mature information from concept through design and build to finished product. This represents information processing, as the predecessors for an activity produce the information needed for the successor activity. It seems reasonable that brief signals of information should be damped out in product development, and that information production should continue through a transient drop of an otherwise persistent input signal.

2.9.1 Motifs in Engineered Systems

Software systems contain many of the motifs seen in previously analyzed biological systems, and a mechanism of evolution with imperfect copies is proposed as the reason for the motifs found in both systems (Valverde & Solé, 2005). Engineering systems can also be represented as a network. Using a network where the nodes represent system components, and edges represent connections between these components, Shaja and Sudhakar have shown that motifs in engineering systems are a consequence of design by decomposition. The have also shown how subsystems play a role in motif emergence in a social network (Shaja & Sudhakar, 2013).

This praxis seeks patterns in precedence networks. The precedence network studied in this praxis is hierarchical and includes the activities related to systems and to the decomposition of these systems. Hence, the understanding of how motifs emerge in engineering systems described above can inform the search for precedence network patterns.

2.10 Link Prediction

Link prediction was first introduced for social network analysis. Link prediction refers to the problem of predicting what links might form in the near future, given the current state of a network (Liben-Nowell & Kleinberg, 2007). Besides predicting social network links, link prediction has been used for predicting links in biological networks, and dynamic networks of email and telephone contact. (D. A. Davis & Chawla, 2011; Martínez, Berzal, & Cubero, 2017). One challenge in link prediction is the extreme class imbalance. In large, sparsely connected networks, the dominant information is lack of connections, rather than the existence of connections. Another challenge is the

longitudinal nature of the data, where the graph of interest evolves through time. Finally, "The work flow to achieve predictions and fair evaluation is time-consuming, challenging, and error prone" (Lichtenwalter & Chawla, 2011).

Link prediction is typically approached as a ranking problem, and pairs of nodes with no link are given a score which is proportional to the probability of a link forming between them. The pairs are sorted, and a cut-off determined, above which the pairs are predicted to form new links (Martínez et al., 2017).

Link prediction techniques include node-based, neighbor based, path based, random walk based, and social theory based. Well-known neighbor-based link prediction approaches which use similarity between nodes include common neighbors, Jaccard coefficient, preferential attachment, and Adamic/Adar. Cannistraci-Alanis-Ravasi index is outside the classification scheme, and has been proposed as a link/community based predictor (Cannistraci, Alanis-Lobato, & Ravasi, 2013; Wang, Xu, Wu, & Zhou, 2015). There are also social theory-based predictors, one example is triadic closure. All of the aforementioned link prediction approaches were evaluated for this praxis.

Brief definitions for the node similarity metrics used in this praxis are shown in Table 2-5. $\Gamma(x)$, $\Gamma(y)$ indicates the set of nearest neighbors of x and y respectively. When bounded by vertical lines, $|\Gamma(x)|$ indicates the cardinality of the neighbor set. $\gamma(s)$ is the sub-set of neighbors of s that are also common neighbors of x and y, i.e. $\gamma(s)$ is the local community degree of s.

Table 2-5: Node Similarity Metrics

Metric	Formulation	Equation	Reference
Adamic-Adar Coefficient	Common neighbors which have few neighbors are weighted more heavily.	$\sum_{z \in \Gamma(x) \cap \Gamma(y)}^{n} 1 / \log\lvert\Gamma(x)\rvert$	(Adamic and Adar 2003)
Cannistraci-Alanis-Ravasi Index	Communities based on links	$CN(x,y) * \sum_{s \in \Gamma(x) \cap \Gamma(y)}^{n} \lvert\gamma(s)\rvert/2$	(Alanis-Lobato, Cannistraci, and Ravasi 2013)
Common Neighbors	For two nodes x and y, the count of neighbors which are common to both x and y.	$\lvert\Gamma(x) \cap \Gamma(y)\rvert$	(Jin, Girvan, and Newman 2001)
Dice's Coefficient	Twice the count of neighbors which are common to both x and y divided by the total node count.	$\frac{2 * \lvert\Gamma(x) \cap \Gamma(y)\rvert}{\lvert\Gamma(x)\rvert + \lvert\Gamma(y)\rvert}$	(Dice 1945)
Jaccard's coefficient	The count of common neighbors divided by the node count.	$\frac{\lvert\Gamma(x) \cap \Gamma(y)\rvert}{\lvert\Gamma(x) \cup \Gamma(y)\rvert}$	(Liben-Nowell and Kleinberg 2007)
Preferential Attachment	For two nodes x and y, the product of the degrees of x and y	$\lvert\Gamma(x)\rvert * \lvert\Gamma(y)\rvert$	(Jin, Girvan, and Newman 2001)
Structural Perturbation Method	Measures how much a network's structural features change with the removal of random links.	See reference.	(Lü et al. 2015)

Proposed link prediction methods can be compared to the traditional approaches, and the area under the Receiver Operator Curve (ROC), or the Precision Recall Curve evaluated, with more area corresponding to better prediction. While ROC curves are

more common, precision recall curves are more accurate for a skewed dataset (J. Davis & Goadrich, 2006).

Much of the literature approaches the link prediction problem as an unsupervised machine learning problem, however a high-performance link prediction (HPLP) framework has been published which provides a 30% improvement over unsupervised methods for the evaluated networks. The HPLP approach is a supervised machine learning framework which uses local network measurements and node similarity measurements as features, along with bagged random forests (Lichtenwalter, Lussier, & Chawla, 2010).

Triadic Closure is a link prediction technique that was first suggested by Rapaport in 1953. He introduced the concept that two people with a mutual friend will likely be introduced, forming a new connection. This concept is called 'triadic closure' because it is represented in a social network as an open triad gaining a third edge, becoming a closed triad (Rapoport, 1953; Tselenti, 2017). An illustration of triadic closure is given in Figure 2-4. Figure 2-4 shows (1). An open triad, with the two directed edges indicating social links into the top node. In the case of a directed acyclic graph (such as used in project management precedence networks), this open triad can be closed only two ways: (2) with an edge directed to the left, and (3) with an edge directed to the right. The implications of the different closing direction would depend on the node labels. With no node labels (as illustrated), triads (2) and (3) are isomorphic.

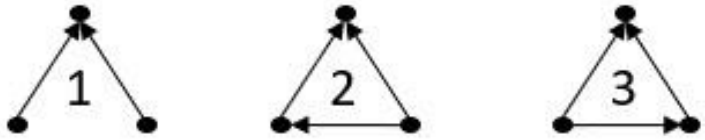


Figure 2-4: Triadic Closure Illustration (1) an open triad, with two directed edges indication information flow into the top node. (2) A closed triad with an edge directed to the left, and (3) a closed triad with an edge directed to the right.

2.11 Labeling Graphs for Mining

For frequent subgraph mining, choices must be made regarding whether and how to label the nodes of the precedence network used as input to the mining task. The task of labeling the nodes is a classification problem.

Literature search into generalization led through anti-unification theory (Bulychev & Minea, 2009; Plotkin, 1970), and eventually into hypernym-hyponym relationships. Hypernym-hyponym relations between terms or term phrases is a specific to general relation, where the hyponym is the least general term which generalizes the hypernym (Grycner, Weikum, Pujara, Foulds, & Getoor, 2015; Zhang, Li, Zhao, & Tang, 2018). Hypernym-hyponym relations were applied (in part) to classify the node labels used for this praxis.

2.12 Work Breakdown Structure Automation

The activities in a project are often elaborated through the construction of a work breakdown structure, which is a hierarchical breakdown of the activities in the project. Precedence networks can be constructed from the leaf-level of the work breakdown structure. This is accomplished by using the leaf-level of the work breakdown structure for the activities, and establishing precedence relationships between the activities. These precedence relations are established based on experience and domain knowledge.

Precedence networks for the construction of facilities (in the civil engineering domain) have been generated using general purpose and domain-specific models. For example, the OARPLAN (objects, actions, and resources plan) system was used to generate plans for low rise frame buildings. Domain-specific methods out-perform general purpose methods; however, they require integration with databases, including computer-aided design models of the product (Darwiche et al., 1988). When a model of the product is required as an input to the planner, the precedence network cannot be produced until the model is available, so the network cannot include the tasks necessary to create the product design.

Artificial neural networks have been applied to develop precedence networks; however, the input requires precedence networks from prior designs, thus limiting their scope to previously known relationships. The case study used was a project consisting of nine activities related to the design, supply, and installation of automotive components. (Golpayegani & Parvaresh, 2011).

2.13 Summary

The current body of knowledge for reuse of precedence network patterns has not considered the latest techniques from other domains. The construction industry has applied domain specific and artificial neural network techniques to work breakdown structure automation to form the nodes in a precedence network. These techniques have limited applicability to new product development. The business process modeling community has applied data mining techniques and statistical clustering using graph edit distance as a similarity measurement, but has not considered the meaning of the

structures represented by motifs, nor the similarity measurements used in social network analysis.

There has been excellent cross-domain work between social network science and bioinformatics, where link prediction based on triad census has out-performed traditional node similarity measurements (Lichtenwalter et al., 2010). Nevertheless, a triadic census with triadic closure approach was not compared to supervised machine learning to determine whether this domain-specific approach can outperform the high-performance link prediction framework. To our knowledge, link prediction techniques from social network analysis have not been applied to predicting links in project management precedence networks.

Chapter 3 Methods

3.1 Introduction

The precedence network to develop a product can be represented as a graph, with the activities as nodes, and the precedence between activities shown as edges. A commercial airplane product development network has been selected for proof of concept of an input recommendation system in this praxis. The airplane product development network used for this study has been evolved for over a decade to describe the activities involved in the design and certification of a commercial airplane. This is a hierarchical network, with five levels of detail. The detailed content of this network is discussed in Chapter 4.

In order to illustrate the nature of a hierarchical network and how it relates to precedence, a toy example of preparing to send a manned rocket to the moon will be shown.

A work breakdown structure (WBS) decomposes the work hierarchically and is illustrated in Figure 3-1. Three levels of a hierarchy are shown, a level 0, level 1, and level 2. Each item in the work breakdown structure is identified with a dot separated WBS code, where the first digit(s) is the first level, there is a dot, then each successive level digit(s), a dot, then the numbered item in the breakdown.

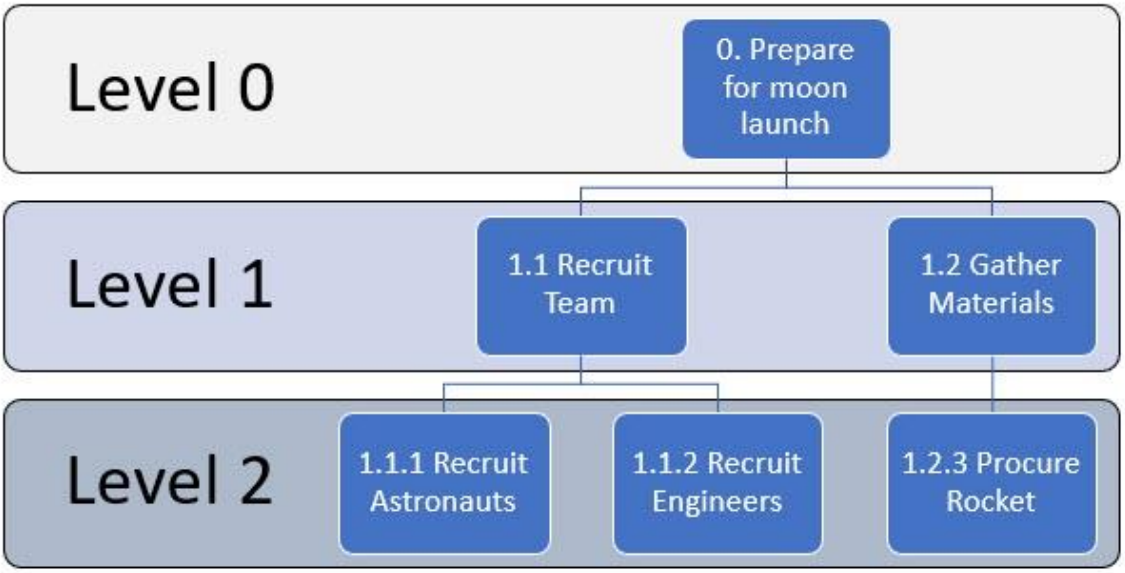


Figure 3-1: Work Breakdown Structure to prepare for a moon launch

Each level of the work breakdown structure can be developed into a precedence network, if the precedence is known. Precedence for real networks is typically established through domain knowledge or by interviewing subject matter experts. In this example, the level 1 precedence network can be formed by relating the work item 1.1 Recruit Team to item 1.2 Gather Materials. Since the team will be needed to gather the materials, a precedence relation is formed, with item 1.1 Recruit Team preceding item 1.2 Gather Materials. This is illustrated in Figure 3-2.

Figure 3-2: Level 1 precedence network

The level 2 precedence network can be formed by relating work items 1.1.1 Recruit Astronauts, 1.1.2 Recruit Engineers and 2.3 Procure Rocket. Since both the astronauts and engineers will be involved in procuring the rocket, precedence relations are established with work item 1.1.1 Recruit Astronauts and 1.1.2 Recruit Engineers preceding item 1.2.3 Procure Rocket. This is illustrated in Figure 3-3.

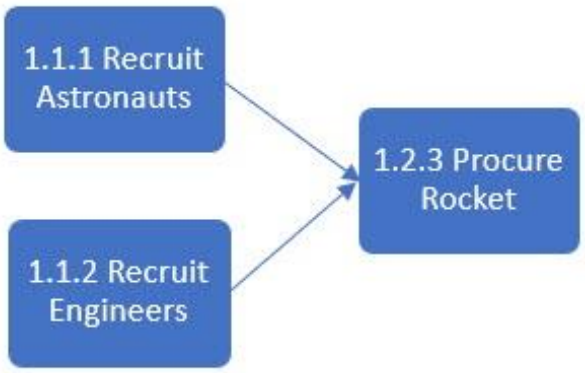


Figure 3-3: Level 2 precedence network

These precedence networks can then be combined with duration estimates for each activity, program scheduling constraints such as target finish, and resource availability to produce an executable schedule. Summary levels can be used for pre-program planning. The lowest level of detail is typically used for execution.

3.2 Research Approach

The research approach used for this work is summarized in Figure 3-4. The following elaboration refers to the parenthetical numbers in Figure 3-4. The process began by mining for frequent subgraphs (1). Implementations evaluated included GraMi, FANMOD, Kavosh, and others. For a collection of mined frequent subgraphs, the graph edit distance matrix was computed (2), and subgraphs were clustered (5). In some cases, (3) the instances of the pattern in the search graph were found, or (4) motifs were

evaluated. This led to an evaluation step regarding whether interesting patterns were discovered (6). If not, various classification schemes for node labels (8), graph partitioning approaches (7), and further literature search for algorithms were used, leading to further mining. Eventually, as discussed in Chapter 4, interesting patterns were discovered, allowing an exit from the mining process (9). As was discussed in 2.8 Measurements of "Interestingness" a pattern is interesting if it is easily understood, valid with a degree of certainty, potentially useful, and novel.

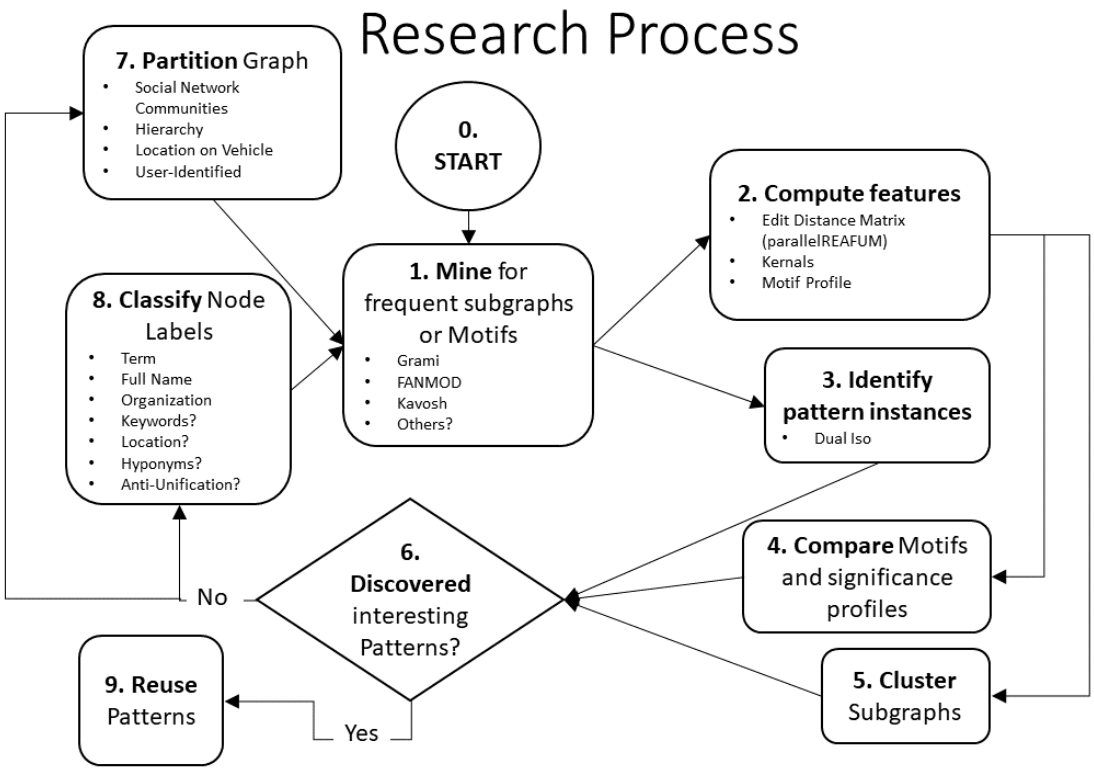


Figure 3-4: Precedence Network Pattern Research Process

A summary table comparing the main approaches used in this research is shown in Table 3-1. The reason several approaches were evaluated was that, although many experts provided input to the direction of this praxis, it was not known a-priori which

approach would be the most promising. The main approaches are all reported to provide a full report on main avenues pursued to provide feedback to these experts and to inform future research.

Table 3-1: Comparison of Analysis Techniques

Technique	Advantages	Disadvantages	Mitigation
Frequent Subgraph Mining (GraMi)	•Fast •Focus on frequent patterns •Patterns directly re-usable. •Can tune search with constraints	•Fixates on a set of labels •No null model. •Patterns omit some edges. •Pattern Meaning Unclear	•Pre-prune network •Try Alternate algorithms •bootstrap a null model if necessary. •Analyze patterns
Network Motif Finding (FANMOD)	•Statistical over/under representation against null model. •Edge permutations considered •Common motifs across disciplines with meaning understood	•Null model weakness. •Patterns cannot be used directly. •Large patterns intractable	•Discuss null model weakness. •Translate motif to labels (or colors) •Don't bootstrap
Supervised Link Prediction (LP-Made)	•Framework with 12 built-in unsupervised link prediction methods and capability to extend to domain-specific methods. •Designed for large-scale networks •Produces publication-quality Receiver-Operator Curves.	No success running on the level 3 airplane network. Reason for errors not obvious.	Postpone as Future Work.
Unsupervised Link Prediction	•Simple •Fast	•Potentially lower prediction accuracy than supervised methods.	•Implement with triadic closure initially, improve prediction later with supervised, longitudinal, and other methods using more information than just the node's term identifier.

For statistical inference, the frequency of occurrence for each pattern found in the mined network was compared to the frequency of occurrence of that pattern in a random network. Only patterns which are statistically over-represented (frequent) with a cutoff p-value of 0.05 were used for hypothesis testing.

