Social Structure Simulation and Inference using Artificial Intelligence Techniques

Maksim Tsvetovat

June 15, 2005
CMU-ISRI-05-115

School of Computer Science
Computation, Organizations and Society
Institute for Software Research, International
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Kathleen M. Carley, chair
Michael Shamos
Chistos Faloutsos
David Krackhardt, Heinz School of Public Policy

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

This work was supported in part by the National Science Foundation under the IGERT program for training and research in CASOS, and the NSF KDI 00-142 1-5-31737, NSF ITR 1040059 and the Office of Naval Research N00014-02-1-0973. Additional support was provided by CASOS - the center for Computational Analysis of Social and Organizational

Keywords: Social network analysis, multi-agent systems, simulation, terrorist networks, semantic networks,

Systems at Carnegie Mellon University. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Office of Naval Research, the National Science Foundation, or the U.S. government.

Abstract

The study of complex social and technological systems, such as organizations, requires a sophisticated approach that accounts for the underlying psychological and sociological principles, communication patterns and the technologies within these systems.

Social Network Analysis and link analysis have since inception operated on the cutting edge bringing together mathematical analysis of social structures and qualitative reasoning and interpretation.

As available computing power grew, social network-based models have become not only an analysis tool, but also a methodology for building new theories of social behaviour and organizational evolution, frequently through the creation of simulation models.

This work examines the past approaches of creating Social Network-based semantically consistent and interpretable models of social structure and social networks, as well as social simulation tools.

I propose the creation of a multi-theory, multi-level simulation model of social structure that relies on social network theory and Artificial Intelligence algorithms. I further propose the creation of a robust and scalable social structure semantic that facilitates interpretable reasoning about evolution of social structure.

For my parents

Acknowledgments

The writing of a dissertation can be a lonely and isolating experience, and it is obviously not possible without the personal and practical support of numerous people. Thus my sincere gratitude goes to my parents for their love, support, and patience over the last few years.

In the past year, I have not only written this thesis, but have also found the love of my life. Without Tanya, my fiancee, my world would have been bleak and grey. Not only she infused colour in my life, she inspired (and occasionally forced) me to write and to finish this dissertation in time for our wedding.

Support of all members of the CASOS group has exceeded all expectations. However, I would like to single out Jana Diesner for inspiring me to write and helping me maintain my sanity, Jeff Reminga for creating and maintaining the NetStatPlus library which has saved me hundreds of hours of programming, Oleg Schiglitchov and David Scofield for taking over maintenance of my side projects and letting me concentrate on writing. This work was edited by Kathy Sutton, whose talent as a writer is hidden behind the mask of an administrator.

I had many teachers in my life, but few have made such a difference in my life as Sergei Fyodorovich Ivanov of School #367 in St. Petersburg, Russia, and Dr. Maria Gini of University of Minnesota. While they are separated by thousands of miles, they carry in common a passion for their students, and a desire for them to succeed. Sergei Fyodorovich has taught me that math goes beyond the page of the homework, and can be not only useful but even fun, that computers could be programmed and not just used, and infused me with an attitude to life that will last a lifetime. Dr. Gini pulled me out of her sophomore LISP class and before I knew it I was involved in research projects where no undergrad has gone before. Without her I would not have even considered graduate school and an academic career path —

thus I should thank her for keeping me from the cubicle farm. The fact that LISP and symbolic reasoning play a significant role in this thesis — and almost every other large-scale project I have done — should be credited to her infinite patience in helping me wrap my mind around λ-calculus.

I would like to thank Professors David Krackhardt, Mike Shamos and Christos Faloutsos for asking hard questions and demanding well-written answers — as well as providing a push towards future directions of my work.

Last, but absolutely not least, I would like to thank my advisor, Professor Kathleen Carley. Not only she provided valuable guidance in all aspects of my work, she also stuck with me through many difficult times – scientific, political and personal. I am grateful for the freedom I was given to pursue side projects and explore new directions and applications of my work.

Going against the work of my advisor generally is a bad idea. However, Kathleen was always open to my ideas, including the times where I questioned the very foundations of her research. Part three of this dissertation grew out of one such idea, and proven to be not only an interesting new direction in social network analysis, but also complimentary to the existing research of the CASOS group.

Needless to say, despite all the assistance provided by Prof. Carley and others, I alone remain responsible for the content of the following, including any errors or omissions which may unwittingly remain.

Table of Contents

List of Figures

List of Tables

Chapter 1

Foreword

The study of complex social and technological systems, such as organizations, requires a sophisticated approach that accounts for the underlying psychological and sociological principles, communication patterns and the technologies within these systems.

Since inception, Social Network Analysis and link analysis have operated on the cutting edge bringing together mathematical analysis of social structures and qualitative reasoning and interpretation.

As available computing power grew, social network-based models have become not only an analysis tool, but also a methodology for building new theories of social behaviour and organizational evolution. This was frequently done through the creation of simulation models that allowed researchers to test theoretical constructs in a safe and ethical manner. Simulations also facilitate large-scale experiments and Monte-Carlo simulations - which were all but impossible in the qualitative analysis world due to cost, time and ethical constraints.

This work can be though of as three functionally independent, yet interlocking parts. Part 1 is centered on issues of simulation of complex organizational networks in general, and covert terrorist networks in particular. The simulation methodologies employed draw heavily on lessons learned from a number of subfields of artificial intelligence - planning, knowledge representation, design of multi-agent systems, and optimization techniques such as simulated annealing and randomized search.

Part 2 goes to the source of modern artificial intelligence – symbolic reasoning and object-oriented knowledge representation – and adapts these techniques to apply to qualitative machine reasoning on social structure and social network data. As result of application of AI techniques, I propose a solution to a long-standing problem of social network analysis — the problem of representing the multi-faceted and complex world of human interactions in a consistent, yet flexible way — and defining rigorous metrics that can be

applied to such data.

Part 3 is essentially an exercise in software engineering. It presents a set of software tools that address numerous inconsistencies currently present in treatment of social network data, from the day it is gathered to the day it is published. The first tool is a design for a common and consistent toolchain for integration of social network data gathering, analysis, simulation and visualization tools. Further, I present an XML-based data interchange language, which addresses the need for software tools to communicate rich social network datasets consistently and efficiently. The final tool is an enhanced SQL database designed specifically for handling, manipulating, merging and searching massive graph and social network datasets.

While the three parts present work on essentially independent projects, they carry a common underlying theme: the future of science of social network analysis depends on mining large amounts of attribute-rich, multi-modal, multi-plex, time-dependent data — and this can be successfully accomplished using lessons learned in the field of Artificial Intelligence.

1.0.1 Simulation of Complex Organizational Networks

These problems suggest the need for a new methodological approach. In chapter 2, I provide an approach based on the use of a multi-agent network model of the co-evolution of "observer" network (the blue network) and the "terrorists" (the red network) in which the observers can capture only partial data on the underlying covert network and the covert network evolves both naturally and in response to attacks by the observers. This approach builds off of organization theory and social network theory, as well as machine learning and dynamic network analysis. Specifically, I have developed a computational model of dynamic cellular organizations and used it to evaluate a number of alternative strategies for destabilization of cellular networks.