3.3 Node Classification

In section 2.11 Labeling Graphs for Mining, various classification schemes for node labels were introduced. In the airplane product development network, the nodes have a plurality of attributes which can be used for classification. These attributes include the work breakdown structure code, the node name, a qualifier, location, verb, and the responsible organization. When mining, a spectrum of node label choices is possible. This spectrum ranges from a unique label for each node, to the same label for each node. An example of a label which is so specific as to be unique in the network is the work breakdown structure code. This would obviously not be a classification scheme at all, since each class would have one member. An example at the other end of this spectrum would be to assign all nodes the same label. This would result in structural patterns with no differentiation of the nodes. While this may provide clues about information flow within the network, it is not ideal for creating an input recommendation system. Somewhere along that spectrum of label choices is a node classification which provides an appropriately general label.

As discussed earlier, literature search identified hypernym-hyponym relationships could be used for node label classification. Hypernym-hyponym relations between terms or term phrases is a specific to general relation, where the hyponym is the least general term which generalizes the hypernym (Grycner et al., 2015; Zhang et al., 2018).

To demonstrate the hypernym-hyponym relationship of specific to least general, consider a particular species, the red fox, Vulpes vulpes: the next rank above, the genus Vulpes, comprises all the "true" foxes. So red foxes and true foxes have a hypernym-hyponym relationship.

To generalize the aerospace terms used for node names in the aerospace product development network, an initial experiment was conducted where an automated SPARQL Protocol and RDF Query Language (SPARQL) query was initiated to the WebIsALOD database (Hertling & Paulheim, 2017). This returned the top two hyponyms (by confidence) for each node name, used as the hypernym query. An example of some of the relations is shown in Table 3-2. Subjectively, some of the suggested hyponyms are not appropriate generalizations of aerospace terms. An alternative generalization was formed from the last word in each hypernym phrase. For each level of the network, a table was constructed with the top two hypernyms and the last word from each hypernym phrase. From the table with the alternative hyponyms, this researcher's judgement was used to select the most appropriate hyponym. Most of the time simply selecting the last word in the node name (with occasional generalization) was the most useful.

Table 3-2: Hypernym-Hyponym Confidence Relations

	Hyponym	**Confidence**
landing gear	aircraft component	0.816224
landing gear	accessory	0.791674
system software	system hardware	0.711787
system software	factor	0.702214
design guide	documentation	0.582163
design guide	material	0.547342
testing	support service	0.734286
testing	related function	0.687746
engine	factor	0.82158
engine	part of the aircraft	0.808049
propulsion	feature	0.586985

propulsion	function	0.586005
system architecture	issue	0.657358
system architecture	discipline	0.556865
vehicle ownership	issue	0.65897
vehicle ownership	characteristic	0.552985
furnishings	item	0.611972
furnishings	personal property	0.606703

Table 3-3 shows examples of the highest confidence hyponyms, the last word algorithm results, and the selected hyponyms which were used in the mining.

Table 3-3: Proposed and Selected Hyponyms

Hypernym	Proposed Hyponym	Last Word	Selected Hyponym
landing gear	aircraft component	Gear	Component
system software	system hardware	Software	Software
design guide	documentation	Guide	Documentation
testing	support service	Testing	Testing
engine	part of the aircraft	Engine	Component
propulsion	feature	Propulsion	Propulsion
system architecture	discipline	Architecture	Architecture
vehicle ownership	issue	Ownership	Ownership

3.4 Link Prediction with Triadic Closure, Toy Example

If a search network has all possible triads identified, and the frequency of occurrence of all possible triads in the search network are compared to the frequency of occurrence in a random network, it is possible to predict whether an edge should be added to close the triad, and in which direction the triad would be closed. This method of predicting directed edges is called triadic closure. An illustration of an open triad and triadic closure was given earlier in Figure 2-4.

To illustrate the triadic closure process, a toy target graph will be mined. A triad will be selected from the target graph, opened by removing an edge, and then the probability of a directed edge closing the triad will be evaluated.

The target graph used for the triadic closure toy example is shown in Figure 3-5.

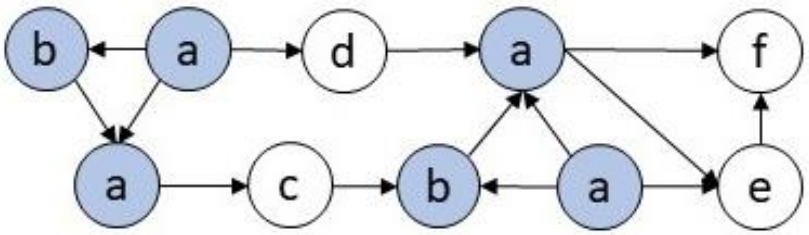


Figure 3-5: Toy example target graph

The first step in the process is to prepare the search graph for mining by assigning labels. Table 3-4 shows the toy example label assignment. The frequency of the hyponyms is computed, labels sorted by frequency, and colors assigned in descending frequency. In this toy example, there are sufficient colors to assign a unique color to each label, so the frequency sorting is not strictly necessary. In a real example, there may be more than sixteen labels. In these cases, the frequency sorting is necessary because the FANMOD tool has a limitation of sixteen colors. The frequency sorting enables assignment of colors to the top 15 most frequently used labels, reserving the color zero for the infrequent labels.

Table 3-4: Label Assignment for Tox Example.

Hypernym	Label	Selected Hyponym	Frequency	Color
Flight Deck	a	Major Assembly	4	1
Systems	b	Systems	2	2
Propulsion	c	Propulsion	1	3
Engine	d	Component	1	4
System architecture	e	Architecture	1	5
Vehicle ownership	f	Ownership	1	6

The second step is to mine the toy example for triads. This was accomplished using FANMOD, and the first third (most relevant) of the results from mining this toy example is shown as Figure 3-6.

Previous Index Next

Size-3 Network Motifs in 'EdgeListNoHypToy.txt' Page 1

Generated with FANMOD - FAst Network MOtif Detection. Written by Sebastian Wernicke and Florian Rasche

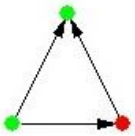
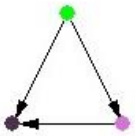
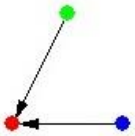
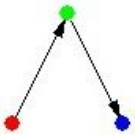
ID	Adj-Matrix	Frequency [Original]	Mean-Freq [Random]	Standard-Dev [Random]	Z-Score	p-Value
38		9.5238%	9.5238%	2.0133e-015	-0.9995	0
12		4.7619%	4.7619%	1.0066e-015	-0.9995	0
38		4.7619%	4.7619%	1.0066e-015	-0.9995	0
36		4.7619%	4.7619%	1.0066e-015	-0.9995	0
12		4.7619%	4.7619%	1.0066e-015	-0.9995	0
12		4.7619%	4.7619%	1.0066e-015	-0.9995	0

Figure 3-6: Partial results of mining toy example.

The third step is to use the mining results to predict links. For illustration with this toy example, a link which already exists in the toy network will be removed then predicted. Figure 3-7 shows a triad extracted from the toy network, color coded, and opened by removing the edge directed from green to red at the base of the triad. When this figure is compared to the mining results in Figure 3-6 and the remaining results in Appendix A, there is no open triad with two green nodes and one red node. However, the top result for this colored triad has an edge directed across the base of the triad, from green to red. Therefore, a link is predicted directed from node a to node b in this toy example. The mining results for that triad returned a frequency of 9.5%, so this proposed edge is assigned a probability of 9.5%. When a set of predictions are made, they are ranked by probability, and the top predictions (above some cutoff) are used. This was the highest probability of any triad found in this toy network, so it would be ranked in the top tier of predictions.

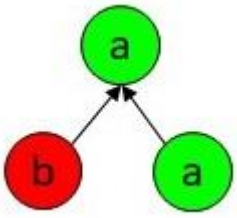


Figure 3-7: Open triad, color coded per color assignments

3.5 Time and Cost

A technique was needed to compare the time and cost results for order-driven versus schedule driven execution. Monte-Carlo analysis is a technique named after the Mediterranean gambling resort. To conduct a simulation, a sample is drawn from a

defined distribution. A set of samples constitute a trial, and the trials are repeated to determine the results.

For a Monte-Carlo simulation of a project, a sample is drawn for each task in the project network. The completion times for each task are aggregated to find the completion time for the entire project. This constitutes a trial, and 40 trials were used in this praxis to compare execution approaches. A two-sample t-test was used to compare the results. One of the assumptions for the two-sample t-test is that the population is normally distributed. This may be tested with a Shapiro-Wilk normality test, and as there was a failure, the non-parametric Wilcoxon signed-rank (AKA Mann-Whitney) test was used instead.

3.6 Summary

Methods to find patterns and predict inputs were discussed, including frequent subgraph mining and motif detection. The most promising approach, motif detection combined with the triadic closure approach from social network analysis was demonstrated with a toy example. This praxis proposes that the above method of motif detection combined with triadic closure can be used to predict precedence in a project management network. This will be used to prove in part **Hypothesis 2***: If we identify and re-use patterns, then the time required to produce well-ordered precedence networks will decrease*. The triads identified through mining are patterns, and these patterns can be used to close open triads, thus predicting inputs and reducing the time to produce well-ordered precedence networks.

Chapter 4 Results

4.1 Introduction

Prior to introducing the network used for this study, some network theory and terminology is needed. **Degree** is the sum of inputs and outputs to a node, and **average degree** is the sum of the inputs and outputs for all nodes, divided by the number of nodes. **Density** is the ratio of the number of edges (L) to the number (N) of possible edges between node pairs (Watts & Strogatz, 1998).

A regular **lattice** is a network with regular connections between nodes, such that neighboring nodes are connected through regular and nearby connections, as illustrated in the network at the left of Figure 4-1. A **random network** has connections between nodes distributed at random, illustrated in the network at the right of Figure 4-1. A "**Small-world**" networks lies between the two extremes of a lattice and a random network, as illustrated in the network in the center of Figure 4-1.

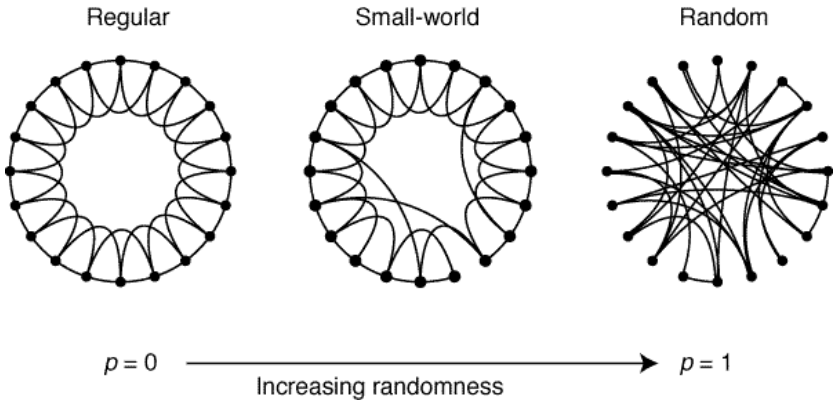


Figure 4-1: Connection topology spectrum. Reprinted with permission from Nature: Watts, D. J., & Strogatz, S. H. Collective dynamics of 'small-world' networks, 1998.

The term "small-world" comes from the observation that in social networks, a connection can be made between any two individuals in six or less hops (Watts &

Strogatz, 1998). Small-world networks are highly clustered like a regular lattice, however, like random graphs they have small **characteristic path lengths**. Characteristic or average path length is the average number of steps it takes to get from one node to another.

The airplane product development network used for this study has evolved for over a decade to describe the activities involved in the design and certification of a commercial airplane. The network was developed initially by 22 precedence network modelers over the course of 18 months, with the assistance of 660 subject matter experts. Since that time, the network has been evolved and re-used with feedback more than 30 times in the cross functional execution of a subset of the activities conducted for airplane product development.

This is a hierarchical network, with five levels of detail. Level one of this network was not included in the study, as it has only 11 nodes and 31 edges. The level one network is too small and simple to gain value from a study of patterns. Level two of this network has 357 nodes and 3,351 edges. The average degree for level two is 18.8. Most real-world networks (other than lattices) are expected to have low density Such networks are termed “sparse”, and this level two network is sparse, as L/N (N -1) (density) is 0.03. A common technique used in network analysis is to plot the degree distribution of the network on a log-log plot for evaluation. Real networks are expected to be sparse, but have a relativity few nodes with high degree, acting as hubs to provide “short cuts” to reduce the distance between neighborhoods (Watts & Strogatz, 1998). The degree distribution for the level two network is shown in Figure 4-2.

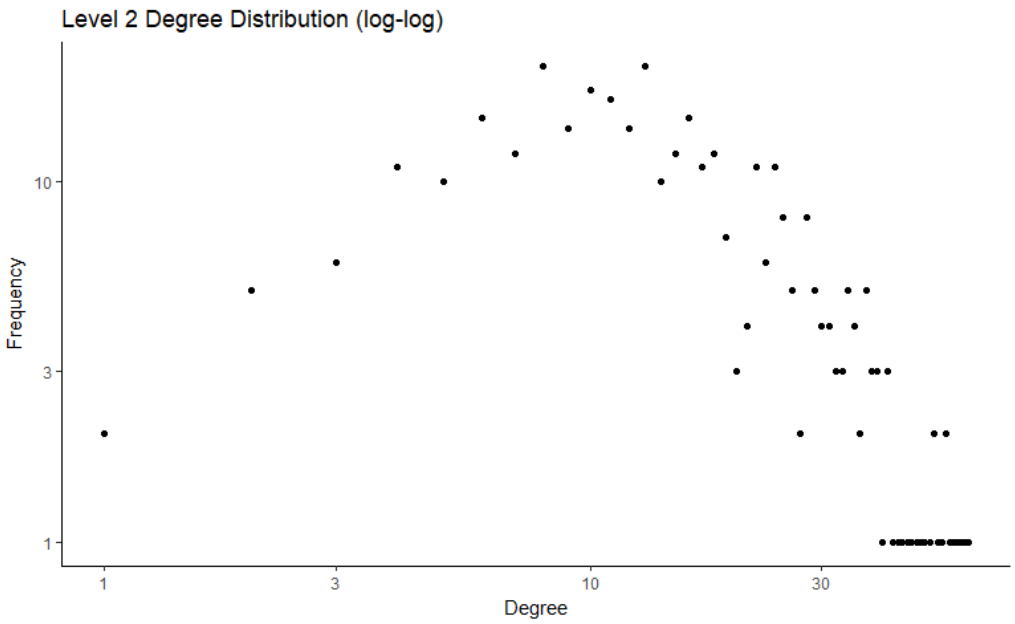


Figure 4-2: Degree Distribution for level two of the airplane product development network

Figure 4-2 shows that there are few nodes with low degree, most nodes have a degree in the range of 5-20 (recall the average degree is 18.8), and then the frequency drops off above degree 20. This shows a bias for higher degree typical of small world networks.

Level 3 of this network has 4847 vertices and 44,043 arcs, with a density ~ 0.002. This level three network has an average degree of ~ 18, which together with the sparse measure suggests there must be a few nodes with very high degree.

Figure 4-3 shows this information for level three. This plot takes on the same general shape as the level two plot, however there is an outlier at degree one, with a frequency around thirteen. This was due to incomplete modeling of the level three precedence network, and has been resolved in later evolutions of the studied network. This incomplete modeling can result in open triads which should be closed. This

condition can result in link predictions which are incorrectly classified as false positives, and will be discussed later.

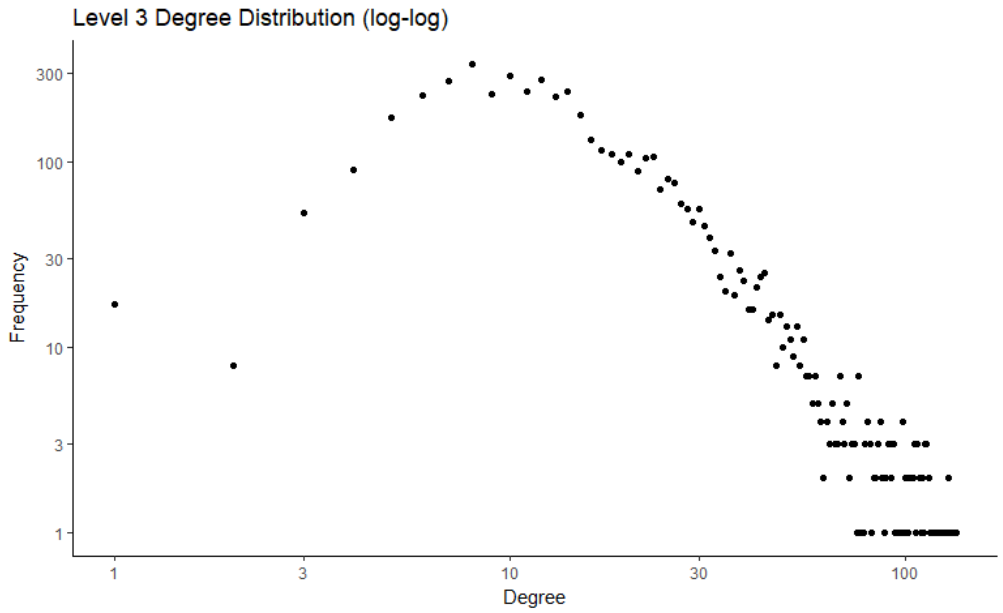


Figure 4-3: Degree distribution for level three of the airplane product development network

The level four network has 18,173 nodes and 117,316 edges. The average degree is ~13, and density is 0.0004. Figure 4-4 is shown below. We see again an outlier with degree one, this time at a frequency of 1,000. In actual product development, it is rare that a task can be completed based on just one input, so these degrees one activities are an indication the modeling is incomplete at level four. This is not surprising, it is known by the developers of this network that it is incomplete at levels four and five.

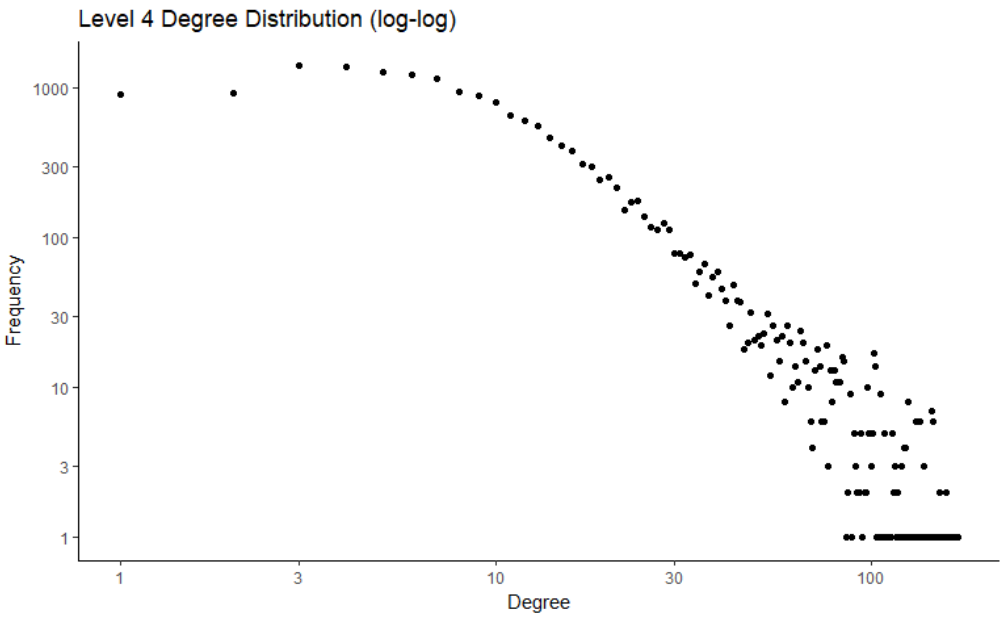


Figure 4-4: Degree distribution for level four of the airplane product development network

Finally, level five network has 6685 nodes and 90,279 edges. The average degree is 27, and density is 0.002. Figure 4-5 is the degree distribution for level five. This degree distribution does not suffer from the low-degree outliers seen at levels three and four. This level-five degree distribution also lacks the defined shape seen at levels three and four. Level five is mostly incomplete, which may be why the shape of the degree distribution lacks definition compared to level three and level four.