Chapter 3 builds upon the findings in social network modelling, multi-agent system engineering, and artificial intelligence to create an advanced multi-agent model of terrorist networks and their evolution. I continue to discuss techniques for sampling communications of a group of agents and building a network representation of an adversarial team. A number of algorithms and heuristics are tested through a set of virtual experiments, with results presented. I further outline deficiencies in simple communication sampling strategies such as snowball sampling, and present a new strategy based on simulated annealing.

1.0.2 Inference in Semantic Social Networks

On March 11, 2004, a series of bombs went off in commuter trains in Madrid. A few days later, I was poring over printouts of newspaper articles and work-

ing to extract the network of connections between suspected terrorists, and link it to the existing data on Al Qaeda that I already possessed. Shortly into the endeavor, I realized that most of the work I was doing consisted of consciously throwing away information that was very relevant to the types of people that were the terrorists, and to the types of connections they had between each other and to the larger organization. Yet, most of that information was not useable — the social network analysis paradigm simply had no place for qualitative descriptions of nodes and edges. I then attempted to design a consistent taxonomy of assigning weights to edges depending on qualitative relationships, and realized that assigning weights to the the edges was akin to comparing apples and oranges – the relationships were too different to be encoded on the same numerical scale.

It is there that I realized that social network analysis can benefit from a means to deal with social relational data in a qualitative, yet machine-interpretable manner.

Most mathematical analysis and social simulation tools operate on abstract numerical representations of social structures, such as graphs, matrices and time series. However, the concrete semantics behind these numbers was frequently only part of the researcher's mental model. Its communication to the outside world was largely a function of the researcher's writing skills. This, and the level of abstraction required by early computer models has resulted in datasets and models that are very difficult to interpret, especially by non-specialists.

In chapter 5, I describe creation of a language for modelling the semantics behind social networks. The language incorporates object-oriented semantics for expressing knowledge about complex social networks and builds upon this semantics to create a robust system for searching and inference in social networks.

A semantic approach to social network analysis essentially attempts to capture and recreate qualitative reasoning in a machine-driven context. This is done through modeling social networks as collections of interdependent objects. Each of the objects is defined as a semantic term — i.e., contains not only the data but also rules and methods for interpretation of the data.

The chief advantage of taking a semantic approach to reasoning about social networks is the ability to consistently describe the interactions of nodes and edges in multi-modal and multi-plex networks. Since the differences of node edge types in such networks are semantic in nature, the semantic encoding of the network structure allows the user to resolve combinatorial closures across edge types, and decompose complex interactions into sets or sequences of simpler ones.

For example, a long-standing problem in the field of social network analysis concerns integration of data in a dual-mode network, incorporating friendship and advice links. At this point, the notion of centrality in such a network

is undefined both mathematically and semantically. However, by decomposing the notions of friendship and advice into more basic semantic notions of information transfer, affinity, respect and authority, a NetInference ontology can be constructed to analyze the dual-mode network as a unit.

1.0.3 Enabling Technologies

To complete large-scale projects using sets of disparate tools such as data collection, simulation, analysis and visualization software, an issue of software interoperability must be addressed. Chapters 7 and 8 introduce the notions of software toolchain and interoperability enhancing data interchange language. Chapter 9 describes a relational database system used for accumulating large datasets consisting of rich social network data.

Part I

Artificial Intelligence and Simulation of Complex Social Systems

Chapter 2

NetWatch: Simulating and Reasoning about Dynamic Covert Networks

> Know your enemy and know yourself; in a hundred battles, you will never be defeated.
>
> Sun Tzu, "The Art of War",6th cent. B.C.

2.1 Background

For reasons of national security it is important to understand the properties of terrorist organizations that make such organizations efficient and flexible. Based on this understanding, strategies can be devised to destabilize such organizations or curtail their efficiency, adaptability, and ability to move knowledge and resources. The assessment of destabilization strategies poses a number of key challenges. What does the underlying organization look like? Does it evolve? How can the evolution of its structure be mapped through observation? What strategies could be used to destabilize such an organization? In this chapter, I provide an approach to assessing destabilization strategies that draws on work in organization science, knowledge management and computer science.

Terrorist organizations are often characterized as cellular — composed of quasi-independent cells and featuring a distributed chain of command. This is a non-traditional organizational configuration; hence, much of the knowledge in traditional organizational theory, particularly that focused on hierarchies or markets, is not directly applicable. Some lessons can be learned from previous work on distributed and decentralized organizations. This work demonstrates that such structures are often adaptive, useful in a volatile environment, and capable of rapid response [Lin and Carley, 2003][Lawrence and Lorsch, 1967]. In other words, one should expect terrorist organizations to adapt, and adapt rapidly.

Organizational form or structural design profoundly influences its performance, adaptability, and ability to move information [Baligh, Burton, and

Obel, 1990]. It follows that organizations can be destabilized by altering their structure. One caveat being that organizations, particularly more distributed and decentralized ones, evolve continuously [Aldrich, 1999].

Terrorist organizations are often characterized as dynamic networks where the connections among personnel define the nature of that evolution. This suggests that social network analysis will be useful in characterizing the underlying structure and in locating vulnerabilities in terms of key actors. Unfortunately, the dynamic nature of these networks makes it unclear whether the actors identified as key using standard network analysis will remain key long enough for destabilization tactics based on standard network analysis to be effective.

A further complication relates to the fact that the only way to obtain information about terrorist networks is by gathering intelligence — via signal interception (SIGINT) or human intelligence (HUMINT) means. By their nature, SIGINT and HUMINT techniques provide incomplete and frequently inaccurate data, and the heuristics for learning shapes of covert networks need to take this uncertainty into account. A cost factor is present as well - each piece of information comes with a price and it would be prudent to maximize its utility.

Organizations evolve as they face unanticipated changes in their environment, along with rapidly evolving technologies and intelligent, adaptive opponents. Over the past decade, progress has been made in understanding the set of factors that enable adaptation and partially validated models of adaptive networks now exist [Carley, 2002a]. A key result is that in the short run, there appears to be a tradeoff between adaptivity and extremely high performance in organizations [Carley and Ren, 2001].

Since the destabilization of terrorist networks could inhibit their ability to effect harm, there is a profound need for an approach that would allow researchers to reason about dynamic cellular networks and evaluate the potential effect of destabilization strategies. To be useful, such an approach must account for the natural evolution of cellular networks. This situation is further complicated by the fact that the information available on the terrorist network is liable to be incomplete and possibly erroneous. Hence, destabilization strategies need to be compared and contrasted in terms of their robustness under varying levels and types of information error. In other words, it would be misleading to judge destabilization strategies in terms of their impact on a static network [Carley, Lee, and Krackhardt, 2002].

In this chapter, I provide an approach based on the use of a multi-agent network model of the co-evolution of the "observer" network (the blue network) and the "terrorists" (the red network) in which the observers can capture only partial data on the underlying covert network and the covert network evolves both naturally and in response to attacks by the observers. This approach builds off organization theory and social network theory, as

well as machine learning and dynamic network analysis. Specifically, I have developed a computational model of dynamic cellular organizations and have used it to evaluate a number of alternative strategies for destabilization of cellular networks. A detailed description of the design decisions and techniques that comprise the NetWatch methodology can be found in chapter 3.

It is important at the outset to note that this examination of destabilization strategies is highly exploratory. I make no claims that the examination of destabilization strategies is comprehensive, nor that the types of "error" in the data that intelligence agencies can collect is completely described. Further, our estimate of the structure of the covert network is based on publicly available data much of which is qualitative and requires interpretation. This work should therefore be read as a study in the power of an empirically grounded simulation approach and a call for future research.

I restrict my analysis to a structural or network analysis and focus on what the covert network looks like, how its structure influences its performance and its ability to pass information, how it evolves, and how its path of evolution can be altered (its behavior destabilized) through interventions focused on the nodes. Admittedly, in this complex arena there are many other factors that are critical but they are beyond the scope of this study. Thus, from a straight social network perspective, this study suggests the types of methodological issues that will emerge when working with dynamic large scale networks under uncertainty.

To ground the ideas modelled by NetWatch, a short case description of Al Qaeda is provided with the focus on the network structure. This is followed by a discussion of the intelligence agencies engaged in anti-terrorist activity and their intelligence-gathering methodologies. My intent here is to demonstrate, at a fairly high level, the context and the resultant information and modelling problems and not provide a full analysis for intelligence or military operations. As good science often emerges from attacking hard real world problems we are trying to provide sufficient detail to understand the basis for the problems that research must address rather than simply provide a high theoretical description of general data problems. This is followed by a brief discussion of the applicability of traditional social network analysis and the need to take a dynamic network perspective. I then describe a computational model of terrorist organizations as dynamic evolving networks and anti-terrorist bodies with emphasis on their information collection and destabilization strategies. A virtual experiment used to examine destabilization strategies and the results are then discussed.

2.2 Covert Terrorist Networks - the Al Qaeda

Terrorism is a modus operandi through which targeted violence is used against non-combatants in order to achieve political objectives or strategic goals [Ruby, 2002]. Terrorist organizations can be classified as state-sponsored or extra-national. State-sponsored terrorist organizations receive direct support from their host countries. This support can manifest in various ways: financial aid to terrorists or terrorist organizations, training of terrorist operatives, up to direct involvement of governmental units of the state in terrorist attacks. Often such organizations serve as extensions of the intelligence or secret service agencies of the host countries.

Activity of such organizations can often be effectively curtailed by political or military pressure upon the sponsoring countries. State-sponsored organizations that receive direct assistance from the sponsor states also tend to be organized in a hierarchical fashion similar to the rank structure of the supporting army and essentially comprises an extension of one — and can be fought with traditional military techniques.

Extra-national terrorist groups generally serve to advance the interests of their leaders or direct backers (whether political, religious or commercial) and span multiple nations in their search for operatives and resources. They may enjoy support of one or several states whose political agendas coincide with the goals of the organization - but ultimately are not dependent on state support due to their ability to find independent financial backing from wealthy sympathizers. Generally such groups are structured in a way similar to organized crime syndicates and employ networks of quasi-independent cells scattered through the operational region of the organization as well as other countries that could be used as resource bases, recruiting and training centers.

Al Qaeda, arabic for "The Base", is the largest known extra-national terrorist organization. In 2002, it was estimated to have the support of six to seven million radical Muslims worldwide, of which 120,000 are willing to take up arms [Gunaratna, 2002]. Its reach is global with outposts reported in Europe, Middle East, East Asia and both Americas. In the Islamic world, its task is to purify societies and governments according to a strict interpretation of the Koran and to use religion as a unification force for the creation of an Islamic superpower state.

In the non-Islamic world, its task is to compel governments to withdraw their cultural influences and military ties from the Islamic world. While Al Qaeda enjoys support of wealthy individuals in a number of countries, it does not have direct support of any government. The Taliban government of Afghanistan directly supported Al Qaeda by allowing them to create training centers and bases on their territory. The involvement of the Afghan government was not crucial for the strength of the Al Qaeda organization.

In fact, the relationship between al Qaeda and the Taliban was more of an exchange with the Taliban hosting the training bases and recruiting centers and Al Qaeda providing the Taliban with trained soldiers and officers as well as serving as a domestic security service within the country [Berry, 2001].

As Goolsby[Goolsby, 2003] stated, Al Qaeda extends its reach and recruits new member cells via the adoption of local Islamic insurgency groups. Beginning with provision of operational support and resources to facilitate growth, Al Qaeda representatives work to transform an insurgency group such as Jemaah Islamiyya (Indonesia) from a group seeking political change to a full-fledged terrorist organization executing multi-casualty attacks such as the Bali bombing in 2002[Group, 2002].

Al Qaeda's global network, as we know it today, was created while it was based in Khartoum, from December 1991 to May 1996. To coordinate its overt and covert operations as Al Qaeda's ambitions and resources increased, it developed a decentralized, regional structure. Al Qaeda pursues its objectives through a network of cells, associate terrorist and guerilla groups and other affiliated organizations. For instance, the Sudanese, Turkish and Spanish nodes ran clandestine military activities in Europe and North America.

The worldwide cells appear to have no formal structure and no hierarchy. Assignments are often carried out by individuals and small groups designated for the purpose as "the person responsible". The regional nodes appear not to have a fixed location and move quickly when dictated by the political situation in the region. Al Qaeda operatives share expertise, provide resources resources, discuss strategy and eventually conduct joint operations with regional terrorist groups.

Although the *modus operandi* of Al Qaeda is cellular, familial relationships play a key role. As an Islamic cultural and social network, Al Qaeda members recruit from among their own nationalities, families and friends. What gives Al Qaeda its global reach is its ability to appeal to Muslims irrespective of their nationality, enabling it to function in East Asia, Russia, Western Europe, Sub-Saharan Africa and North America with equal facility.

Unlike conventional military forces which are hierarchical and centralized, terrorist militant units are generally small, geographically dispersed and, at the first glance, disorganized. Nevertheless, they have been able to effectively counter much larger conventional armies. Large terrorist organizations operate in small, dispersed cells that can deploy anytime and anywhere [Ronfeldt and Arquilla, 2001]. Dispersed forms of organization allow these networks to operate elusively and secretly.

The apparent structure of the Al Qaeda is not exclusive to such militant or terrorist groups. Indeed, they bear a familiar resemblance to the structure of other resistance groups. For example, a study published in 1970 by L. Gerlach and V. Hine [L.P.Gerlach and V.H.Hine, 1970] concluded that U.S. social movements, such as the environmental and anti-war movements

in the 1960s, were structured as "segmented, polycentric, and ideologically integrated networks" (SPINs):

> "By segmentary I mean that it is cellular, composed of many different groups... . By polycentric I mean that it has many different leaders or centers of direction... . By networked I mean that the segments and the leaders are integrated into reticulated systems or networks through various structural, personal, and ideological ties. Networks are usually unbounded and expanding... . This acronym [SPIN] helps us picture this organization as a fluid, dynamic, expanding one, spinning out into mainstream society."

The dynamics exhibited by SPINs appear to exist in both these social movement groups as well as in various terrorist, criminal and fundamentalist networks around the world [Ronfeldt and Arquilla, 2001].