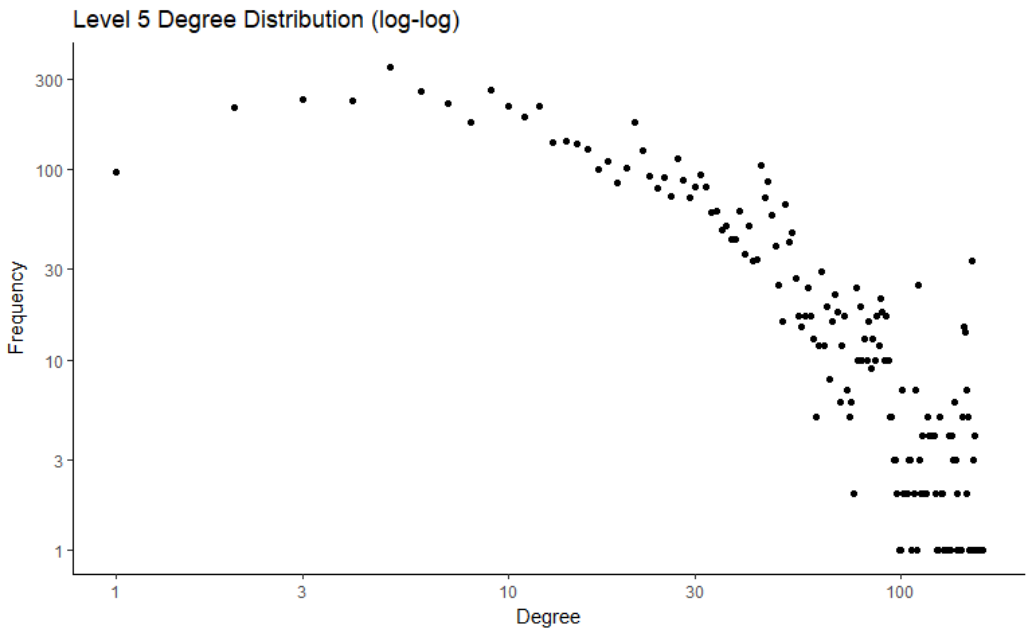


Figure 4-5: Degree distribution for level five of the airplane product development network

As is typical of a small-world network (Vespignani, 2018; Watts & Strogatz, 1998), most nodes in the airplane product development network studied here have a low degree, while a few nodes have a very high degree. Many real-world networks have been shown to be small-world (Vespignani, 2018; Watts & Strogatz, 1998). Another characteristic of real-world networks is that they tend to have a power-law degree distribution, and such networks are called "scale free"(Artzy-Randrup et al., 2004; Valverde & Solé, 2005; Vespignani, 2018; Watts & Strogatz, 1998; Wong, Baur, Quader, & Huang, 2011). A power law degree distribution means that the frequency varies as the power of the degree. A 2014 analysis found this airplane product development network to be small-world and scale-free (Raecker, 2014).

4.2 Successful Experiments

The literature classes networks into small, large, very large, and massive. Large networks consist of several thousands or millions of vertices (Batagelj & Mrvar, February 12-16, 2003). To the best of our knowledge, the other network size classes are not quantified. As the level four of the airplane product development network used for this study has over 18,000 nodes, algorithms for large graphs are needed. High-performance distributed computing frameworks such as MapReduce and Apache Spark are used for subgraph mining of large graphs in the literature. Such systems have a drawback in development and maintenance cost compared to using standard computers.

For this Praxis, an algorithm was sought which could mine a single large graph without special-purpose computing resources.

Three implementations of algorithms for subgraph mining were evaluated in this work: APROMORE, GraMi, and FANMOD; also, algorithms were written in Java for a triadic census and triadic closure prediction.

As discussed earlier, the APROMORE system was eliminated from the evaluation as it is designed for a database of process models and uses a method for creating fragments which is not applicable to project networks. The GraMi (Graph Mining) algorithm was next evaluated as it was designed for a single large graph, faster than most competing algorithms, was implemented in the same language as the target system (Java), and was reported to run on an ordinary PC with 4 Gigabytes (GB) of random access memory (RAM) for most of the tested datasets (Elseidy et al., 2014). After reproducing results for some of the smaller graphs in (Elseidy et al., 2014), trials were

conducted as depicted in Table 4-1: Network Characteristics and Mining Results. These trials were conducted on a HP-15-bs070wm laptop with a 2-core Intel I5-7200 X64, running Windows 10 Home v1709 at 2.50 gigahertz (GHz), with 8GB of random access memory. The first trials would run out of memory and crash Java before returning results. Since the algorithm prunes off low frequency nodes, best results were obtained by mining at high support frequencies, then gradually lowering the support frequency.

Another step-improvement came by switching to an approximate graph mining algorithm, AGraMi. AGraMi uses a time-out during search to avoid long-running, exhaustive searches. Hence, AGraMi will not have any false positives, but it may miss some frequent graphs (Elseidy et al., 2014). Reasonably large patterns returned, but the node labels had no variety.

An analysis of existing threads used for subject matter expert review indicated that the label variety should be around 80%. This was accomplished with a constraint "L2", which allowed a label to appear at most twice in the results, which is 20% for pattern sizes of around ten nodes. Another user constraint imposed was a specific area of interest to search. This was accomplished by constructing a list of labels of user interest, finding all nodes that had that label, expanding to all direct predecessors and successors of those nodes, and exporting this pruned network for mining.

When running GraMi, it was observed from watching the intermediate results on the screen that at low support frequencies, large patterns would appear on the screen early in the runs, but the final results were written only when all results were returned, after running as long as four days. So, the algorithm was modified to accept an overall run-time time constraint as an input. The time-constrained algorithm yielded four large (17

node) interesting subgraph isomorphs in two minutes of mining, and over two dozen 17 node subgraph isomorphs in 30 minutes of mining on a standard PC. A random sample of six of the 17 node patterns is shown in Appendix B. As with the AGraMi algorithm, an exhaustive search is omitted in favor of some search results returned within a short time budget.

As there were many mining experiments with different support frequencies, varying constraints, and various network levels and node label assignments, much of the mining work was switched to the Colonial One high performance computing center to run the mining experiments in parallel. Most of the mining work was competed on the generation one nodes, using a Dell C8220 Dual 8-Core Intel Xeon E5-2670 running CentOS 6.7, at 2.6GHz (The George Washington University, 2013). Nodes can be selected with memory size of 64, 128, or 256 GB. Most of the mining work was completed on the 64GB nodes, with increased memory nodes selected when needed for low support frequencies. As the support frequency is lowered, many more graphs are returned, which increases the need for random access memory. Selected results of mining levels one through five of the airplane product development networks are shown in Table 4-1: Network Characteristics and Mining Results.

A drawback to the GraMi algorithm is the lack of concrete instances of the pattern in the searched graph. For efficiency, GraMi does not keep track of the source nodes and edges in the searched graph. To address this drawback, a fast and efficient subgraph isomorphism search algorithm, DualIso was identified (Saltz et al., 2014). After testing on toy and sample graphs provided with the installation, instances of the pattern in the large search graphs were found with a maximum of two hours of search time. On

occasions, the algorithm does not properly return the matrix results, but the results can be found by reading through the screen output.

Table 4-1: Network Characteristics and Mining Results.

Network Level	Nodes	Arcs	Frequency	No. of subgraph isomorphs	No. of nodes in largest pattern	Computing Time, minutes
1	11	31	2	0	0	0.0
2	357	3,351	6	2,535	11	0.5
3	4,847	44,043	259	49	14	88.5
4	18,173	117,316	184	474	13	93.8
4p*	1,042	18,029	4	2,475	17	30.0
5	6,685	90,279	394	23	9	3.7

*4p: pruned network containing nodes related to mid-body structure.

4.3 Failed Experiments

Several GraMi algorithm modification attempts failed and are included here for completeness:

Experiments were conducted with constraining results to a minimum pattern size. However, this pruned out small patterns, and since the large patterns are grown from small patterns, no results were returned.

Attempts to prune out results which were not contained in a list of labels loaded from a file did not help. It is likely that the GraMi algorithm is already quickly pruning on frequency.

Pruning of the network before searching was tried to allow for a smaller search. Results were unchanged. Nevertheless, this approach led to the pruning of the search area to that of user interest, before mining.

4.4 Analysis

In order to analyze the results, a method was needed to compute the graph edit distance. This is the minimum-cost series of edits needed to transform one graph into another (Fankhauser, Riesen, & Bunke, 2011; A. Fischer, Suen, Frinken, Riesen, & Bunke, 2015; X. Gao, Xiao, Tao, & Li, 2010; Ibragimov, Malek, Guo, & Baumbach, 2013; Neuhaus & Bunke, 2007; Riesen, Emmenegger, & Bunke, 2013). The algorithm F24Threads, the winner of a graph edit distance contest (Abu-Aisheh et al., 2017; Lerouge, Abu-Aisheh, Raveaux, Heroux, & Adam, 2015) seemed a promising candidate, and the authors were kind enough to make their source code available. However, a transformation of the pattern graphs to the Graph Exchange Language (GXL) format proved troublesome. Further search led to ParallelREAFUM, (Li & Wang, 2015) a parallel graph edit distance algorithm implemented in Java. This produced a fast calculation of the graph edit distance matrix, which enabled clustering and Principal Component Analysis (PCA), shown in Appendix C. Other than that these are frequently occurring subgraph isomorphs, it was difficult to understand the significance of these frequent subgraphs. This led to a search for meaning.

4.5 Patterns in project management networks: A search for meaning.

Considerations of the raw network structure (ignoring labels) led to a dissertation (Ahmed, 2015) which acted as a gateway paper into motifs. Network motifs are subgraph isomorphs which are statistically over-represented compared to their frequency in a random graph (Milo et al., 2002).

Using the software tool "FANMOD" (Wernicke & Rasche, 2006), a search was conducted in the structure (ignoring labels) of the Level 4 airplane product development

precedence network for motifs. Of 15 statistically significant motifs discovered, two size four motifs were discovered which were identified by Milo are shown in Figure 4-6.

Generated with FANMOD - FAst Network MOtif Detection. Written by Sebastian Wernicke and Florian Rasche

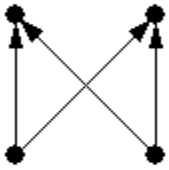
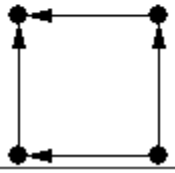
ID	Adj	Frequency [Original]	Mean-Freq [Random]	Standard-Dev [Random]	Z-Score	p-Value
204		1.2102%	0.15293%	0.00013562	77.959	0
2182		0.24112%	0.059234%	7.3541e-005	24.732	0

Figure 4-6: BiFan (#204) and Bi-Parallel (#2182), two of fifteen motifs found in the Level 4 Airplane Product development Network.

The corresponding structures shown by Milo were provided earlier in Table 2-4. A normalized z-score indicates how over-represented or under-represented a triad is compared to random networks. By creating the normalized z-score for each type of triad in a network, a profile can be created for that network. In a 2004 paper, Milo et. al combined the set of all possible size three triads (there are 13) into a “Triad significance profile”. Milo groups similar profiles into "Superfamilies" (Milo et al., 2004).

Again using "FANMOD", a triad significance profile for Level 1 to Level 5 was created in the airplane product development network and is shown in Figure 4-7 at the top of Milo's figure, identified with an illustration of a commercial airplane. The uppermost superfamily in Milo's figure includes "*...sensory transcription networks that*

control gene expression in bacteria and yeast in response to external stimuli". The airplane product development superfamily shows strong presence of Triad #7, the feed forward loop. This loop has been shown to perform signal-processing tasks such as persistence detection, pulse generation, and acceleration of transcription responses (Milo et al., 2004).

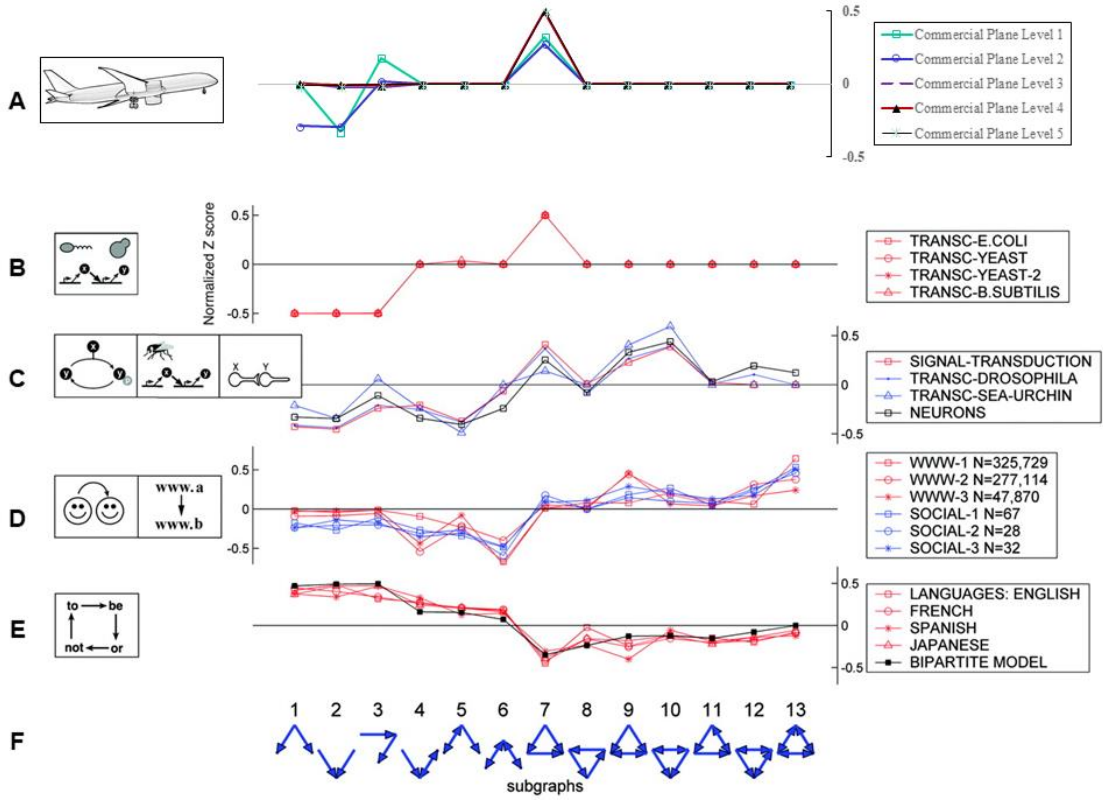


Figure 4-7 The Triad significance profile (TSP) of Networks from Various Disciplines: A: Button Praxis; B-F from (Milo et al., 2004). B-F Reprinted with permission from AAS

The airplane product development network at the top of the figure shows the strong presence of the characteristic feed forward loop (Triad 7) seen in the super family of bacteria transcription networks e-coli and yeast. In the bacteria transcription networks, the feed forward loop performs signal processing tasks including persistence detection

and pulse generation. Milo notes the bacteria transcription super family network represents information processing which is rate limited, constrained by the response time of their components. Of note is the nominal to over-representation of Triad 3, the 3-chain in the commercial plane network compared to the under-representation in microorganism networks. Under-representation of Triad 3 in microorganisms supports faster response times to signals (Milo et al., 2004). This suggests that the airplane product development network has a relatively slower response time to signals.

A further search for meaning in these patterns led to a paper looking at motifs in complex engineering systems (Shaja & Sudhakar, 2013). A key finding was provided in this paper:

"*One major design rule in complex engineering systems is 'design by decomposition' that is invoked to conquer complexity. System is decomposed into subsystems (and recursively so for very complex systems) such that nodes within each sub-system is densely inter-connected and nodes from across sub-systems are sparsely inter-connected*." (Shaja & Sudhakar, 2013).

Shaja and Sudhakar created a profile of motifs to show that if a random network is decomposed into k clusters, where connections within a cluster are higher probability than connections across clusters, the motif profile matches the motif profile of several engineering system architectures. They used Walktrap community detection algorithms and a system modularity index to find community grouping in their architecture graphs. This community detection approach provided a key idea for decomposition of the airplane product development network. Once the network is decomposed into communities, motifs found in multiple communities are promising candidates for reuse.

4.6 Social Network Analysis

To aid in demonstrating that motifs found in one region of the network could be re-used in another region of the network, social network analysis was used to decompose the network. Social network analysis detects community structures in networks by removing edges from the original graph with the highest betweenness. However, in this initial algorithm the direction of the edges is ignored (Girvan & Newman, 2002). For this praxis a directed graph algorithm was needed. The info map algorithm implemented in the Igraph package in "R" yielded a community structure, but this implementation is old and has not been updated due to licensing issues (Csardi & Nepusz, 2006; Nepusz, 2012). This resulted in communities which had many disconnected members. Ultimately, the latest implementation of the info map algorithm available on the mapequation.org website was used (Edler & Rosvall, 2014). The info map algorithm partitions a network into modules where a random walker spends more time within a module than crossing between modules (Rosvall, Axelsson, & Bergstrom, 2009). Using the info map algorithm, the level three airplane product development network was partitioned into 147 communities.

FANMOD is capable of mining for "colored" motifs, where up to 16 colors can be used to differentiate the node labels (Rasche & Wernicke, 2006). After constructing a hypernym-hyponym table and converting the 15 most frequent hyponyms to colors (the 16^{th} color is reserved for the remaining less frequent hyponyms), the eleven largest communities were mined for motifs with FANMOD. The color key for these colored motifs is provided as Figure 4-8.

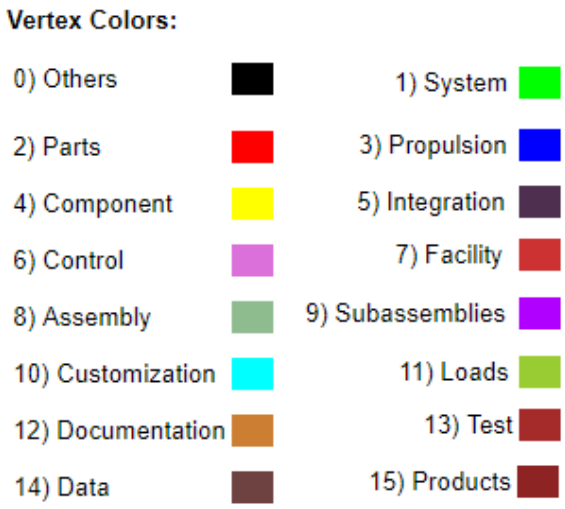


Figure 4-8: Vertex color to hyponym assignment for level three airplane product development network

The settings used to mine this network, along with the network characteristics are given in Figure 4-9.

Size-3 Network Motifs in 'EdgeList_Level_3.txt'

Generated with FANMOD - FAst Network MOtif Detection. Written by Sebastian Wernicke and Florian Rasche

Network

Filename: 'EdgeList_Level_3.txt' (Directed)
Number of nodes: 4847
Number of edges: 44043 (44043 single, 0 bidirectional)
Number of vertex colors: 16

Algorithm

Size of subgraphs: 3
Algorithm: enumeration

Random Networks

Number of random networks: 1000
Random model: with locally constant number of bidirectional edges, (regarding vertex colors)
Edge exchange parameters: 3 exchanges per edge, 3 tries per exchange

Algorithm Statistics

Running time: 1945.13 s (233.25 s for randomization, 1711.88 s for subgraph detection)
Number of detected subgraphs: 1685473278 in total (1552470 in original network, 1683920808 in random networks)
Edge exchanges: 159904416 attempts of which 129397806 (=80%) were successful

Output Parameters

Subgraphs per file: 20
Subgraph sorting: By |Z-Score|
Handling of subgraphs not detected in random networks: Put at end of list
Output Filters:
Ignored subgraphs for which |Z-Score| <= 2
Ignored subgraphs for which the P-Value >= 0.05
Ignored subgraphs that were found less than 5000 times

Figure 4-9: Network characteristics and FANMOD settings used to mine level three airplane product development network

An example of the mining results is included as Figure 4-10.