However, unlike many protest movements, terrorist and criminal networks must remain covert. The need for security dictates that terrorist organizations must be structured in a way that minimizes damage to the organization from arrest or removal of one or more members [Erickson, 1981]. This damage may be direct - making key expertise, knowledge or resources inaccessible for the organization, or indirect - exposing other members of the organization during interrogations. There are several factors that allow a terrorist organization to remain covert, including:

- Strong religious (in case of Islamic groups) or ideological (in case of Sendero Luminoso and other South American guerilla groups) views that allow members to form extremely strong bonds within a cell.
- Physical proximity among cell members, often to the extent of sharing living quarters, working and training together.
- Lack of rosters on who is in which cell.
- Cell members being given little knowledge of the organizational structure and the size of the organization.
- Inter-cell communication on as-needed basis only.
- Information about tasks issued on a need-to-know basis so very few people within the organization know about the operational plans in their entirety.

A need-to-know information policy can be counterproductive when an organization needs to complete a task that is larger than any one cell. Further, such policies tend to lead to duplication of effort and reduce the ability of one

cell to learn from another. To fix these inefficiencies, terrorist organizations have been known to employ "sleeper links" - where a small number of members of each cell have non-operational ties (such as family ties, ties emerging from common training, etc) to members of other cells [Krebs, 2001]. These links are rarely activated and are used mainly for coordinating actions of multiple cells in preparation for a larger operation.

To remain covert, the Al Qaeda has structured itself as a leaderless design characterized by its organic structure, horizontal coordination, and distributed decision making. However, the need to maintain a strong ideological foundation and resolve coordination issues has led to the need for strong leadership. One apparent solution has been to have multiple leaders diffused throughout the network and engaged in coordinating activities without central control or a hierarchy among the cells. Whether the leaders are themselves hierarchically organized, even though the cells are not, is less clear.

Under constant pressure from various world governments, terrorist organizations have evolved a structure that appears to be resilient to attacks. However, information on these terrorist organizations, their membership, the connections among the members is, at best, incomplete. Available information is often obtained during *post-factum* investigations of terrorist acts and may offer little insight into the "main body" of the organization or the way in which it is evolving.

Substantial intelligence effort is needed to piece together the massive amount of often misleading information, both post-factum and "logs" of activity, to generate a picture of the entire organization. Nevertheless, the picture that is emerging suggests that terrorist organizations are organized at the operational level as *cellular networks* rather than as hierarchies [Carley, 2003a].

2.3 Open-Source Data on Terrorist Networks

Until recently, social network datasets were extremely difficult to obtain and limited in size and scope. The prevailing methodology for collecting social network data was by survey, either administered to an entire group of people or collected in a snowball fashion. Collection of social network data was done in a way reminiscent of anthropological data collection - by a human observer embedded into an organization to be studied.

This presented a number of problems. First of all, it was very costly to collect all but the smallest of datasets. While a number of sampling strategies were investigated, it was difficult or infeasible to canvass a larger organization or population. Furthermore, presence of an observer or a survey instrument in an organization inevitably altered the behaviour of individuals

Figure 2.1: Data on Hamas collected by AutoMap

in the organization.

Finally, for some networks, especially terrorist networks, it is physically impossible to collect a dataset via direct survey administration. The modus operandi of such networks is covertness and this necessarily limits the data that can be collected on them.

Thus, for study of terrorist organizations, one must obtain information via indirect means. One approach to gathering indirect social network data is via analysis of texts. Originally used as manual coding technique, text analysis has now been automated to extract network structure from corpora of text based on co-appearance of people, organizations and other entities.

Between September 14, 2001 and November, 2001 Valdis Krebs[Krebs, 2001] assembled a corpus of texts regarding events preceding September 11th attacks. Manual analysis of these texts yielded a dataset (see figures 2.1 and 2.2) which became one of the definitive sources of data on terrorist organizations and structure of a terrorist plot.

Since 2001, much larger datasets on covert networks are available due to both increased interest in the research and improvements in tools for machine collection of network data.

Some of the newer more complete datasets include these collected by IntelCenter[IntelCenter.com, 2003], R. Renfro[Renfro, 2003] and M. Sageman[Sageman, 2004]

In the aftermath of the September 11th attacks, it was noted that coher-

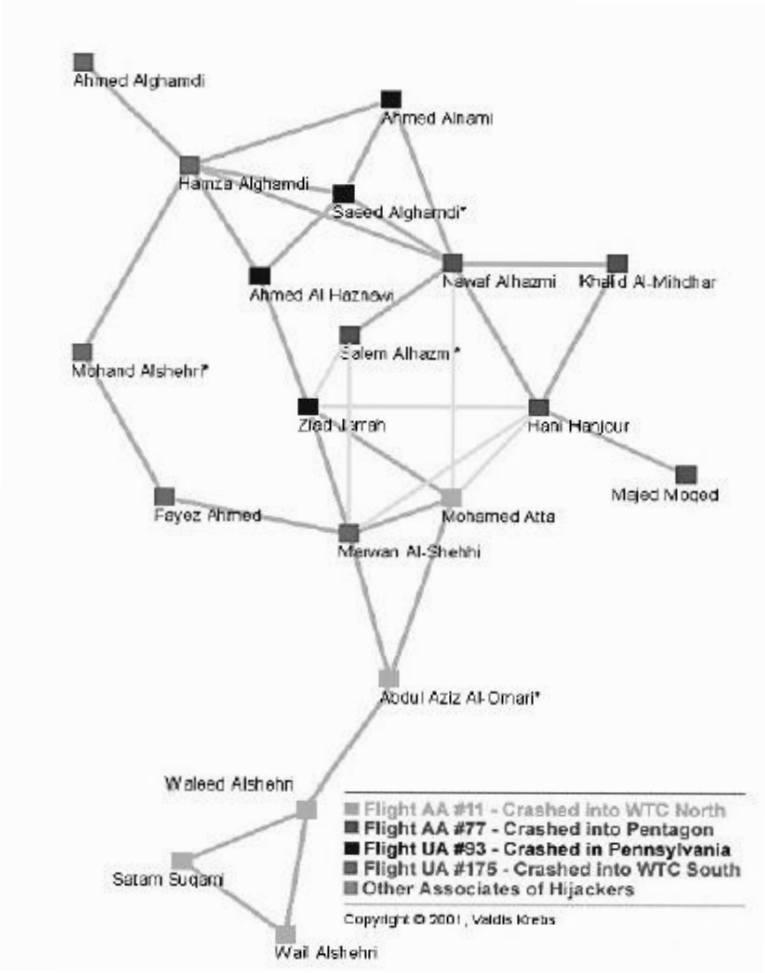


Figure 2.2: Data on September 11th hijackers collected by Valdis Krebs

ent information sources on terrorism and terrorist groups were not available to researchers[Gruenwald, McNutt, and Mercier, 2003]. Information was either available in fragmentary form, not allowing comparison studies across incidents, groups or tactics, or made available in written articles - which are not readily suitable for quantitative analysis of terrorist networks. Data collected by intelligence and law-enforcement agencies, while potentially better organized, is largely not available to the research community due to restrictions in distribution of sensitive information.

To counter the information scarcity, a number of institutions developed unified database services that collected and made available publicly accessible information on terrorist organizations. This information is largely collected from open source media, such as newspaper and magazine articles, and other mass media sources.

Such open-source databases include:

- RAND Terrorism Chronology Database[Corporation, 2003] - including international terror incidents between 1968 and 1997
- RAND-MIPT (Memorial Institute for Prevention of Terrorism) Terrorism Incident Database[Houghton, 2002], including domestic and international terrorist incidents from 1998 to the present
- MIPT Indictment Database[Smith and Damphousse, 2002] - Terrorist indictments in the United States since 1978.

  Both RAND and MIPT databases rely on publicly available information from reputable information sources, such as newspapers, radio and television.

- IntelCenter Database (ICD)[IntelCenter, 2005] includes information on terrorist incidents, groups and individuals collected from public sources, including not only traditional media outlets and public information (such as indictments), but also information learned from Middle East-based news wire services. Separately, IntelCenter also collects information from Arabic chat-rooms and Internet-based publications - although value of such data is questionable and data may be tainted by propaganda.

2.4 Terrorist Organizations and Scale-Free Networks

An argument has been made[Robb, 2004] that terrorist networks may exhibit features of scale-free networks and can thus be treated as such in analysis and derivation of attack scenarios.

Scale-free networks have been observed in many contexts ranging from networks of airline traffic to sexual networks and Web link patterns. The high probability of emergence of scale-free networks, as opposed to evenly distributed random networks, is due to a number of factors, including:

- Rapid growth confers preference to early entrants. The longer a node has been in place the greater the number of links to it. First mover advantage is very important.
- In an environment of too much information people link to nodes that are easier to find - thus nodes that are highly connected. Thus preferential linking is self-reinforcing.
- The greater the capacity of the hub (bandwidth, work ethic, etc.) the faster its growth.

It has also been observed that scale-free networks are extremely tolerant of random failures. In a random network, a small number of random failures can collapse the network. A scale-free network can absorb random failures up to 80% of its nodes before it collapses. The reason for this is the inhomogeneity of the nodes on the network – failures are much more likely to occur on relatively small nodes.

However, scale-free networks are extremely vulnerable to intentional attacks on their hubs. Attacks that simultaneously eliminate as few as 5-15% of a scale-free network's hubs can collapse the network. Simultaneity of an attack on hubs is important. Scale-free networks can heal themselves rapidly if an insufficient number of hubs necessary for a systemic collapse are removed.

Scale-free networks are also very vulnerable to epidemics. In random networks, epidemics need to surpass a critical threshold (a number of nodes infected) before it propagates system-wide. Below the threshold, the epidemic dies out. Above the threshold, the epidemic spreads exponentially. Recent evidence[Pastor-Satorras and Vespignani, 2001] indicates that the threshold for epidemics on scale-free networks is zero.

However, the reality of terrorist networks does not fit neatly into the scale-free network model. It has been observed[Rothenberg, 2002] that non-state terrorist networks are not only scale-free but also exhibit small world properties. This means that while large hubs still dominate the network, the presence of tight clusters (cells) continue to provide local connectivity when the hubs are removed.

For example, attack on Al Qaeda's Afghanistan training camps did not collapse its network in any meaningful way. Rather, it atomized the network into anonymous clusters of connectivity until the hubs could reassert their priority again. Many of these clusters will still be able to conduct attacks even without the global connectivity provided by the hubs.

Furthermore, critical terrorist social network hubs cannot be identified based on the number of links alone. For example, Krebs observed[Krebs, 2001] that strong face-to-face social history is extremely important for trust development in covert networks. Of similar importance is the relevance of skills and training of agents inside a cell to the task at hand. Thus, importance of any individual within the network should be rated on a vector of factors pertaining to its qualities as an individual as well as types and qualities of its links.

Rothenberg[Rothenberg, 2002] notes that postulating a path of a set length from everyone in the global network to everyone else (i.e. scale-free nature of a terrorist network) runs contrary to the instructions for communication infrastructure set forth in the Al Qaeda training manual[Al-Quaeda, 2001]. Thus, if a terrorist network was observed to be scale-free, it can be argued that its scale-free nature is not a matter of design and can possibly be an artifact of the data collection routines. For example, snowball sampling[Granovetter, 1976] is biased toward highly connected nodes, so extensive use of this technique may result in observation of scale-free core-periphery structures where none exist[Biernacki and Waldorf, 1981].

2.5 Developing the Formalism of a Cellular Network

Given the case studies of Al Qaeda and other terrorist networks, it is clear that terrorist organizations cannot be adequately described as random graphs or as scale-free networks. Therefore, a different model of terrorist networks has emerged, namely cellular networks [Rothenberg, 2002][Carley, Dombroski, Tsvetovat, Reminga, and Kamneva, 2003][Carley, Lee, and Krackhardt, 2002]. While this model may not fit a simple mathematical definition such as scale-free or small-world network, its base is in empirical and field data[Goolsby, 2003]. In section 6.0.1, I will show that cellular networks in fact are not characterized by a lack of a formal representation but are defined through a more complex process which takes as a goal improvement of fit between the model network and empirical data.

Cellular networks[Carley, 2003a] are different from traditional organizational forms as they replace a hierarchical structure and chain of command with sets of quasi-independent cells, distributed command, and rapid ability to build larger cells from sub-cells as the task or situation demands. In these networks, the cells are often small and are only marginally connected to each other. The cells are distributed geographically, and may take on tasks independently of any central authority[Carley, 2003b].

Rothenberg[Rothenberg, 2002] observed a number of properties of a cellular network:

- The entire network is a connected component.

  > ...It is likely that on the local level, individual ties are very strong...On the higher level, individual ties are likely to be weaker but the strength of association [people known in common, doctrine] is likely to remain high...

- The network is redundant on every level: Each person can reach other people by multiple routes - which can be used for both transmission of information as well as material. On the local level, there will be a considerable structural equivalence[Tsvetovat and Carley, 2005], which will ameliorate the loss of an individual. The redundancy in communication channels may also be mirrored in the redundance of groups engaged in a particular task.

- On the local level, the network is small and dynamic, consisting of small cells (4-6 people) that operate with relative independence and little oversight on the operational level.

- The network is not managed in a top-down fashion. Instead, its command structure depends on vague directives and religious decrees, while leaving local leaders the latitude to make operational decisions on their own.

- The organizational structure of a terrorist network was not planned, but emerged from the local constraints that mandated maintenance of secrecy balanced with operational efficiency.

Each cell is, at least in part, functionally self-sufficient and is capable of executing a task independently. Cells are loosely interconnected with each other for purposes of exchanging information and resources. However, the information is usually distributed on a need-to-know basis and new cell members rarely have the same exact skills as current members. This essentially makes each individual cell expendable. The removal of a cell generally does not inflict permanent damage on the overall organization or convey significant information about other cells. Essentially, the cellular network appears to morph and evolve fluidly in response to anti-terrorist activity[Sageman, 2004].

This leads to a hypothesis that cells throughout the network contain structurally equivalent[F.Lorrain and White, 1971] and essential roles, such as ideological or charismatic leaders, strategic leaders, resource concentrators and specialized experts.

Given this hypothesis, one can further reason that operations of a particular cell will be affected in a negative way by the removal of an individual

filling one of these roles. I further posit that a further development of a cellular network formalism as an empirically driven and yet mathematically sound concept, is necessary for creation of computational models that combine face validity towards real-world data as well as veridicality towards formal models of organizational evolution.