Size-3 Network Motifs in 'EdgeList_Level_3.txt' Page 1

Generated with FANMOD - FAst Network MOtif Detection. Written by Sebastian Wernicke and Florian Rasche

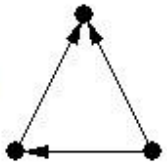
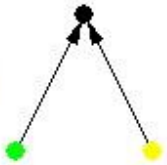
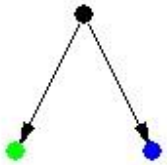
ID	Adj	Frequency [Original]	Mean-Freq [Random]	Standard-Dev [Random]	Z-Score	p-Value
38		0.90649%	0.052925%	1.9828e-005	430.49	0
38		0.35608%	0.053437%	2.2551e-005	134.21	0
6		0.39969%	0.37529%	2.5738e-006	94.771	0
36		0.33849%	0.31708%	2.8808e-006	74.343	0
6		0.50262%	0.4749%	4.0827e-006	67.894	0
6		0.3468%	0.33371%	4.3108e-006	30.376	0

Figure 4-10: Colored triads found in level three airplane product development network

As can be seen by the frequencies in Figure 4-11, the same motif is found in up to five communities.

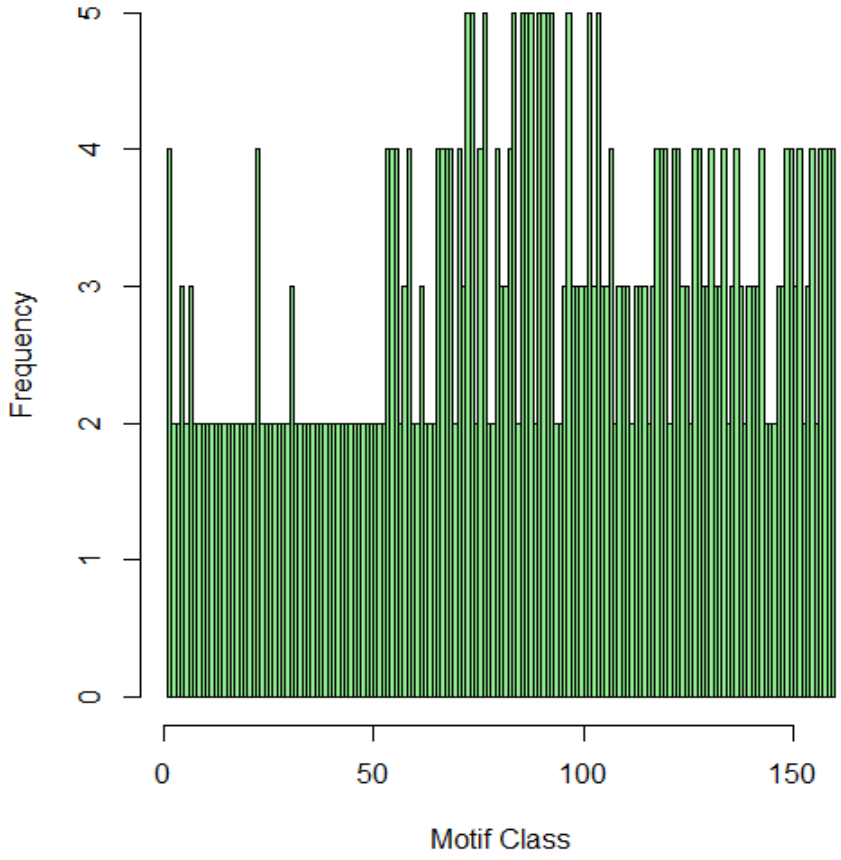


Figure 4-11: Motifs Across Communities

Z-scores and p-values are shown in Table 4-2. This table contains the subset of the results in Figure 4-11, filtered down to p values < 0.001 and z-scores > 20.

Table 4-2: Statistics for Motifs Found Across Communities

Motif Class	Z.Score	p.Value	Community
1	21.690	0	C27
1	26.843	0	C12
2	23.193	0	C23
2	23.478	0	C24
2	24.344	0	C12
2	24.697	0	C25
3	24.546	0	C24
3	26.059	0	C25
3	27.868	0	C23
3	29.442	0	C12
4	21.156	0	C27
4	31.281	0	C24
4	33.206	0	C25
4	34.759	0	C23
4	36.683	0	C12
5	25.646	0	C25
5	26.213	0	C24
5	28.802	0	C23
5	30.153	0	C12
6	32.801	0	C27
6	33.484	0	C24
6	39.423	0	C25
6	43.390	0	C23
6	44.726	0	C12

The fact that statistically significant motifs are found in up to five communities supports Hypothesis 1: There are patterns of tasks and inputs in well-ordered product development precedence networks. It is not yet obvious how these patterns can be applied to **Hypothesis** 2*: If we identify and re-use patterns, then the time required to produce well-ordered precedence networks will decrease*.

The first challenge is to develop a method to reuse these patterns. Advisor guidance to develop a toy example provided the impetus for a further literature search, and the discovery of the well-known link prediction problem in social network analysis.

4.7 Link Prediction

Link prediction was first introduced for social network analysis. This addresses the problem of predicting what links might form in the near future, given the current state of a network (Liben-Nowell & Kleinberg, 2007).

Gao et al. studied the correlation between link prediction methods and their accuracy with different social networks (F. Gao, Musial, Cooper, & Tsoka, 2015), and their results are presented in combination with results from the level 1-5 of the airplane product development network as Table 4-3.

Table 4-3: Networks ranked by predictability

Dataset	GCC	ACC	Diameter	ASP	AvgRank
APD_Level_1	0.7725	0.8143	4.0000	1.5818	1
APD_Level_2	0.2963	0.3967	9.0000	2.8250	3
APD_Level_3	0.1443	0.3058	19.0000	4.3044	6.5
APD_Level_4	0.1480	0.3091	33.0000	7.3391	7
APD_Level_5	0.0763	0.2314	15.0000	4.3519	7.25
Flickr	0.0658	0.3294	6.0000	2.3447	15
YouTube	0.0286	0.2838	5.0000	3.0709	16
Enron	0.0290	0.1946	10.0000	3.6818	17
UC Irvine	0.0197	0.1075	7.0000	3.0463	18
Facebook	0.0341	0.1176	16.0000	5.7235	19
PWr	0.0048	0.2666	16.0000	4.0162	20

In Table 4-3, APD is the airplane product development network studied in this praxis. For details of the other networks and formulation of the metrics, please see (F.

Gao et al., 2015). GCC is the global clustering coefficient and is the ratio of closed triads to possible triads. ACC is the average of the clustering coefficient for each node. From the set of all shortest paths in the network, Diameter is the maximum, and average shortest path (ASP) is the average (of all shortest paths). The average rank shows that the airplane product development network is expected to be more predictable than the other networks studied and that the network becomes less predictable with increasing level. This is expected as the first three levels of the network are nearly complete, while levels four and five are substantially incomplete.

A triadic closure link prediction algorithm was implemented in Java and used to predict links for level 2-5 of the airplane product development network. Pseudocode is a way to describe algorithms in structured English (Dalbey, 2003). The main algorithm and subroutines for predicting links using triadic closure, expressed using Dalbey's pseudocode standard is provided in Appendix D.

There are two versions of the method predictEdges. The version predictEdges_TCCA (meaning Triadic Closure Close All) predicts closing edges for all open triads. This is provided in Appendix D, Algorithm D-5 predictEdges_TCCA routine. The modification predictEdges_TCPC (meaning Triadic Closure Percent Close) of this algorithm considers open triads which should remain open. This is provided in Appendix D, Algorithm D-6 predictEdges_TCPC routine

Run times for the Triadic Closure Close All (TCCA) algorithm on the on a Dell Precision M4800 laptop i7-4810MQ CPU @2.80 GHz with 16GB memory and running Windows 10 Enterprise 64 bit v. 1709 build 16299.611 are given in Table 4-4. Run times were similar for the Triadic Closure Percent Close. As described in Algorithm D-1:

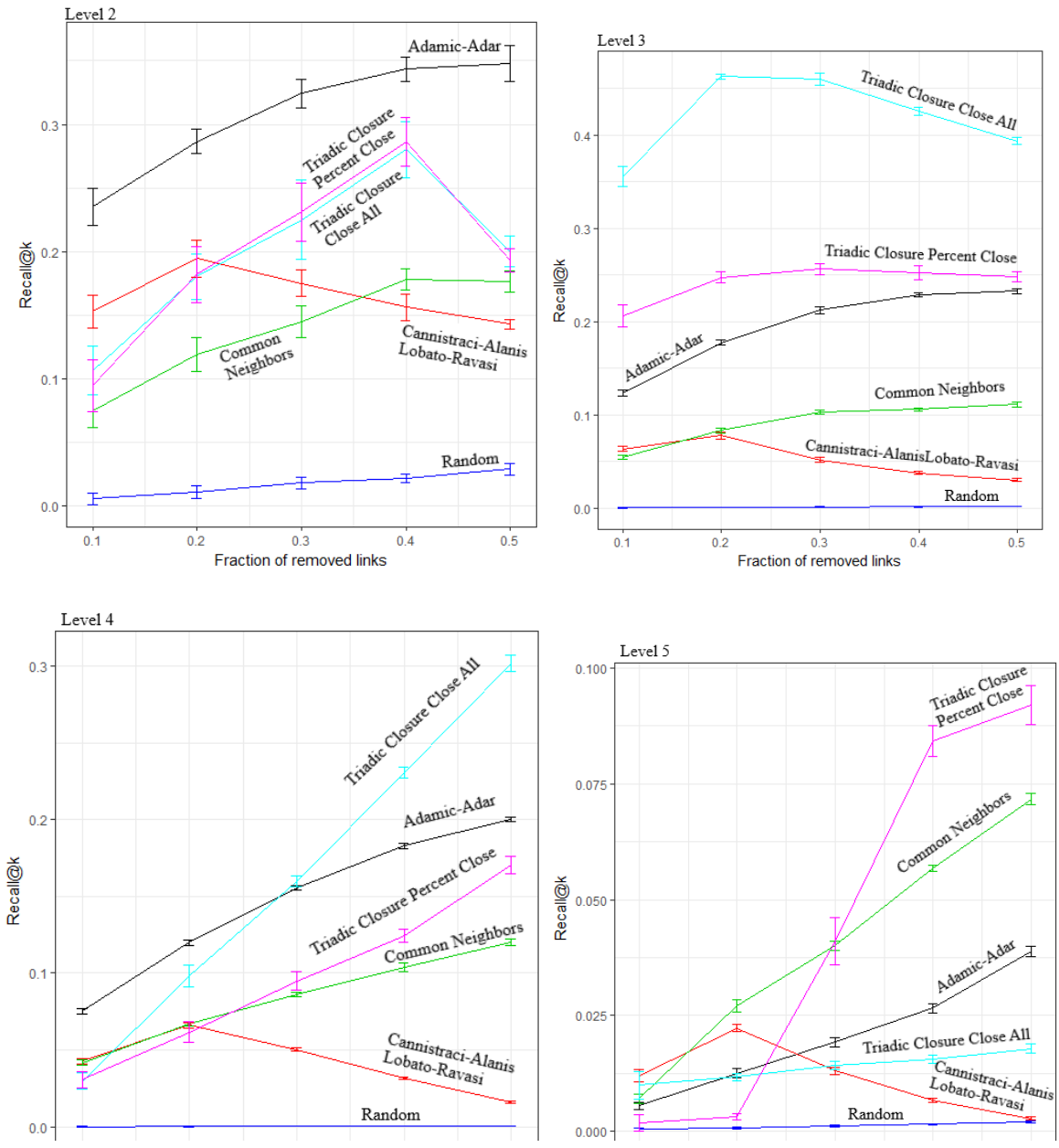
Triadic Closure Main Routine, each run includes removing a fraction of random edges five times, then predicting and measuring the predictions for the entire level network ten times for each level. These times are not representative of response times when using triadic closure for prediction in practice. For this study predictions are made 50 times for each network level.

Table 4-4: Run times for Triadic Closure Close All

Dataset	Run Time, minutes
APD_Level_2	27
APD_Level_3	89
APD_Level_4	59
APD_Level_5	184

4.7.1 Link Prediction Results

The "R" library "LinkPrediction" implements many common link predictors and provides visualizations to compare the predictors for a network of interest (Alanis-Lobato, 2017). This facilitates the comparison of triadic closure with other common predictors. The results of these comparisons to Triadic Closure Close All (TCCA) and Triadic Closure Percent Close (TCPC) are given as Figure 4-12. The remaining algorithms were documented earlier. Run times varied from a few minutes to ten days on the high-performance computing center. The results from the traditional predictors are shown with the triadic closure results for comparison.



These plots show recall@k vs the fraction of removed links. Recall@k is the fraction of top candidate links that are in the set of removed edges (Alanis-Lobato, 2017), interpreted as prediction accuracy. Figure 4-12 shows that the prediction accuracy depends on the network level, and at times, the fraction of removed links. For level two, Adamic-Adar provides the best predictions at all tested fractions of removed links, ranging from 25% to 35% accurate. For level three, Triadic Closure Close All (TCCA)

provides the best results at all fractions of removed links, ranging from 35% to 45% accurate. Best prediction for level four and level five depends on the fraction of removed links. For level four, from a 10% to 30% fraction of removed links, Adamic-Adar again is the best predictor, with accuracies ranging from 8% to 15%. Above 30% removed links, Triadic Closure Close All provides the best prediction accuracies, ranging from 15% to 30%. With level five, there are three predictors which are the best, depending on fraction of removed links. Cannistraci-Alanis-Lobato-Ravasi is best from 10%-15% of removed links, then Common Neighbors from 15% to 30% of removed links, and finally Triadic Closure Percent Close takes over as the best predictor from 30% to 50% of removed links. Level five has the worst prediction accuracy, with a 10% prediction accuracy at the top of the range. The lower prediction accuracy of level four and level five is vexing since these levels are larger-scale and incomplete, which makes prediction at these levels more useful.

Area under the curve would give a ranking for overall performance. However it loses information of which algorithm performs better at each fraction of links removed; thus, the area was not computed.

4.7.2 Binary Classification of Link Prediction Results

Binary classification places a data set in four categories: true positives, true negatives, false positives, and false negatives. A binary classification of the prediction accuracy at level three using the triadic closure close all (TCCA) algorithm is show in Figure 4-13. The top- most curve shows the recall at each fraction for the triadic closure close all algorithm, with the recall fraction (recall@k) shown as a secondary axis at the right of the figure. The initial open triad count is on the left side of the figure. This is the

count of open triads in the network before removing links and is constant at 8,960. The current open triad count starts around 8,600 and drops to 7,000. A count of the false negatives is shown. A false negative (not detecting a condition when it is present) means an open triad should have been predicted to close but was not. True positives are a count of the triads which are correctly predicted to close. False positives are a count of the open triads which are incorrectly predicted to close.

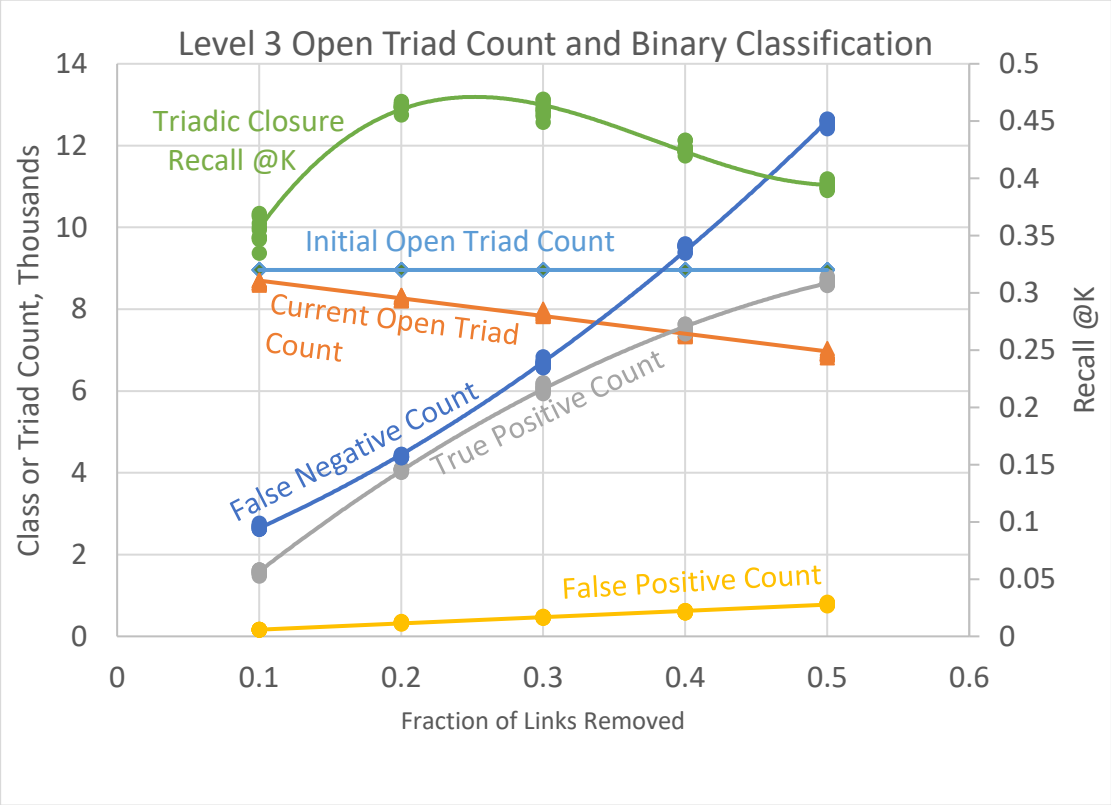


Figure 4-13: Binary classification for airplane level 3 triadic closure network predictions

Figure 4-13 provides clues as to why the performance of the TCCA algorithm improves with increased fraction of removed links, then degrades. The number of true positives (correct predictions) grows as the fraction removed is increased; however, the

number of false negatives (incorrect predictions) grows at a higher rate. The number of open triads drops as the fraction of removed links is increased, indicating there is a loss of possible correct predictions. This may be due to open triads getting destroyed when removing a link. The triadic closure recall@k curve is formed from the true positives divided by sum of true positives, false positives, and false negatives.

A binary classification of the prediction accuracy at level four using the triadic closure close all (TCCA) algorithm is shown in Figure 4-14. In this case, the true positives divided by sum of true positives, false positives, and false negatives results in a linear response of the prediction accuracy to fraction of links removed.

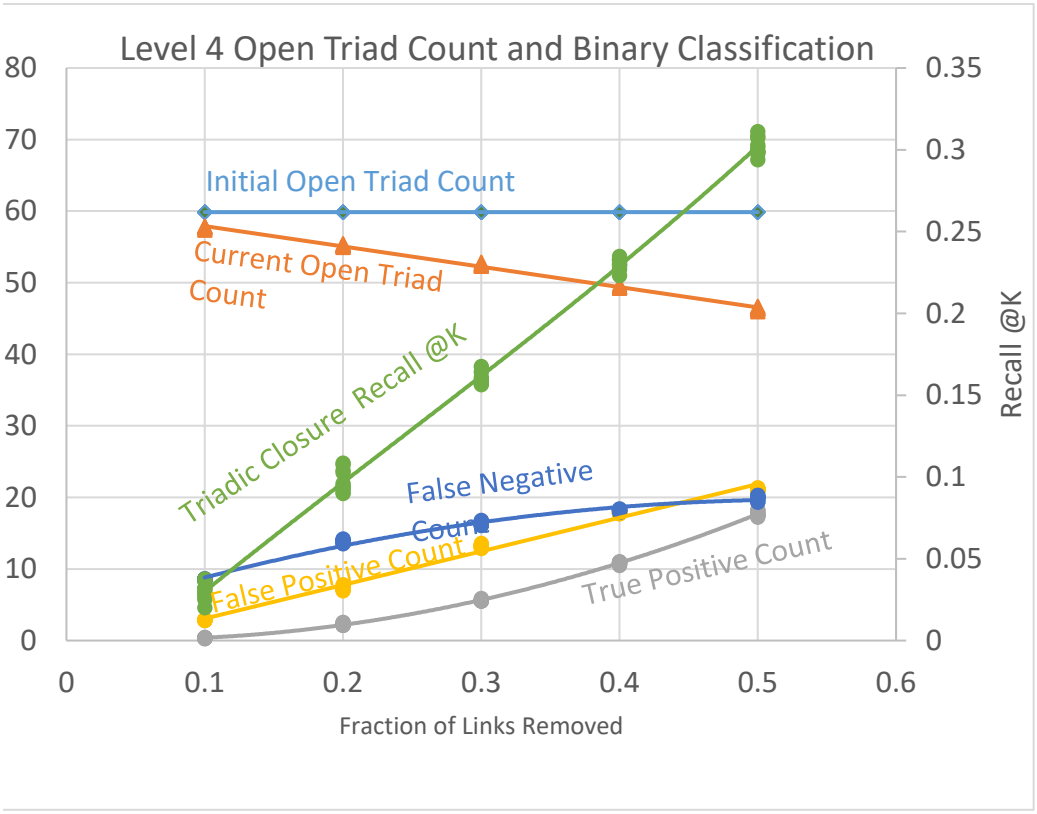


Figure 4-14: Binary classification for airplane level 4 triadic closure network predictions

A binary classification of the prediction accuracy at level five using the triadic closure close all (TCCA) algorithm is show in Figure 4-15. Once again, a linear response forms for the relationship of prediction accuracy to fraction of links removed.

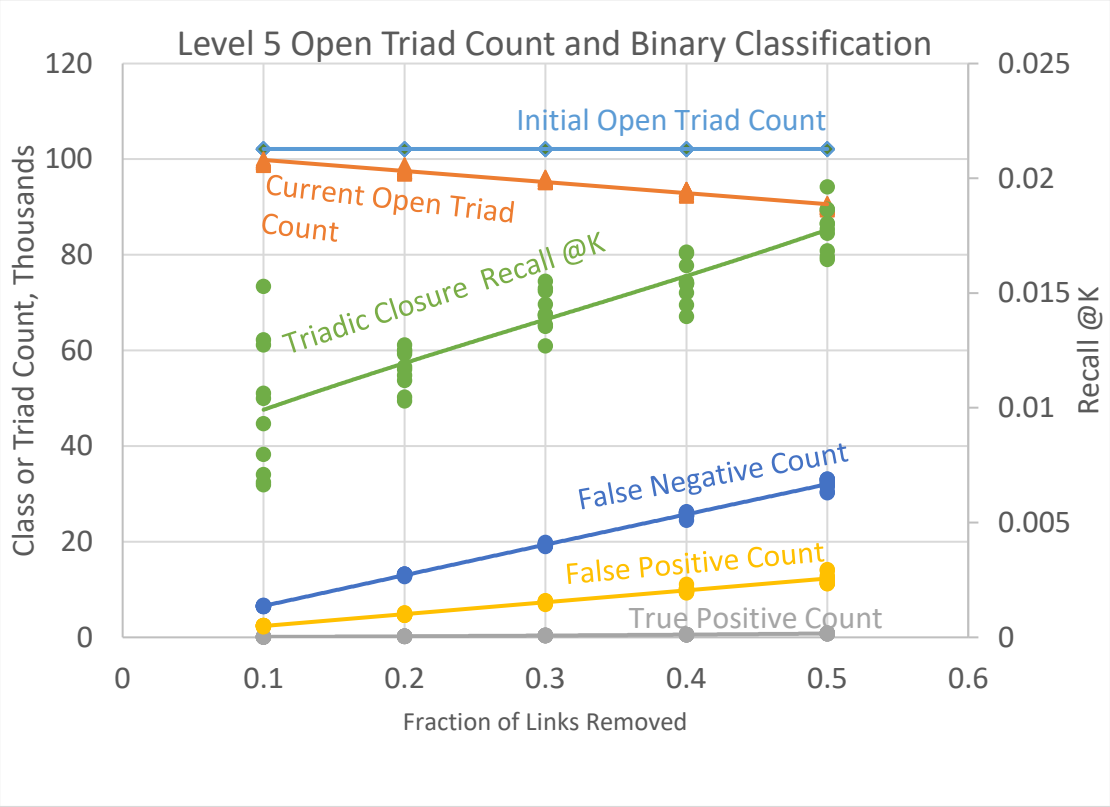


Figure 4-15: Binary classification for airplane level 5 triadic closure network predictions

Prediction accuracy at level four for larger fractions of removed edges (0.4 – 0.5) is around ranges from 20%-30%. Given the large negative class (many possible connections in the sparse network) and considering that predictions labeled false could actually be correct, this prediction approach may be useful. Predictions are labeled false if a link is predicted which is not in the set of removed links used to test the prediction. However, the predicted links could represent valid connections which had not yet been established in the source network. This prediction accuracy is also a starting point for

improvement. The triadic closure algorithms may be tuned and improved and other more advanced link prediction techniques, such as Katz and PropFlow (Lichtenwalter et al., 2010) might provide better predictions. Furthermore, as discussed in Future Work, if a supervised high-performance machine learning prediction framework were implemented, it would likely provide substantial improvement.

4.7.3 Additional Link Predictors

There were a number of link predictors evaluated, but not reported in Figure 4-12 to avoid cluttering the results graphs. The performance of these predictors was neither particularly good nor particularly bad. Predictors evaluated but not reported earlier include Preferential Attachment, Jaccard's Coefficient, Dice's Coefficient, the Structural Perturbation Method, and the CAR-based predictors, CAR-Preferential Attachment, CAR-Adamic-Adar, and CAR-Resource Allocation. Algorithm details may be found in Table 2-5, and for the Cannistraci-Alanis-Ravasi (CAR)-based predictors, see (Cannistraci et al., 2013). The results for these additional predictors are given in Figure 4-16. Results from Jaccard's Coefficient, Dice's Coefficient, the Structural Perturbation Method are shown for level 2 and 3, but not level 4 and 5, as these runs timed out after two weeks running on the high-performance computing center with no results. As the level 2 and 3 results were not promising, these predictors were not evaluated further.

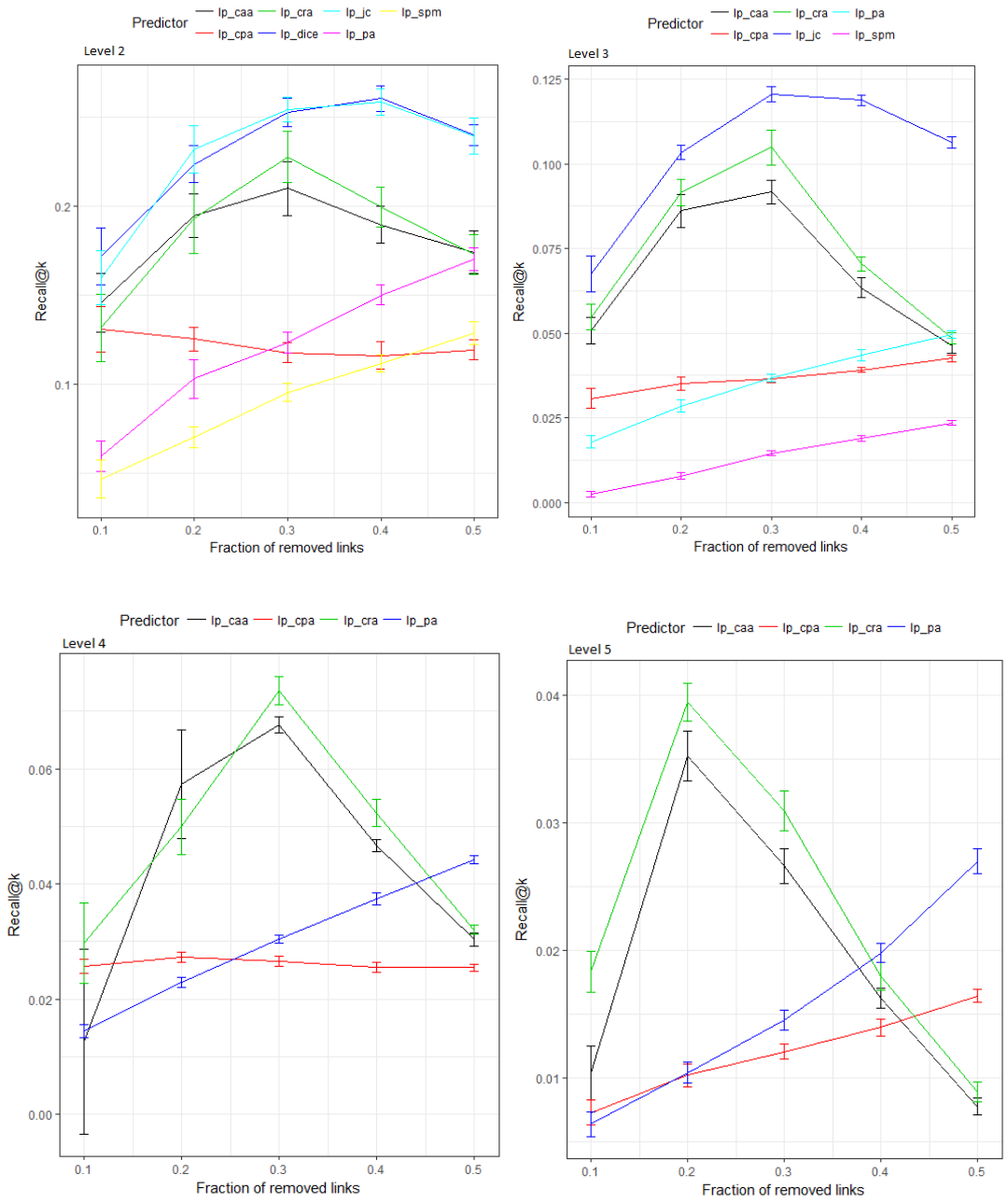


Figure 4-16: Other predictors for recall by fraction for airplane network level 2-5

4.8 Precedence Network Development Rate

The link prediction section has shown that patterns in the form of triads can be identified and used for prediction, which proves the re-use part **Hypothesis 2***: If we identify and re-use patterns, then the time required to produce well-ordered precedence*

networks will decrease. In order to test the second half of the hypothesis, time to create precedence can be reduced, a small experiment was conducted comparing selection of inputs in the conventional method vs. confirming recommended inputs provided through the triadic closure. The results are in Table 4-5. While there is not enough data for statistical tests, the results suggest that the time to create precedence networks can be reduced with link prediction. With the understanding that triads are patterns, this is the remaining evidence in resolution of **Hypothesis 2**.

Table 4-5: Time (seconds) to create precedence in a thread, old method vs. triadic closure method

Trial	Old Method	New Method	% Reduction
1	720s.	46s.	93%
2	720s.	32s.	96%

Note: Use of triadic closure method assumes system takes care of most setup in the background, including mining the motifs. Motif mining on the Level 3 network took 4.5 minutes on a HP-15-bs070wm laptop using sampling. A different computing environment would affect the mining time.

Critics might consider whether selecting inputs is a bottleneck in the rate of producing precedence network. Perhaps some other activity, such as the identification of work packages, conducting subject matter expert reviews, or defining terms could be the bottleneck? In order to address this question, the relative time invested in selecting precedence was investigated.

The software system used to develop these precedence networks archives a copy of the network for every fifteen minutes of client operation. To provide a sense of the

rate of network input selection using the current methods on the level 4 airplane network, samples in the archive were measured by counting the edges at level 4 in each file. This is presented as Figure 4-17. This figure shows that the number of edges in the level four network are growing exponentially. This substantiates the need for tools to augment the human selection of inputs.

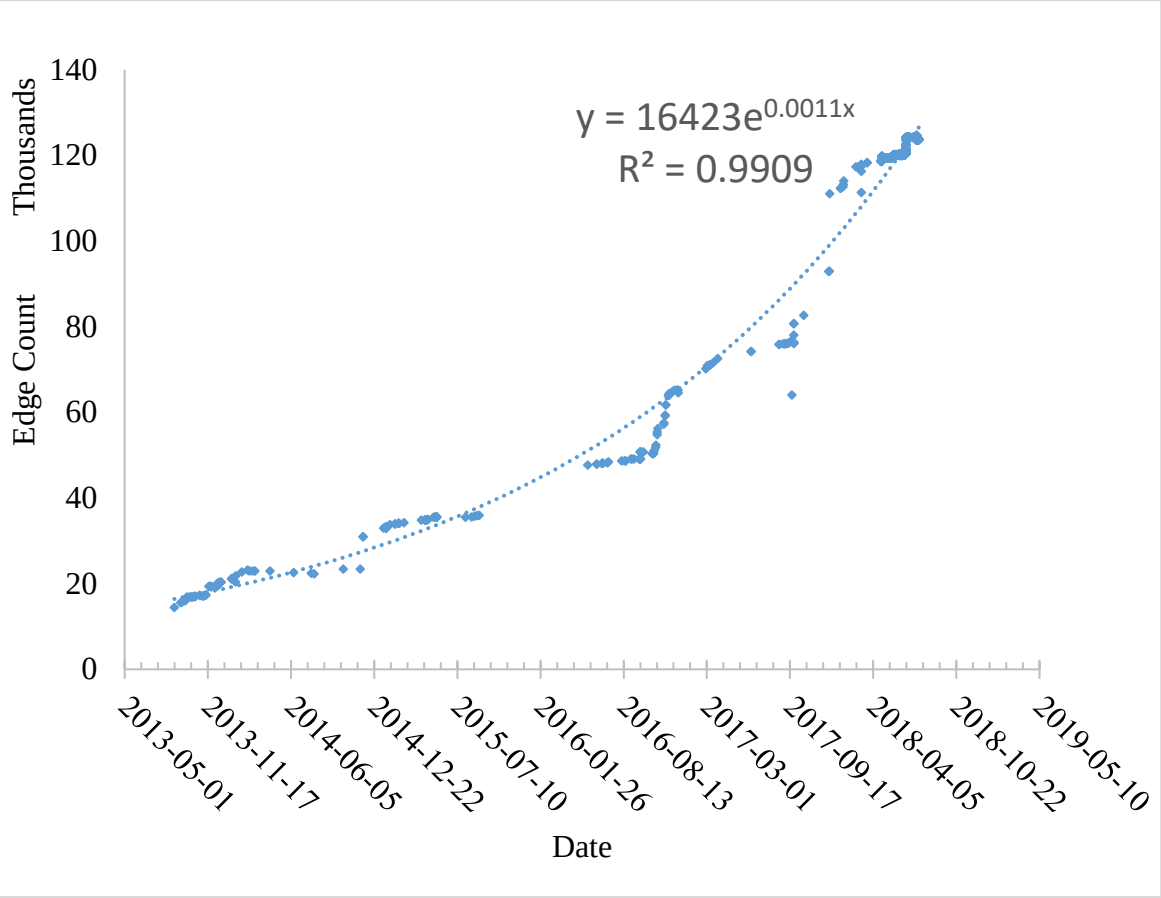


Figure 4-17: Five years of network edge edits

A Rich Site Summary (RSS) feed captures edits which are made to the network. Figure 4-18 aggregates these edits for 2018. Compared to the sum of all actions, adding edges ("5Connect Data") is the most frequent data change, and constitutes 45% of all

network modeling actions. This substantiates that as the most frequent data change, adding edges, is a bottleneck in the precedence network modeling process.

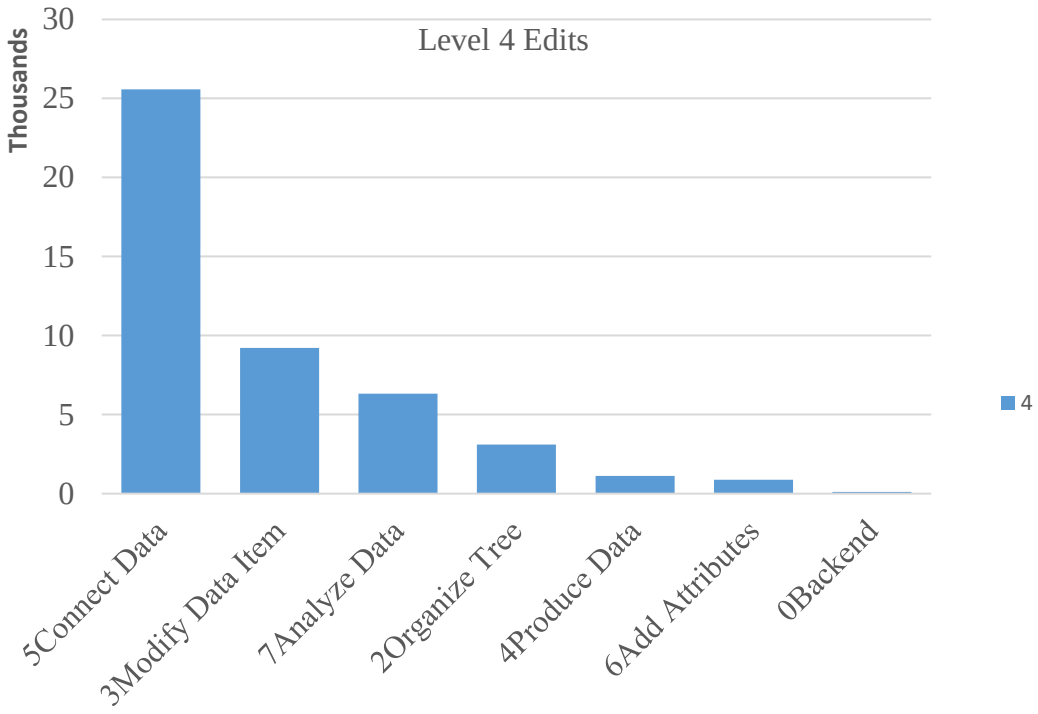


Figure 4-18: Edits to level 4 airplane network for first half of 2018

With prediction techniques developed, an evaluation framework tested which demonstrates how patterns can be reused, and evidence provided to show that input selection is a bottleneck, attention is now turned to **Hypothesis 3**, *time and cost to develop aerospace products is reduced by executing well-ordered precedence networks*.

4.9 Time and Cost

A Monte-Carlo simulation was developed to compare different execution rules for product development networks. This simulation was used to contrast the difference between executing a well-ordered precedence network as schedule driven in comparison to order driven (Grose, 2012).

In the order driven scenario, the downstream tasks are not started until the upstream task completes. In the schedule-driven scenario, downstream tasks are started on schedule. When results are available from a delayed upstream task, one rework cycle is initiated for each downstream task which has started. The rework duration is modeled as one half of the original duration.

A baseline schedule was created by running the Monte-Carlo analysis 20 times under order-driven rules. This established average start and finish times for each task. Then the finish time of one task was increased by two months to represent a delay in obtaining experimental results. The elapsed time at which the last task in the simulation completes is used to measure the time span of the simulated program. The number of resource units consumed is used to measure the cost. The results for time and cost are shown in Figure 4-19.