2.6 Robust Representation of Organizational Data

Traditional social network analysis (SNA) techniques have focused on analysis of communication networks between individuals. Moreover, most SNA studies have been conducted on single-mode networks (i.e. relationships between people) with binary data (i.e. presence or absence of a connection). Also, most studies have been concerned with analysis of a single network.

Krackhardt and Carley [Krackhardt and Carley, 1998] proposed concentrating knowledge about an organization in a format that could be analyzed using standard network methods, called the MetaMatrix. The MetaMatrix approach treats organizations as evolving networks. Actors in these networks actively engage in processes defined as task performance, knowledge exchange and resource transfer. This conceptualization made it possible to link organizational performance to social networks.

Carley [Carley, 1999] [Carley, 2002c] generalized this approach and extended the perspective into the realm of knowledge networks enabling the researcher to question how changes in the social network could effect changes in the distribution of information and the resultant impact of knowledge disruption strategies on organizational performance. By taking an information processing perspective we are explicitly linking knowledge management and social networks[Carley and Hill, forthcoming] and enabling network evolution through learning mechanisms. From a conceptual and data perspective this means that we examine the co-evolution of all networks in the meta-matrix as described in table 2.1.

2.6.1 MetaMatrix Measures

A number of metrics have been defined on the MetaMatrix models. These metrics can be used to estimate the likelihood of a new link being formed between two agents, and find critical or redundant nodes in the network and locate emergent leaders based on their cognitive demand.

Homophyly and Relative Expertise

The following two measures are used to estimate the probability of creation of a new communication link in the social network or the motivation to communicate. Empirical studies of human communication behavior suggest that, without any external motivation, individuals will spend about 60% of the time interacting on the basis of homophyly and 40% on the basis of need.

Carley has defined homophyly[Carley, 2002c] to be based on a measure of relative similarity RS between agent i and agent j: the amount of knowledge

	People	Knowledge and Skills	Resources	Tasks
People	Structural knowledge: command structures and relationships between agents	Knowledge Network: who has access to what knowledge	Resource Network: who can use what resources	Task Assignment: who does which tasks
Knowledge		Knowledge Precedence: types of skills that go together	Resource Skills: skills needed to use a resource	Skill Requirements: skills needed to accomplish a task
Resource			Resource Precedence: which types of resources go together	Resource Requirements: Which resources are needed to accomplish a task
Task				Task Precedence: the sequencing and precedence of tasks.

Table 2.1: Meta-Matrix of Organizational Knowledge

that i and j have in common divided by the amount i shares with all other agents (including self), or

$$RS_{i,j} = \frac{\sum_{k=0}^{K}(S_{ik}S_{jk})}{\sum_{l=0}^{I}\sum_{k=0}^{K}(S_{ik}S_{lk})} \tag{2.1}$$

where S_{ik} is 1 if agent i knows fact k and 0 otherwise.

In contrast, relative expertise is defined from a purely knowledge perspective: how much agent i thinks j knows that i does not know divided by how much i thinks all others know that i does not know, or

$$RE_{ij} = \frac{\sum_{k=0}^{K}((1 - S_{ik}) * S_{jk})}{\sum_{l=0}^{I}\sum_{k=0}^{K}((1 - S_{ik}) * S_{lk})} \tag{2.2}$$

Cognitive Demand: Finding emergent leaders

Cognitive demand, described by Carley [Carley and Ren, 2001], measures the amount of effort each person expends in performing actual tasks using the knowledge, resource, task and communication networks of the MetaMatrix.

Cognitive demand is a notion similar to the task load measure developed at NASA [Hart and Staveland, 1988]. It measures the extent to which the person has to engage in mental activity to do the assigned tasks, defined as:

1. number of people person i interacts with / total number of people in the group
2. number of tasks person i is assigned to / total number of tasks
3. sum of number of people who do the same tasks person i does / (total number of tasks * total number of people)

The *cognitive demand* measure combines static measures of centrality with dynamic measures of information flow, task performance and resource distribution. These measures are based on the meta-matrix knowledge about the organization and have been shown to accurately detect emergent leaders in an organization.

Key players: Task and Knowledge Exclusivity

Understanding the relative criticality of employees is important in managing turnover and security risks associated with human capital in organizations. Traditional social network analysis measures are based on static, survey-based assessments of centrality and other sociometric aspects of organizations. This limits their effectiveness in fully evaluating human capital criticality, particularly criticality that may be "hidden" in the non-social dimensions of an organization.

M. Ashworth and K. Carley [Ashworth and Carley, 2002][Ashworth, 2003] introduced new task- and knowledge-based measures based on the MetaMatrix[Krackhardt and Carley, 1998] designed to overcome such limitations. Their results suggest that while each class of measures provides useful insight on criticality of organization actors, knowledge-based measures provide the most robust predictions of each actor's contribution to organizational performance.

Ashworth and Carley[Ashworth, 2003] proposed a number of new task- and knowledge-based measures, the first of which is a Task Exclusivity Index(TEI) that essentially measures the extent to which each actor is the only one who can do certain tasks.

The second proposed measure, The Knowledge Exclusivity Index (KEI), measures the extent to which each actor is the only one who possesses certain skills, knowledge or expertise.

The study found that no single measure or class of measures perfectly identifies all critical employees but that a heuristic application of the proposed knowledge-based measures results in the highest overall accuracy.

2.6.2 Applications of MetaMatrix Analysis

The measures detailed above have found a number of uses in both analysis of corporate structure and adversarial networks.

Schreiber and Carley[Craig Schreiber, 2004] link the concepts of the MetaMatrix and measures based on MetaMatrix data with personnel data from NASA Jet Propulsion Laboratory. The paper shows that measures of cognitive demand and knowledge exclusivity are accurate predictors of turnover risk posed by key employees. The study used the collected data to inform simulation and project the impact of turnover within the subject teams.

Further, the Organizational Risk Analyzer (ORA) tool developed by J.Reminga and K.Carley[Carley and Reminga, 2004] frames the multitude of social network and MetaMatrix measures into a framework of a consistent risk report. The risk reports issued by ORA point out critical personnel, measure turnover risk as well as incongruities in information and resource distribution.

While the main use of ORA is to improve performance of organizations and corporate networks, the tool can be also used to point out vulnerabilities in adversarial and covert networks - thus finding effective and convenient avenues of attack against them.

Another organizational analysis tool, DyNet[Carley, Dombroski, Tsvetovat, Reminga, and Kamneva, 2003] enables reasoning about dynamic networked organizations by adding a simulation component to MetaMatrix analysis. The core tool is DyNet - a reasoning support tool for reasoning under varying levels of uncertainty about dynamic networked and cellular organizations, their vulnerabilities, and their ability to reconstitute themselves. Using DyNet the analyst would be able to see how the networked organization was likely to evolve if undisturbed, how its performance could be affected by various information warfare and isolation strategies, and how robust those strategies were in the face of varying levels of information assurance.

Finding key players in a dynamic organizational network is of top importance for analysis of covert networks. Covert networks often exhibit informal or highly spread out command structures that make emergent leaders (such local operational chiefs) pivotal for day-to-day operation of the organization. Elimination of these emergent leaders has proven an effective strategy for decreasing the operational capacity of terrorist organizations.