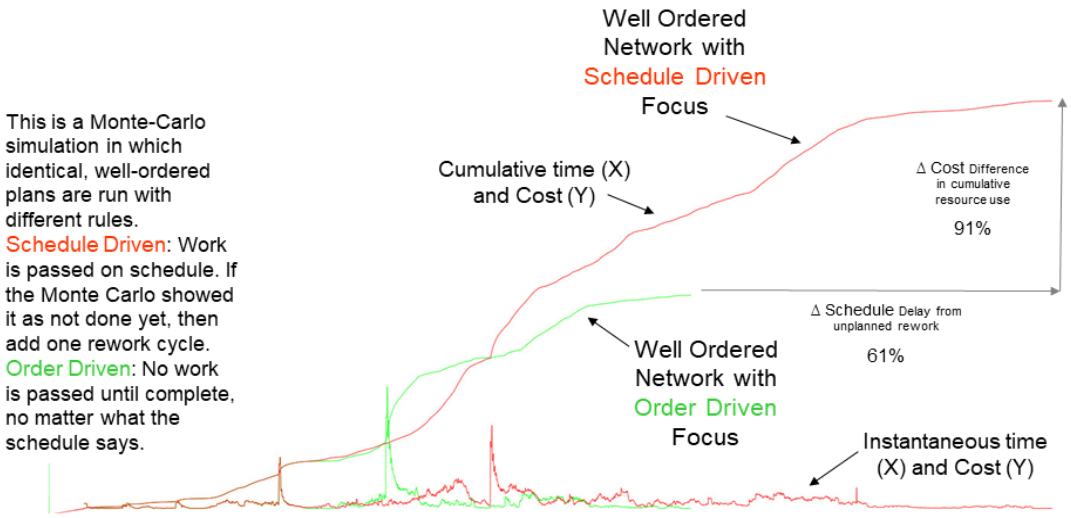


Figure 4-19: One of 40 runs of a Monte-Carlo Simulation comparing schedule-driven to order-driven execution

Figure 4-19 shows one of 40 runs. The results of a paired t-test for all 40 runs are shown Figure 4-20. This shows that order-driven execution provides significantly better performance for both time and cost, $p<<0.01$. In the case of the schedule driven duration, an outlier caused a failure of the Shapiro-Wilk normality test. The non-parametric Wilcoxon signed-rank (AKA Mann-Whitney) test was used instead to reject the null hypothesis that the means values of schedule duration are equal between the schedule-driven and order driven method. In other words, order-driven schedule execution performed better in terms of both time and cost in this simulation.

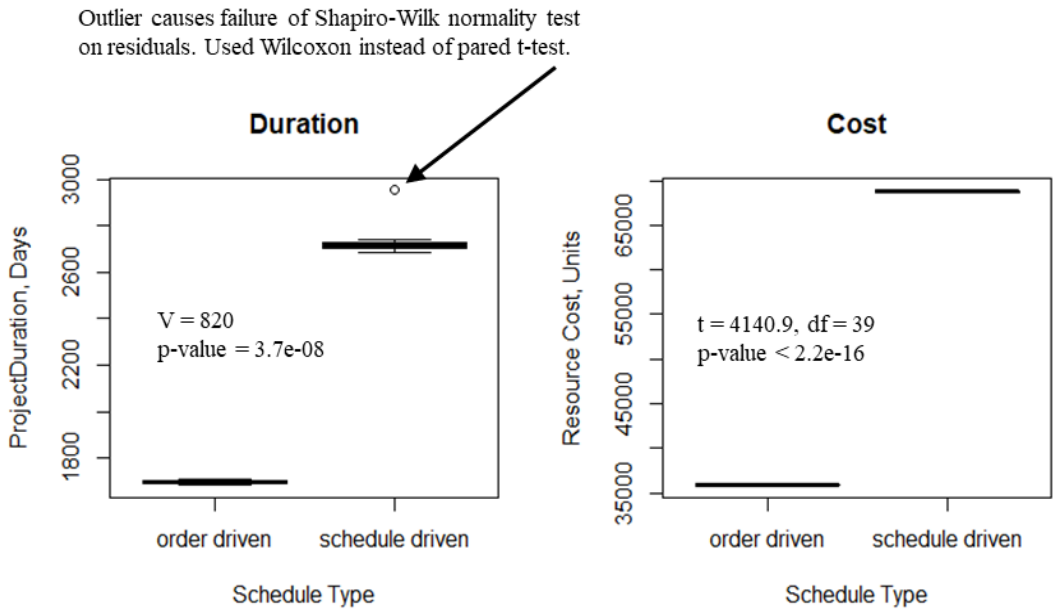


Figure 4-20: Results of 40 runs for each of a Monte-Carlo simulation comparing schedule-driven to order-driven execution.

Critics might suggest that a two-month delay in obtaining experimental results is unreasonable or extreme. Results from an actual program showing the payoff for using a well-ordered plan is shown as Figure 4-21. There was a four-month delay in establishing the configuration definition, which depends on experimental results mentioned earlier. Standard business policy would have been to delay to the expected delivery of the remaining work by four months. Rather than delaying the expected delivery, management was convinced to proceed using the well-ordered predictive model. After executing using the order-driven rules, actual delivery of this phase of the program was three days early.

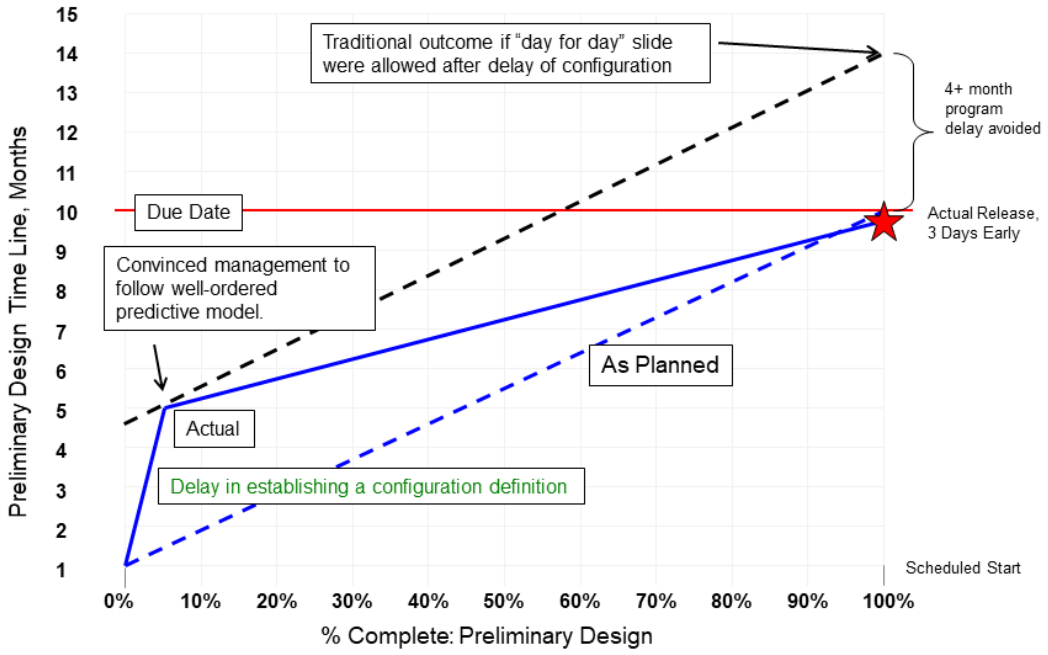


Figure 4-21 Well-Ordering Payoff on Preliminary Design

4.10 Summary

Several methods were evaluated to detect patterns and predict inputs for an airplane product development network. These methods included frequent subgraph mining, using the implementation GraMi. This resulted in finding large patterns, however the meaning and use of these patterns were not ascertained.

Motif detection using FANMOD combined with hypernym/hyponym categorization of node labels was applied. This work established that there are patterns repeated across communities, and also established that the motif detection results can be used for triadic closure. To overcome the limitation of sixteen categories (colors) in FANMOD, a Java algorithm was developed and implemented to conduct a simple triadic census and predict inputs via triadic closure. This provides a method to predict inputs in an example well-ordered product development precedence network.

A Monte-Carlo simulation was used to compare schedule driven execution with order driven execution. The results show that starting with a well-ordered precedence network, order driven execution can significantly reduce cost and schedule for product development.

Data from five years of the development of the airplane product development network, and a half-year of network edits shows that adding edges is the most frequent action taken on the network, and constituting up to 45% of all network edit actions.

Chapter 5 Discussion/Conclusion

5.1 Summary of Contribution

This praxis has applied motifs and motif significance profiles developed for bioinformatics to project management precedence networks. To our knowledge, the closest similar work is on graphs representing system architecture (Shaja & Sudhakar, 2013).

This praxis has also applied link prediction techniques developed for social network analysis to a recommendation system which augments human selection of inputs during precedence network development. To our knowledge, link prediction techniques from social network analysis have not been applied to project management precedence networks.

We propose an input recommendation system can be developed, which uses the subgraph mining and link prediction techniques discussed earlier. The high-level view of the proposed input recommendation system is shown in Figure 5-1.

.

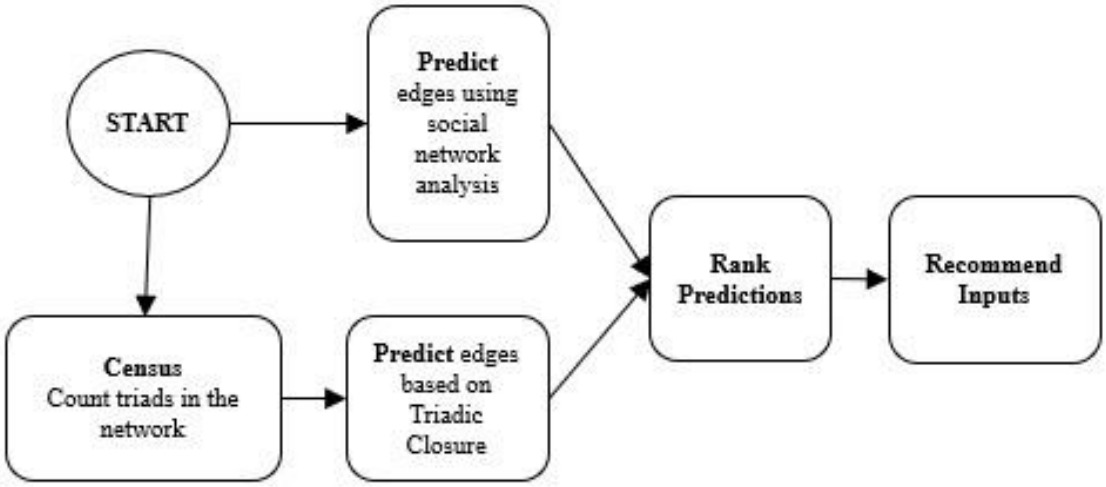


Figure 5-1: Input Recommendation System

More details on the conceptual design of the input recommendation system is provided as Appendix E and Appendix F. This conceptual design uses Unified Modeling Language (UML) to "*specify, visualize, construct, and document the artifacts of a software system*" (Rumbaugh, Booch, & Jacobson, 2017).

5.2 Results Discussion

The research questions and hypothesis are summarized in Table 5-1.

Table 5-1: Research Questions and Hypotheses

ID	Description
Q1	Are there patterns of tasks and inputs in an example well-ordered aerospace precedence network?
Q2	What techniques can be used, (machine learning, neural networks, data mining, etc.) to discover patterns in networks?
Q3	What are appropriate categories for precedence network patterns?
Q4	Does the use of patterns lead to cost and schedule reductions?

H1	There are patterns of tasks and inputs in well-ordered Product development precedence networks.
H2	If we identify and re-use patterns, then the time required to produce well-ordered precedence networks will decrease.
H3	Time and cost to develop aerospace products is reduced by executing well-ordered precedence networks.

A summary of the research hypothesis table is included as Table 5-2.

Table 5-2: Research Hypothesis Table

ID	Response Variable (Y)	Predictor Variables (X)	Analysis Method	Results
H1	Number, size, and label variety of patterns identified at each network level.	Frequency, τ;	Graph Theory; t-test against bootstrapped random networks.	Found isomorphs and colored motifs
H2	Number and size of re-use instances.	Network Partition, Pattern Classification	T-tests within and between network partitions	Found significant patterns repeated across communities.
H2	Network Production Rate	No. and size of patterns used.	Examples/logic	experimental results.
H3	Project Duration and Cost in Resource Units	Order Driven vs. Schedule Driven Execution Rules	Monte-Carlo Analysis; paired t-tests, Wilcoxon signed-rank test.	Order driven significantly better.

Nodes of the network were classified using a hypernym-hyponym table. The most frequent hyponyms were used to classify the nodes using sixteen colors. The colored network was then mined for motifs. Colored motifs have been discovered which characterize patterns of nodes and inputs that are statistically over-represented in this network as compared to a random graph. The results show that there are interesting patterns in the airplane product development precedence network studied in this praxis. For any open triad, the probability that the triad will be closed, and in which direction may be predicted. This probability can be used to provide recommended inputs to the user of the system which develops precedence networks. An algorithm to overcome the limitation of sixteen categories (colors) using simple triadic census has also been shown.

5.3 Potential Future Work

Promising topics have been raised by this research, allowing for further studies. Social network analysis using the map equation algorithm can decompose the network into communities. These community structures could provide insight into how to better align organizations to maximize information flow and collaboration.

The degree distribution plots confirm incomplete modeling at level four and five. These levels have a relative abundance of nodes with degree one. These nodes appear as outliers in the degree distribution plots. For level three, this condition was found to be related to incomplete modeling, and could be used as a guide in the future to assess network modeling maturity.

The toy network constructed to have a higher probability for connections between nodes which are physically close (Artzy-Randrup et al., 2004) has intriguing implications for modeling product development. Systems and components which interface could be

analyzed by computing a distance function. This information could be used to further inform an input recommendation system, and to catch errors related to connecting unrelated and physically distant components. While the distance function for physical structures is obvious, more thought would need to go into a distance function for system-system and system-structure relations.

Traditional algorithms for link prediction as an unsupervised machine learning were examined even though, however a high-performance link prediction (HPLP) framework has been published which provides a 30% improvement over unsupervised methods for the evaluated networks. The HPLP approach is a supervised machine learning framework which uses local network measurements and node similarity measurements as features, along with bagged random forests. (Lichtenwalter et al., 2010). Running this framework was straightforward against the provided sample networks. Attempts to analyze using the airplane product development network resulted in errors which are as of this date unresolved.

Link prediction using supervised, longitudinal, and other methods using more information than just the node term identifier should yield substantial improvements in link prediction accuracy, at the cost of increased code complexity and longer run times.

Bibliography

Abu-Aisheh, Z., Gaüzère, B., Bougleux, S., Ramel, J., Brun, L., Raveaux, R., . . . Adam, S. (2017). Graph edit distance contest: Results and future challenges. *Pattern Recognition Letters, 100*, 96-103.

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *Acm Sigmod Record, 22*(2), 207-216.

Ahmed, N. K. (2015). *Scaling up network analysis and mining: Statistical sampling, estimation, and pattern discovery* Available from Dissertations & Theses Europe Full Text: Science & Technology. Retrieved from https://search.proquest.com/docview/1747392403

Akhtar, N. (2014). (2014). Social network analysis tools. Paper presented at the *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference On*, 388-392.

Alanis-Lobato, G. (2017). LinkPrediction: Implementation of different classes of link predictors and methods to assess and visualise their performance. Retrieved from https://github.com/galanisl/LinkPrediction

Alon, U. (2007). Network motifs: Theory and experimental approaches. *Nature Reviews Genetics, 8*(6), 450.

Arora, A., Sachan, M., & Bhattacharya, A. (2014). (2014). Mining statistically significant connected subgraphs in vertex labeled graphs. Paper presented at the *Sigmod*, 1003-1014. doi:10.1145/2588555.2588574 Retrieved from http://dl.acm.org.proxygw.wrlc.org/citation.cfm?id=2588574

Artzy-Randrup, Y., Fleishman, S. J., Ben-Tal, N., & Stone, L. (2004). Comment on 'network motifs: Simple building blocks of complex networks' and 'superfamilies of evolved and designed networks'. *Science, 305*(5687), 1107.

Babai, L. (2016). (2016). Graph isomorphism in quasipolynomial time [extended abstract]. Paper presented at the *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, 684–697. doi:10.1145/2897518.2897542 Retrieved from http://doi.acm.org.proxygw.wrlc.org/10.1145/2897518.2897542

Batagelj, V., & Mrvar, A. (February 12-16, 2003). (February 12-16, 2003). How to analyze large networks with pajek. Paper presented at the *SUNBELT XXIII, International Social Network Conference,* Westin Regina Resort Hotel, Cancun, Quintana Roo, Mexico.

Becker, J., Weiß, B., & Winkelmann, A. (2010). *Utility vs. efforts of business process modeling an exploratory survey in the financial sector*

Bell, C. E. (1989). Maintaining project networks in automated artificial intelligence planning. *Management Science, 35*(10), 1192-1214. doi:10.1287/mnsc.35.10.1192

Bonner, J. M., Ruekert, R. W., & Walker, O. C. (2002). Upper management control of new product development projects and project performance. *The Journal of Product Innovation Management, 19*(3), 233-245. doi:10.1016/S0737-6782(02)00139-X

Borgelt, C. (2012). Frequent item set mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2*(6), 437-456. doi:10.1002/widm.1074

Brown, A. (2018). Research guides: What graduate students need to know: Citation chasing & cited reference searching. Retrieved from https://libguides.gwu.edu/c.php?g=259097&p=1729224

Bulychev, P., & Minea, M. (2009). (2009). An evaluation of duplicate code detection using anti-unification. Paper presented at the *Proc. 3rd International Workshop on Software Clones,* Proc. 3rd International Workshop on Software Clones.

Button, S. D., Brown, J. W., Kim, T. D., & Sherer, T. E. (2018). Monte-carlo simulation of large-scale complex life-cycle product patterns for program management. *Boeing Technical Journal, 8*(2) Retrieved from https://www.boeing.com/features/innovation-quarterly/aug2018/btj-simulations.page

Cannistraci, C. V., Alanis-Lobato, G., & Ravasi, T. (2013). From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks. *Scientific Reports, 3*, 1613.

Conte, D., Foggia, P., Sansone, C., & Vento, M. (2004). Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence, 18*(03), 265-298.

Cook, D. J., Greengold, N. L., Ellrodt, A. G., & Weingarten, S. R. (1997). The relation between systematic reviews and practice guidelines. *Annals of Internal Medicine, 127*(3), 210-216.

Cooper, R. G. (2017). Idea-to-launch gating systems. *Research Technology Management, 60*(1), 48-52. Retrieved from http://www.econis.eu/PPNSET?PPN=887358284

Crosby, P. (2012). Key success drivers: Meta-study findings applicable to large high-technology projects. *International Journal of Information Technology Project Management (IJITPM), 3*(2), 1-20. doi:10.4018/jitpm.2012040101

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal, Complex Systems, 1695*(5), 1-9.

Dalbey, J. (2003). Pseudocode standard. Retrieved from http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html

Darwiche, A., Levitt, R. E., & Hayes-Roth, B. (1988). OARPLAN: Generating project plans by reasoning about objects, actions and resources. *Ai Edam, 2*(3), 169-181. doi:10.1017/S0890060400000639

Davis, D. A., & Chawla, N. V. (2011). Exploring and exploiting disease interactions from multi-relational gene and phenotype networks. *PloS One, 6*(7), e22670.

Davis, J., & Goadrich, M. (2006). (2006). The relationship between precision-recall and ROC curves. Paper presented at the *Proceedings of the 23rd International Conference on Machine Learning*, Proceedings of the 23rd international conference on Machine learning. 233-240.