NetWatch (see chapter 2.7) has been implemented as a simulation testbed for detecting key players in dynamically evolving covert networks. One of the major findings in NetWatch is the emergence of a recovery behaviour

in a dynamic network, as key players were eliminated[Tsvetovat and Carley, 2003].

None of static key player detection algorithms predicted this recovery and while using Knowledge Exclusivity Index to detect and eliminate emergent experts was effective, it suffered greatly from lack of data (knowledge data is more difficult to find through signal intelligence then simpler connection data).

Using domain knowledge and symbolic reasoning, as I propose, will enable researchers to define and search for patterns in the network structures, such as patterns of succession in event of contingencies. Moreover, results obtained from a symbolic reasoner are more interpretable by human users then these from a statistical measure - and thus more effective in mixed initiative scenarios.

2.6.3 Caveats

Meta-matrix data for the entire organization affords a birds-eye view of the organization at a point in time, similar to a static snapshot. Moreover, if collected in the real world, it provides a picture of the entire organization, which is quite distinct from what a single person is likely to know. A set of such matrices over time represent the change trajectory for an organization. Boundedly rational individuals make decisions and operate in a climate of uncertainty with incomplete and inaccurate knowledge of the world. In essence, they make decisions using their perception of the meta-matrix - their personal knowledge and their knowledge of other's relations (*transactive memory* [Wegner, 1987]).

This distinction between the actual meta-matrix and the agent's perception is at the heart of the difference between social network analysis, agent modelling, and multi-agent network simulation.

Social network and MetaMatrix analysis affords researchers considerable power at representation of social structure. Social network data can be used to analyze organizations in terms of information flow and structural inconsistencies. Furthermore, findings in social network theory have some predictive power, allowing one to make estimates regarding further evolution of a network given its current state.

However, static social network analysis does not capture the dynamic trends in organizations and does not provide rigorous semantic framework for such analysis. To fully capture dynamism inherent in human networks, one must turn to computer-enabled approaches of simulation models. In the next chapters, I discuss creation of such models and their descriptive power.

2.7 Simulating Covert Networks

While shifts from analyzing single-mode networks to working with complex multi-mode and multi-plex networks such as the MetaMatrix resulted in a significant increase in the fidelity of representation of human relations, one important aspect remained un-addressed: the dimension of time. While static analysis may be adequate for slow-changing interpersonal networks, covert networks are characterized by their fluidity and dynamism. Thus, analysis of covert networks needs to be approached from a dynamic perspective, tracking change inside the network as well as its static parameters.

Human networks evolve continuously and rapidly. Understanding the dynamics of network evolution under different circumstances is paramount to being able to predict effects of policies and management strategies.

A number of approaches at introducing dynamic qualities to network analysis have been proposed including analytical models, artificial-life based models and multi-agent network models.

Analytical and statistical models, are generally designed to study effects of a number of distinct variables on an observable or measurable outcome. Relationships between causes and outcomes are generally established via statistical analysis. While analytical and statistical models provide valuable insights into causal relationships, they do fairly poorly at representing complex systems. This is mainly due to weak scalability of statistical models in relation to growth of dimensionality of the problems space. As number of independent variables (i.e. complexity of representation of individual actors), and number of dependent variables (i.e. complexity of representation of the resulting system) increases, the analysis quickly becomes mathematically intractable. This growth in mathematical complexity forces researchers to take a reductionist view of the phenomena or the outcomes, thus limiting the ability to study interactions between local rules and global behaviours.

Other analytical models such as Blanche[Contractor and Monge, forthcoming], combines modelling of social structure and individual attributes. However, application of analytical models to real world problems have been shown to be computationally intractable, resulting in over-simplified models.

A different class of models based on equations of system dynamics[Karnopp, Margolis, and Rosenberg, 1990] or expert system technologies [Burton and Obel, 1998][Laird, Newell, and Rosenbloom, 1987] strives toward maximum veridicality of the phenomena, albeit by very different means.

The goal of System Dynamics[Karnopp, Margolis, and Rosenberg, 1990] is study of overall system behaviour by means of systems of differential equations modeling processes and flows in the system. Thus, system dynamics models can achieve very high degree of veridicality on the overall system level. However, the complexity of design of such models hinders their understandability on the level of an individual actor.

Richard Burton, in his OrgCon system[Burton and Obel, 1998] approximates overall system behaviour by creating a complex rule-based system that is rooted in literature of management science and organizational behaviour. While the expert system is capable of making verifiable and understandable statements about the subject organization, its main limitation lies in the fact that rules at the heart of the system have been developed through top-down observation of organizational behaviour as opposed to a bottom-up emergence of organizational-level behaviours from individual-level decisions.

SOAR[Laird, Newell, and Rosenbloom, 1987] achieves high veridicality, but at the other end of the spectrum. SOAR individual-level models are so complete and accurate for some of the domains, that it can outperform human actors in some of the tasks (e.g. piloting a plane in a flight simulator). However, SOAR (as many other AI systems) is designed to be a "better human then humans" - faster, more accurate, unaffected by emotion and discomfort. This quality makes SOAR-based simulations of social phenomena prone to under-estimation of the variables that make humans vague and unpredictable - often the very things that make us human. However, SOAR rule bases can be rewritten for simulation, rather then emulation (i.e. replicating the subject matter with all of its imperfection rather then striving to achieve the task perfectly).

Artificial life and cellular automata models have been shown to reflect some of the complexity and dynamics exhibited by social systems. However, artificial life simulations generally do not operate on empirical data, are constrained to grid based structure, and represent cognition at a high level of abstraction. This results in an oversimplified view of the phenomena.

Multi-agent systems can serve as effective tools for reasoning about human and group behavior. Their effectiveness is enhanced when the algorithms lead the simulated agents to behave as humans behave, rather than doing what is optimal for the task. Such systems are even more effective when the model's input is real data and the generated outputs are comparable to the actual data files in the real world. Such systems can be created by combining sophisticated planning and learning algorithms with extensive knowledge of human behavior and underlying networks.

Elliot and Kiel2002 propose treatment of terrorist organizations a system of "fluids" combined as a complex adaptive system. They create a complex-system model of information flows within a terrorist organization based on principles of fluid dynamics and finite element analysis.

In traditional social network analysis, behavioral interpretations are drawn from the actual network, which is viewed as constraining and enabling behavior. In agent modeling systems, agents are designed to act optimally for the task at hand.

In multi-agent network simulations, agents act in a boundedly rational fashion on the basis of their personal perception by emulating what people

might do. In order to produce a high-fidelity model of a dynamic organization, it is necessary to simulate the world of the agents, including all imperfections. We enable this by affording every agent a *belief structure*, which is essentially the agent's private meta-matrix structure populated as the agent interacts with others, learns and performs tasks. In the simplest case, the meta-matrix contains binary values whose meaning is simply existence or lack of connection between two nodes. However, this can be extended with weights assigned to every edge. For example, edge weights in the interpersonal network can be interpreted as trust or frequency of communication, as opposed to the existence of a connection.