Demesmaeker, F., Ghrab, A., Nijssen, S., & Skhiri, S. (2017). (2017). Discovering interesting patterns in large graph cubes. Paper presented at the *2017 IEEE International Conference on Big Data (Big Data)*, 3322-3331. doi:10.1109/BigData.2017.8258317

Dijkman, R. R., Dongen, v., BF Boudewijn, Dumas, M., García-Bañuelos, L., Kunze, M., Leopold, H., . . . Yan, Z. (2013). A short survey on process model similarity. *Seminal contributions to information systems engineering : 25 years of CAiSE* (pp. 421) Springer. Retrieved from http://www.narcis.nl/publication/RecordID/oai:library.tue.nl:762181

Edler, D., & Rosvall, M. (2014). The MapEquation software package. Retrieved from http://www.mapequation.org

Ekanayake, C. C., Dumas, M., Garcia-Banuelos, L., & La Rosa, M. (2013). Slice, mine and dice : Complexity-aware automated discovery of business process models. *Business process management* (pp. 49-64). Berlin, Heidelberg: Springer. Retrieved from https://eprints.qut.edu.au/58949

Elseidy, M., Kalnis, P., Abdelhamid, E., & Skiadopoulos, S. (2014). GRAMI: Frequent subgraphpattern mining in a single large graph. *Proceedings of the VLDB Endowment, 7*(7), 517-528. doi:10.14778/2732286.2732289

Emblemsvåg, J. (2014). Lean project planning in shipbuilding. *Journal of Ship Production and Design, 30*(2), 79-88. doi:10.5957/JSPD.30.2.130054

Faghihi, V., Nejat, A., Reinschmidt, K. F., & Kang, J. H. (2015). Automation in construction scheduling: A review of the literature. *The International Journal of Advanced Manufacturing Technology, 81*(9), 1845-1856. doi:10.1007/s00170-015-7339-0

Fankhauser, S., Riesen, K., & Bunke, H. (2011). (2011). Speeding up graph edit distance computation through fast bipartite matching. Paper presented at the *International Workshop on Graph-Based Representations in Pattern Recognition,* 102-111.

Fischer, A., Suen, C. Y., Frinken, V., Riesen, K., & Bunke, H. (2015). Approximation of graph edit distance based on hausdorff matching. *Pattern Recognition, 48*(2), 331-343.

Fischer, I., & Meinl, T. (2004). (2004). Graph based molecular data mining - an overview. Paper presented at the *IEEE International Conference on Systems, Man and Cybernetics, , 5* 4582 vol.5. doi:10.1109/ICSMC.2004.1401253 Retrieved from https://ieeexplore.ieee.org/document/1401253

Foggia, P., Percannella, G., & Vento, M. (2014). Graph matching and learning in pattern recognition in the last 10 years. *International Journal of Pattern Recognition and Artificial Intelligence, 28*(01), 1450001.

Gao, F., Musial, K., Cooper, C., & Tsoka, S. (2015). Link prediction methods and their accuracy for different social networks and network metrics. *Scientific Programming, 2015*, 1.

Gao, X., Xiao, B., Tao, D., & Li, X. (2010). A survey of graph edit distance. *Pattern Analysis and Applications, 13*(1), 113-129.

Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences, 99*(12), 7821-7826.

Golpayegani, S. A. H., & Parvaresh, F. (2011). The logical precedence network planning of projects, considering the finish-to-start (FS) relations, using neural networks. *The International Journal of Advanced Manufacturing Technology, 55*(9), 1123-1133. doi:10.1007/s00170-010-3125-1

Grose, D. L. (1994). (1994). Reengineering the aircraft design process. Paper presented at the *5th Symposium on Multidisciplinary Analysis and Optimization,* Panama City Beach,FL,U.S.A.

Grose, D. L. (2012). (2012). An approach to managing risk in non-recurring product development programs. Paper presented at the *Boeing Technical Excellence 19,* Los Angeles, CA.

Grose, D. L., Sherer, T. E., & Button, S. D. (2014). In The Boeing Company (Ed.), *Total-ordering in process planning*. United States:

Grose, D. L., Sherer, T. E., & Button, S. D. (2016). In The Boeing Company (Ed.), *Methods and systems for correspondence pattern automation of scalable feed forward processes*. United States:

Grycner, A., Weikum, G., Pujara, J., Foulds, J., & Getoor, L. (2015). (2015). Relly: Inferring hypernym relationships between relational phrases. Paper presented at the *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* 971-981.

Han, J., Kamber, M., & Pei, J. (2000). *Data mining: Concepts and techniques (the morgan kaufmann series in data management systems)* Morgan Kaufmann.

Hertling, S., & Paulheim, H. (2017). (2017). WebIsALOD: Providing hypernymy relations extracted from the web as linked open data. Paper presented at the *16th International Semantic Web Conference,* Vienna, Austria. Retrieved from http://webisa.webdatacommons.org/

Ibragimov, R., Malek, M., Guo, J., & Baumbach, J. (2013). (2013). Gedevo: An evolutionary graph edit distance algorithm for biological network alignment. Paper presented at the *OASIcs-OpenAccess Series in Informatics, , 34*

Jalili, Y., & Ford, D. N. (2016). Quantifying the impacts of rework, schedule pressure, and ripple effect loops on project schedule performance. *System Dynamics Review, 32*(1), 82-96. doi:10.1002/sdr.1551

Kashtan, N., Itzkovitz, S., Milo, R., & Alon, U. (2004). Topological generalizations of network motifs. *Physical Review E, 70*(3), 031909.

King, W. R., & Lukas, P. A. (1973). An experimental analysis of network planning. *Management Science, 19*(12), 1423-1432. doi:10.1287/mnsc.19.12.1423

Klarreich, E. (2014). Landmark algorithm breaks 30-year impasse. Retrieved from https://www.quantamagazine.org/algorithm-solves-graph-isomorphism-in-record-time-20151214/

Konagurthu, A. S., & Lesk, A. M. (2008). On the origin of distribution patterns of motifs in biological networks. *BMC Systems Biology, 2*(1), 73.

Koschmider, A., Fellmann, M., Schoknecht, A., & Oberweis, A. (2014). Analysis of process model reuse: Where are we now, where should we go from here? *Decision Support Systems, 66*, 9-19. doi:10.1016/j.dss.2014.05.012

Kotsiantis, S., & Kanellopoulos, D. (2006). Association rules mining: A recent overview. *GESTS International Transactions on Computer Science and Engineering, 32*(1), 71-82.

La Rosa, M., Dumas, M., Ekanayake, C. C., García-Bañuelos, L., Recker, J., & Hofstede, t., AHM Arthur. (2015). Detecting approximate clones in business process model repositories. *Information Systems, 49*(April), 102-125. doi:10.1016/j.is.2014.11.010

La Rosa, M., Reijers, H. A., van der Aalst, Wil M.P, Dijkman, R. M., Mendling, J., Dumas, M., & García-Bañuelos, L. (2011). APROMORE: An advanced process model repository. *Expert Systems with Applications, 38*(6), 7029-7040. doi:10.1016/j.eswa.2010.12.012

Lerouge, J., Abu-Aisheh, Z., Raveaux, R., Heroux, P., & Adam, S. (2015). Graph edit distance: A new binary linear programming formulation. *arXiv Preprint arXiv:1505.05740,*

Li, R., & Wang, W. (2015). (2015). REAFUM: Representative approximate frequent subgraph mining. Paper presented at the *SIAM International Conference on Data Mining,* SIAM International Conference on Data Mining.

Liben-Nowell, D., & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology, 58*(7), 1019-1031.

Lichtenwalter, R. N., & Chawla, N. V. (2011). Lpmade: Link prediction made easy. *Journal of Machine Learning Research, 12*(Aug), 2489-2492.

Lichtenwalter, R. N., Lussier, J. T., & Chawla, N. V. (2010). (2010). New perspectives and methods in link prediction. Paper presented at the *Proceedings of the 16th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, 243-252.

Mackin, M. (2009). *Best practices: High levels of knowledge at key points differentiate commercial shipbuilding from navy shipbuilding.* (). Washington, D.C.: U.S. Government Accountability Office. Retrieved from Social Science Premium Collection Retrieved from https://search.proquest.com/docview/1820807425

Martínez, V., Berzal, F., & Cubero, J. (2017). A survey of link prediction in complex networks. *ACM Computing Surveys (CSUR), 49*(4), 69.

Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., & Alon, U. (2004). Response to comment on 'network motifs: Simple building blocks of complex networks' and 'superfamilies of evolved and designed networks'. *Science, 305*(5687), 1107.

Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., . . . Alon, U. (2004). Superfamilies of evolved and designed networks. *Science*, *303*(5663), 1538-1542.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., & Alon, U. (2002). Network motifs: Simple building blocks of complex networks. *Science, 298*(5594), 824-827.

Nepusz, T. (2012). What are the differences between community detection algorithms in igraph? Retrieved from https://stackoverflow.com/a/9478989/4230743

Neuhaus, M., & Bunke, H. (2007). *Bridging the gap between graph edit distance and kernel machines* World Scientific.

Plotkin, G. D. (1970). A note on inductive generalization. *Machine Intelligence, 5*(1), 153-163.

Powner, D. A. (2008). *OMB and agencies need to improve planning, management, and oversight of projects totaling billions of dollars.* (No. GAO-08-1051 T). Washington, D.C.: U.S. Govt. Accountability Office. Retrieved from http://purl.access.gpo.gov/GPO/LPS100001

Powner, D. A. (2004). *Information technology investment management: A framework for assessing and improving process maturity.* (No. 2004). Washington, D.C.: United States Government Accountability Office.

Raecker, J. S. (2014). Network analysis of complex commercial airplane product development (APD) to identify priorities for management focus. *Boeing Technical Journal, 4*, 1-11.

Rapoport, A. (1953). Spread of information through a population with socio-structural bias: I. assumption of transitivity. *The Bulletin of Mathematical Biophysics, 15*(4), 523-533.

Rasche, F., & Wernicke, S. (2006). *FANMOD*

fast network motif detection manual

Rashid, M. M., Karim, M. R., Jeong, B., & Choi, H. (2012). Efficient mining of interesting patterns in large biological sequences. *Genomics & Informatics, 10*(1), 44-50. doi:10.5808/GI.2012.10.1.44

Riesen, K., Emmenegger, S., & Bunke, H. (2013). (2013). A novel software toolkit for graph edit distance computation. Paper presented at the *International Workshop on Graph-Based Representations in Pattern Recognition*, 142-151.

Rosvall, M., Axelsson, D., & Bergstrom, C. T. (2009). The map equation. *The European Physical Journal Special Topics, 178*(1), 13-23.

Rumbaugh, J., Booch, G., & Jacobson, I. (2017). *The unified modeling language reference manual* Addison Wesley.

Rupani, S. (2011). *Standardization of product development processes in multi-project organizations* Retrieved from https://search.proquest.com/docview/1636542221

Saigo, H., & Tsuda, K. (2008). (2008). Iterative subgraph mining for principal component analysis. Paper presented at the *Eighth IEEE International Conference on Data Mining*, 1007-1012. doi:10.1109/ICDM.2008.62 Retrieved from http://ieeexplore.ieee.org/document/4781216

Saltz, M., Jain, A., Kothari, A., Fard, A., Miller, J. A., & Ramaswamy, L. (2014). (2014). DualIso: An algorithm for subgraph pattern matching on very large labeled graphs. Paper presented at the *2014 IEEE International Conference on Big Data*, 498-505.

doi:10.1109/BigData.Congress.2014.79 Retrieved from http://ieeexplore.ieee.org/document/6906821

Schoknecht, A., Thaler, T., Fettke, P., Oberweis, A., & Laue, R. (2017). Similarity of business process models-A state-of-the-art analysis. *ACM Computing Surveys (CSUR), 50*(4), 1-33. doi:10.1145/3092694

Schumm, D., Karastoyanova, D., Kopp, O., Leymann, F., Sonntag, M., & Strauch, S. (2011). Process fragment libraries for easier and faster development of process-based applications. *Journal of Systems Integration, 2*(1), 39-55.

Shaja, A. S., & Sudhakar, K. (2013). Why do motifs occur in engineering systems? *ICoRD'13* (pp. 15-27) Springer.

Shasha, D., Tsong-Li Wang, J., & Zhang, S.Unordered tree mining with applications to phylogeny. Paper presented at the *Proceedings. 20th International Conference on Data Engineering*, Proceedings. 20th International Conference on Data Engineering.

Skouradaki, M., Andrikopoulos, V., Kopp, O., & Leymann, F. (2016). (2016). RoSE: Reoccurring structures detection
in BPMN 2.0 process model collections. Paper presented at the *OTM Confederated International Conferences ''On the Move to Meaningful Internet Systems",* 263-281. doi:10.1007/978-3-319-48472-3_15

Staden, R. (1989). Methods for calculating the probabilities of finding patterns in sequences. *Bioinformatics, 5*(2), 89-96.

Sullivan, M. J. (2017). *Defense acquisitions, assessments of selected weapon programs; no. GAO-17-333SP*. (). Washington, D.C.: United States Government Accountability Office. Retrieved from http://purl.fdlp.gov/GPO/gpo79184

Sullivan, M. J. (2010). *Best practices: DOD can achieve better outcomes by standardizing the way manufacturing risks are managed.* (No. 2010). Washington, D.C.: United States Government Accountability Office.

The George Washington University. (2013). About colonial one. Retrieved from https://colonialone.gwu.edu/about/about/

Tselenti, P. (2017). *Improving accuracy of recommender systems through triadic closure*

Ullmann, J. (1976). An algorithm for subgraph isomorphism. *Journal of the ACM (JACM), 23*(1), 31-42. doi:10.1145/321921.321925

Ullmann, J. (2010). Bit-vector algorithms for binary constraint satisfaction and subgraph isomorphism. *Journal of Experimental Algorithmics (JEA), 15*(15), 1.64. doi:10.1145/1671970.1921702

Valverde, S., & Solé, R. V. (2005). Network motifs in computational graphs: A case study in software architecture. *Physical Review E, 72*(2), 026107.

Vespignani, A. (2018). *Twenty years of network science* Nature Publishing Group. doi:10.1038/d41586-018-05444-y

Wang, P., Xu, B., Wu, Y., & Zhou, X. (2015). Link prediction in social networks: The state-of-the-art. *Science China Information Sciences, 58*(1), 1-38.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *Nature, 393*(6684), 440.

Wernicke, S., & Rasche, F. (2006). FANMOD: A tool for fast network motif detection. *Bioinformatics, 22*(9), 1152-1153.

Winston, P. H. (1975). *Learning structural descriptions from examples* McGraw-Hill.

Wong, E., Baur, B., Quader, S., & Huang, C. (2011). Biological network motif detection: Principles and practice. *Briefings in Bioinformatics, 13*(2), 202-215.

Ying Li, Bin Cao, Lida Xu, Jianwei Yin, Shuiguang Deng, Yuyu Yin, & Zhaohui Wu. (2014). An efficient recommendation method for improving business process modeling. *IEEE Transactions on Industrial Informatics, 10*(1), 502-513. doi:10.1109/TII.2013.2258677

Yusof, Y., & Latif, K. (2014). Survey on computer-aided process planning. *The International Journal of Advanced Manufacturing Technology, 75*(1), 77-89. doi:10.1007/s00170-014-6073-3

Zaaboub Haddar, N., Makni, L., & Ben Abdallah, H. (2014). Literature review of reuse in business process modeling. *Software & Systems Modeling, 13*(3), 975-989. doi:10.1007/s10270-012-0286-4

Zahn, C. T. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers, 100*(1), 68-86.

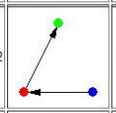
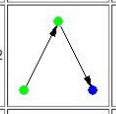
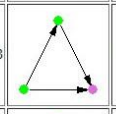
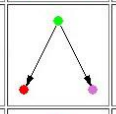
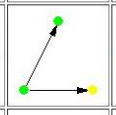
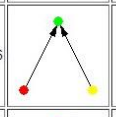
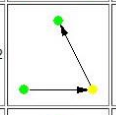
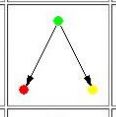
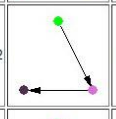
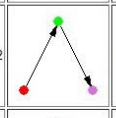
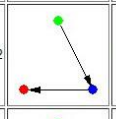
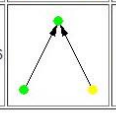
Zhang, Z., Li, J., Zhao, H., & Tang, B. (2018). SJTU-NLP at SemEval-2018 task 9: Neural hypernym discovery with term embeddings. *arXiv Preprint arXiv:1805.10465,* Retrieved from http://arxiv.org/abs/1805.10465

Appendix A Toy Example Remaining Mining Results

Previous Index Next

Size-3 Network Motifs in 'EdgeListNoHypToy.txt' Page 1

Generated with FANMOD - FAst Network MOtif Detection. Written by Sebastian Wernicke and Florian Rasche

ID	Adj-Matrix	Frequency [Original]	Mean-Freq [Random]	Standard-Dev [Random]	Z-Score	p-Value
12		4.7619%	4.7619%	1.0066e-015	-0.9995	0
12		4.7619%	4.7619%	1.0066e-015	-0.9995	0
38		4.7619%	4.7619%	1.0066e-015	-0.9995	0
6		4.7619%	4.7619%	1.0066e-015	-0.9995	0
6		4.7619%	4.7619%	1.0066e-015	-0.9995	0
36		4.7619%	4.7619%	1.0066e-015	-0.9995	0
12		4.7619%	4.7619%	1.0066e-015	-0.9995	0
6		4.7619%	4.7619%	1.0066e-015	-0.9995	0
12		4.7619%	4.7619%	1.0066e-015	-0.9995	0
12		4.7619%	4.7619%	1.0066e-015	-0.9995	0
12		4.7619%	4.7619%	1.0066e-015	-0.9995	0
36		4.7619%	4.7619%	1.0066e-015	-0.9995	0

Previous Index Next

Generated with FANMOD - FAst Network MOtif Detection. Written by Sebastian Wernicke and Florian Rasche

Figure A-1: Remaining results of mining toy example.

Appendix B Size 17 Frequent Subgraphs

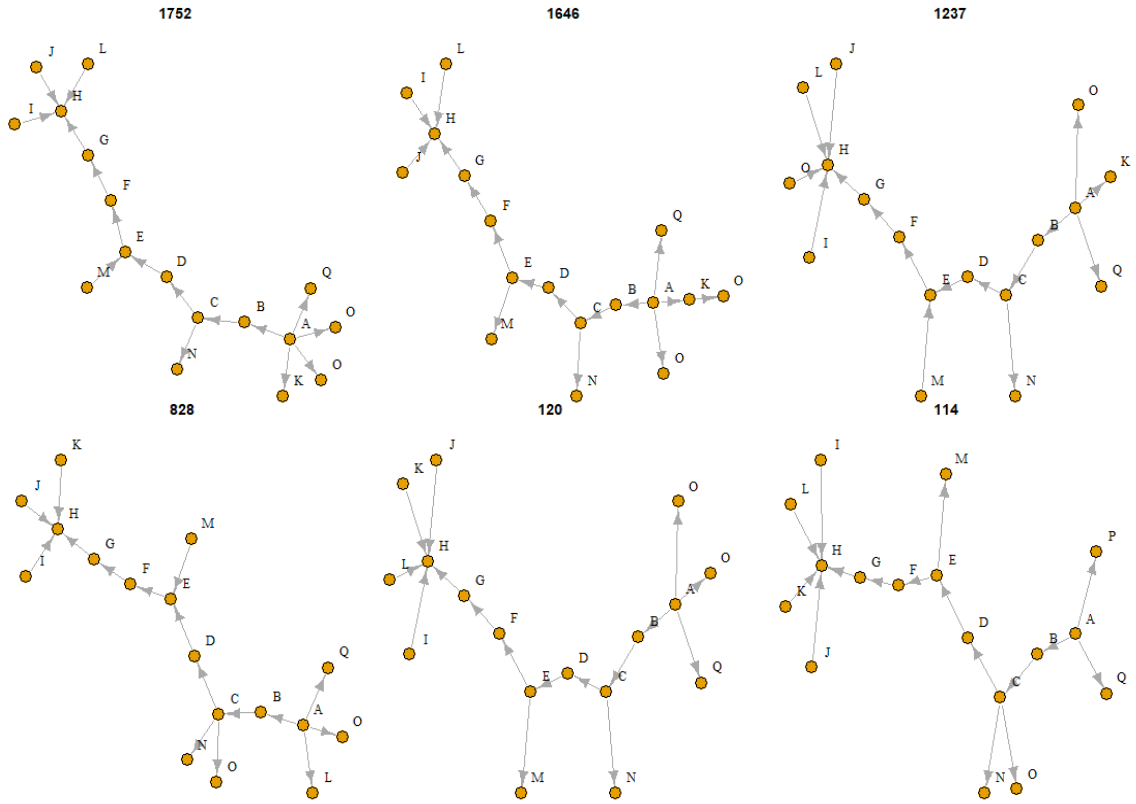


Figure B-1: Level 4 Isomorphs for Wing Center Structures. Frequency τ =4. These are six of the 2475 patterns found by mining the large (18,000 nodes, 117,000 edges) level 4 precedence network, pruned to 1,000 nodes and 18,000 edges. Each pattern is repeated at least four times in the large network.

Appendix C Frequent Subgraph Clusters

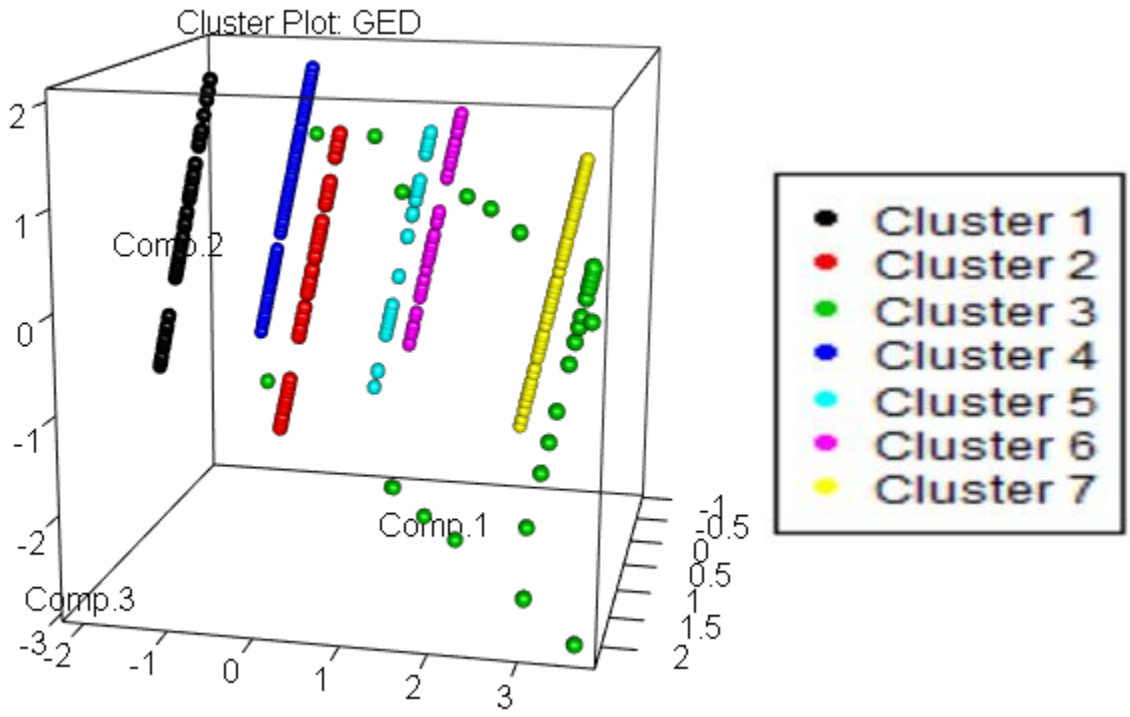


Figure C-1: Cluster Plot of Graph Edit Distance for Size 2-7 Isomorphs

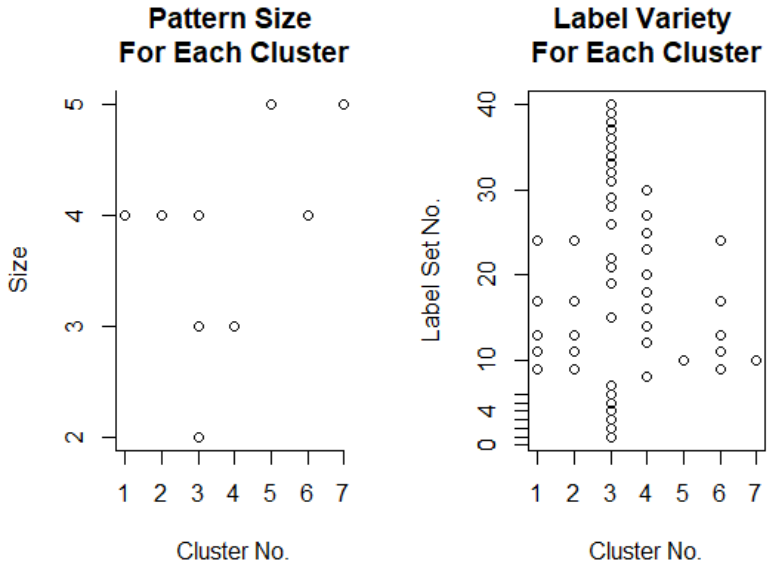


Figure C-2: Size and Label Set Number Versus Cluster No. for Size 2-7 Isomorphs.

Appendix D Triadic Closure Algorithms

Algorithm D-1: Triadic Closure Main Routine

```
FOR each level of the network (2-5)
      CALL  storeInitialOpenNodePairs
      FOR each fraction of edges removed (0.1 to 0.5)
            FOR each epoch (1-10)
                  READ network for level
                  CALL removeRandomEdges with network, fraction;
                  RETURNING modifiedNetwork with the fraction of edges
                  removed at random
                  CALL triadicCensus with modifiedNetwork
                  CALL predictEdges with modifiedNetwork, fraction
                  CALL countPredictions
            END FOR
      END FOR
END FOR
END Main routine
```

Algorithm D-2 Pseudocode for storeInitialOpenNodePairs routine

```
storeInitialOpenNodePairs routine
      FOR each product in modifiedNetwork
            FOR each pair of inputs  to the product input1 and input2
                  IF input2's inputs contains Input1 THEN
                        // Right TRIAD
                  ELSE IF input1's inputs contains Input2 THEN
                        // Left Triad
                  ELSE
                        // Open Triad
                        STORE IDs for node b and c in OpenNodePairs.
                  END IF
            END FOR
      END FOR
      RETURN OpenNodePairs.
END storeInitialOpenNodePairs routine
```

Algorithm D-3: removeRandomEdges routine

```
removeRandomEdges routine:
      SET currentInputs to the number of inputs in the network.
      SET inputsToRemain =  currentInputs * (1 - fraction)
      while currentInputs > inputsToRemain
            randomly select a node in the network
            randomly select an input to that node
            remove the selected input from the selected node
            STORE the removed edge in removedEdges list
            DECREMENT currentInputs
      END WHILE
      RETURN the modifiedNetwork.
END removeRandomEdges routine
```

Algorithm D-4: triadicCensus routine

```
FOR each product in modifiedNetwork
      FOR each pair of inputs to the product input1 and input2
            IF input2's inputs contains Input1 THEN
                  // Right Triad
                  // ........a
                  // ...... ^ ^.......... a,0,0
                  // ....../...\......... 1,b,1
                  // .... b---->c........ 1,0,c
                  // Triad Code: a-0-0|1-b-1|1-0-c|
                  // where a,b,c are replaced with the Term ID for each
                  node.
                  // where b is input1 and c is input2.
                  //
                  STORE the above Triad Code
                  INCREMENT count for triads with above Triad Code
            ELSE IF input1's inputs contains Input2 THEN
                  // Left Triad
                  // .........a
                  // ....... ^ ^........... a,0,0
                  // ......./...\.......... 1,b,0
                  // ..... b<----c......... 1,1,c
                  // Triad Code: a-0-0|1-b-0|1-1-c|
                  // where a,b,c are replaced with the Term ID for each
                  node.
                  // where b is input1 and c is input2.
                  //
                  STORE the above Triad Code
                  INCREMENT count for triads with above Triad Code
            ELSE
                  // Open Triad
                  // ..........a
                  // ........ ^ ^.......... a,0,0
                  // ......../...\......... 1,b,0
                  // ..... b......c........ 1,0,c
                  // Triad Code: a-0-0|1-b-0|1-0-c|
                  // where a,b,c are replaced with the Term ID for each
                  node.
                  //
                  STORE the above Triad Code
                  INCREMENT count for triads with above Triad Code
                  STORE unique ID's for node b and c in OpenNodePairs.
            END IF
      END FOR
END FOR
RETURN triadcount for each Triad code.
END triadicCensus routine
```

Algorithm D-5 predictEdges_TCCA routine

```
predictEdges_TCCA routine:
      FOR each product in modifiedNetwork
            FOR each pair of inputs  to the product input1 and input2
                  IF NOT input2's inputs contains Input1
                        Construct triad code for this Left Triad
                        READ census counts for this Left Triad

                        STORE the left input pair and the left triad
                         count as a predicted edge.

                  ELSE IF NOT input1's inputs contains Input2
                        Construct triad code for this Right Triad
                        READ census counts for this Right Triad
                        STORE the Right input pair and the Right
                         triad count as a predicted edge.
                  ELSE
                        Construct triad code for this Left Triad
                        READ census counts for this Left Triad as P1
                        Construct triad code for this Right Triad
                        READ census counts for this Right Triad as P2
                        IF P1> p2
                              STORE the left input pair and the left
                               triad count as a predicted edge.
                        ELSE
                              STORE the right input pair and the
                              right triad count as a predicted edge.
                        END IF
                  END IF
            END FOR
      END FOR
END predictEdges_TCCA routine
```

Algorithm D-6 predictEdges_TCPC routine

```
FOR each product in modifiedNetwork
        FOR each pair of inputs  to the product input1 and input2
                IF NOT input2's inputs contains Input1
                        Construct triad code for this Left Triad
                        READ census counts for this Left Triad
                        Construct triad code for this open Triad
                        READ census counts for this Open Triad
                        IF open triad count > 0 THEN
                                COMPUTE pOpen = openTriadCount * (1 - fraction);
                                COMPUTE pOpen = pOpen / (pOpen + triadCount for this left
                                triad);
                                if (RANDOM > pOpen)
                                        STORE left input pair, left triad count as a predicted edge.
                        ELSE
                                STORE the left input pair and the left triad count as a predicted edge.
                        END IF
                ELSE IF NOT input1's inputs contains Input2
                        Construct triad code for this Right Triad
                        READ census counts for this Right Triad
                        Construct triad code for this open Triad
                        READ census counts for this Open Triad
                        IF open triad count > 0 THEN
                                COMPUTE pOpen = openTriadCount * (1 - fraction);
                                COMPUTE pOpen = pOpen / (pOpen + triadCount for this Right
                                triad);
                                if (RANDOM > pOpen)
                                        STORE Right input pair, Right triad count as predicted edge
                        ELSE
                                STORE Right input pair, Right triad count as predicted edge.
                        END IF
                ELSE
                        Construct triad code for this Left Triad
                        READ census counts for this Left Triad
                        Construct triad code for this Right Triad
                        READ census counts for this Right Triad
                        Construct triad code for this open Triad
                        READ census counts for this Open Triad
                        COMPUTE cOpen = openTriadCount * (1.0d - fraction);
                        COMPUTE sum = triadCount1 + triadCount2 + cOpen;
                        COMPUTE p1 = triadCount1 / sum;
                        COMPUTE p2 = p1 + (triadCount2 / sum);
                        COMPUTE choice = RANDOM
                        IF choice < p1
                                STORE the left input pair and the left triad count as a predicted edge.
                        ELSE IF (choice < p2) AND (triadCount2 > 0)
                                STORE the right input pair and the right triad count as a predicted
                                edge.
                        END IF
                END IF
        END FOR
END FOR; END predictEdges_TCPC routine
```

Algorithm D-7 countPredictions routine

```
countPredictions routine
        //
        // Binary Classification:
        // True Positive: predicted link and link is present
        // False Positive: link predicted but link not present – open triads predicted
        closing but should leave open.
        // False Negative – no link predicted but link is present - Closed triads left
open incorrectly.
        //
        SORT edge predictions from highest triad count to lowest triad count.
        WHILE truePositives + falsePositives + falseNegatives >= number of removed
edges
                FOR each edge prediction
                        IF (removedEdges contains(edge Prediction)
                                        INCREMENT truePositives
                                ELSE {
                                        if initialOpenNodePairs contains edgePrediction
                                                INCREMENT falsePositives
                                        ELSE
                                                INCREMENT falseNegatives
                        END IF
                NEXT edge prediction
        END WHILE
        COMPUTE precision =  truePositives / (truePositives + falsePositives);
        COMPUTE recall =  truePositives / (truePositives + falsePositives
.                               +falseNegatives);
END countPredictions
```

Appendix E Input Recommender System Design

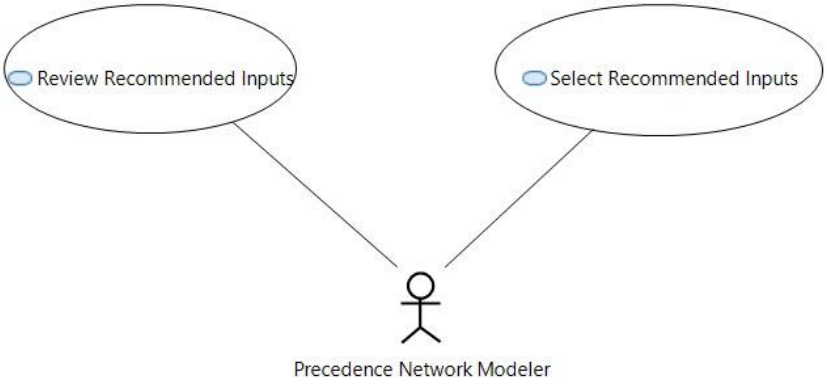


Figure E-1: Use Case Diagram

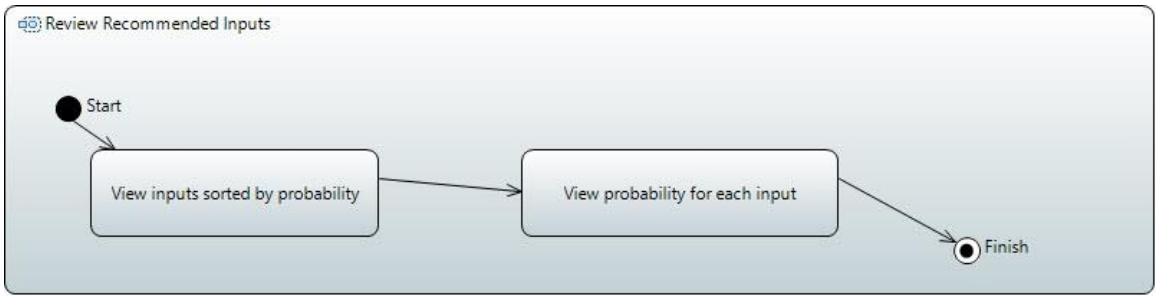


Figure E-2: Review Recommended Inputs Activity Diagram

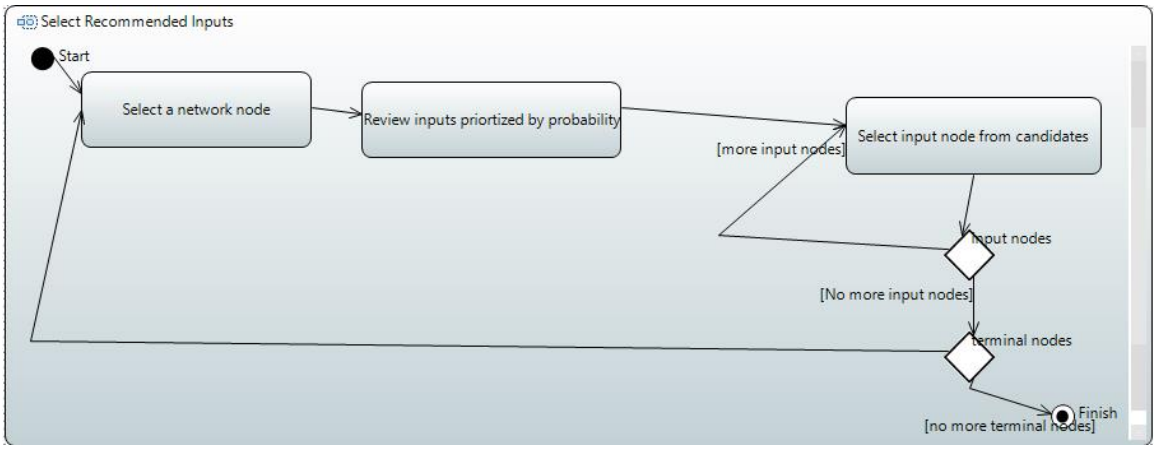


Figure E-3: Select Recommended Inputs Activity Diagram

Appendix F Component Diagram

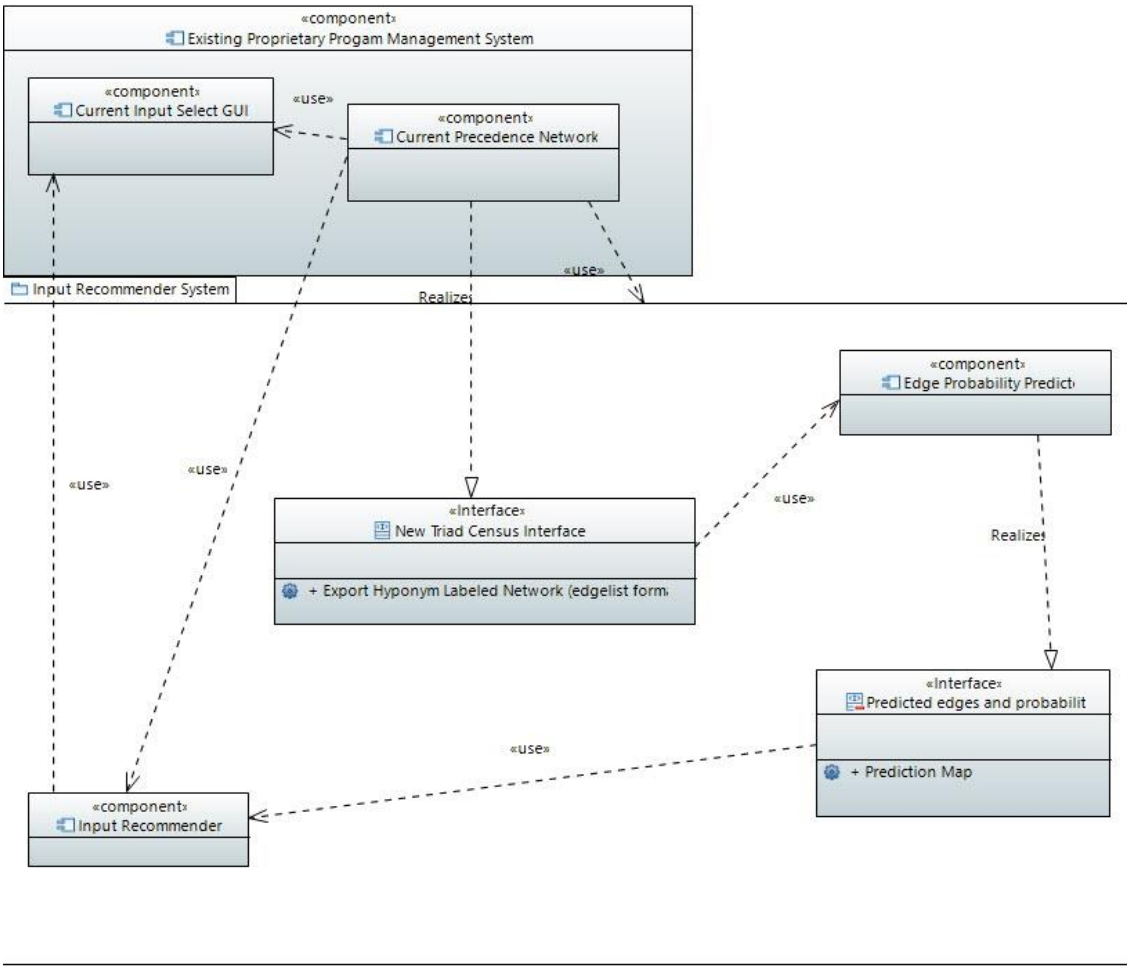


Figure F-1: Component Diagram