While studying real-world networks, and specifically covert networks such as terrorist organizations, data can be collected for each cell of the meta-matrix, but it is very difficult to obtain a complete and accurate snapshot of the organization. A key difficulty is in discerning whether a change in the networks is due to better intelligence or actual changes in the network. Thus, analysis algorithms and simulation systems must be able to operate under uncertainty, provide confidence estimates for results, and approach the domain from a satisficing rather then optimization perspective.

Networks can be defined by a number of interdependent levels of analysis: personal, group, organizational, inter-organizational and community[Stohl and Stohl, 2002]. Historically, network research was conducted at the individual or small-group level and rarely approached from several of the levels simultaneously[Contractor and Monge, forthcoming].

However, at every level networks are dynamic and a significant proportion of the links are in flux while the overall shape of the network remains somewhat stable across time. Carley, Lee and Krackhardt[Carley, Lee, and Krackhardt, 2002] state "Whether the topic is terrorism, the global economy, or the nature of the Internet, we are dealing with complex socio-technical systems that are large, multi-lex, multi-modal and adaptive. It is critical that we ... develop a new set of tools ... to meet this challenge of understanding, predicting and explaining behaviour of multi-agent networks".

In the case of terrorist networks, through computational models scholars now possess the new ability to look at multiple levels over multiple time points and to explore how communication processes constitute and shape the organization. This is yielding important results in understanding the way such networks develop and operate in the social settings of their host countries or span national boundaries.

One of the emerging research approaches within the field of social simulation is agent-based modeling (ABM). The goal of ABM is creation of computer-based micro-worlds in which heterogenous agents interact on the world with both reacting to the surrounding conditions and effecting change. Agent-based models of complex adaptive systems are frequently based on the following principles[Langton, 1989]:

1. The model consists of a population of simple agents.
2. There is no single agent that directs all of the other agents.
3. Each agent details the way in which a simple entity reacts to local situations in its environment, including encounters with other agents.
4. There is no rule in the system that dictates global behaviours.
5. Any behaviour at levels higher then individual agents is therefore emergent.

While the above principles produce many interesting multi-agent models of social and organizational behaviour, the main strength of the agent-based methodology outlined by Langton is also its main weakness.

Langton's first principle emanates from the general assumption within the sciences of complexity that "simple rules can generate complex behaviours and structures". However, this mandates the practitioner of the art of agent-based modeling to create, for each studied domain, sets of rules that fulfill two seemingly contradictive requirements.

First and foremost, the rules must adequately represent the subject of study. This can be formalized by maximizing the behavioral likeness of the individual agent and the subject of study (e.g. a terrorist).

However, the above rules need to also be simple and reliant on a fully myopic view represented in Langton's third principle.

The pressures of reconciling simplicity and veridicality in general produce a tendency towards simplicity, thus sacrificing face validity of the emergent behaviours or the model itself[Carley, 2002b]. Furthermore, the temptation upon the practitioner to create "interesting" emergent behaviours can result in local rules within each agent that are designed to implicitly generate or alter a global rule - thus also tainting the validity of the simulation.

A solution to this dilemma may require a rethinking of Langton's principles and acceptance of universal complexity, or existance of complexity at every level of study - individual, group, orgniazational and systemic.

Figure 2.3 illustrates the relationship of existing methodologies for modeling of social systems in regards to their place in the spectrum of individual and system-level complexity.

Axtell and Epstein[Axtell and Epstein, 1994], have summarized the levels of agent-based model performance and analysis as follows:

Level 0: the model is a caricature of reality, as established with simple graphical devices (e.g. allowing visualization of agent motion).

Level 1: the model is in qualitative agreement with empirical macro-structure, as established by plotting the distributions of some attributes of the agent population.

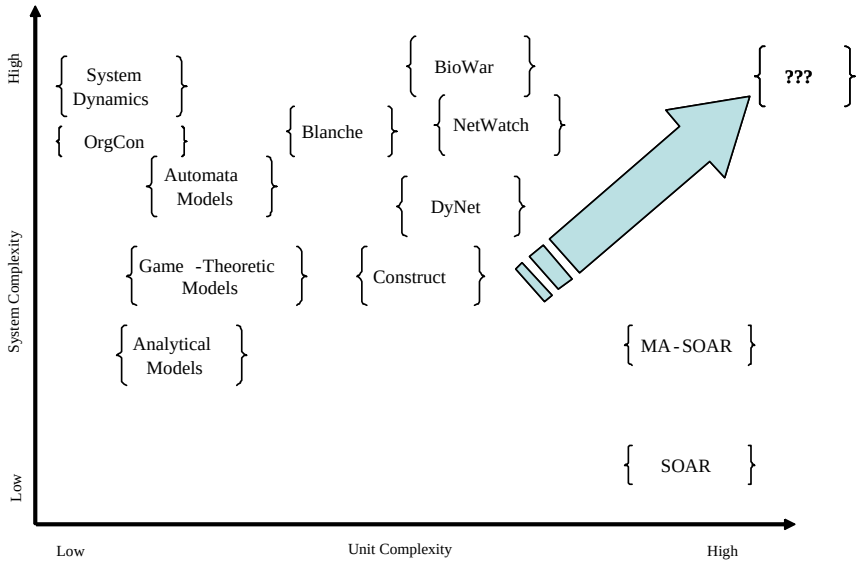


Figure 2.3: Spectrum of Complexity in Agent-Based Simulation

Level 2: the model produces quantitative agreement with empirical macrostructure, as established through statistical estimation routines.

Level 3: the model exhibits quantitative agreement with empirical microstructures, as determined from cross-sectional and longitudinal analysis of the agent population.

Until mid-1990s it was considered impossible to create symbolic reasoning systems such as SOAR that could operate in packs - i.e. collections of *complex* agents in a *complex* world - as even one of such agents required immense amounts of computational power. The advent of cheap and plentiful computation allows modelers of social systems to accept such multi-level complexity, and build models that do not sacrifice individual-level veridicality for system-level accuracy.

I posit that the creation of high-fidelity models of socio-technical systems requires the combination of analytical models with empirically grounded simulation.

In the next chapter, I present NetWatch, a multi-agent model designed to combine high individual-level complexity and individual-level strategic reasoning with high system-level and organization-level complexity through the use of social network analysis and artificial intelligence techniques.

As table 2.2 shows, NetWatch combines the advantages of simulations that accentuate fidelity of an individual agent with advantages of simulations that accentuate fidelity of the agent's environment and networks.

While NetWatch is highly dependent on network topologies, it does not require a preset or constant network topology. In fact, in NetWatch the

topology is viewed as a result of cumulative agent interactions - and thus evolves continuously. This sets NetWatch apart from other multi-agent simulation tools where the network topology is constant, pre-set as a parameter, or a part of design assumptions. Such fluid network configuration allows us to model emergence of novel network topologies in response to stress or task pressures, as well as study self-healing behaviours in dynamic networks. I demonstrate such emergent behaviours in NetWatch in sections3.12 and 4.13.

NetWatch combines the ability to simulate complex, large organizational networks with the ability to realistically model individual agents and their interactions. Such realism is rooted in the use of artificial intelligence techniques and robust knowledge representation - as a backbone of every agent. Using realistic (or empirically derived) specifications of complex tasks, agents plan their actions according to their goals and their resource and information requirements.

Literature shows[Krackhardt and Hanson, 1993] that informal networks are often more important and more effective then formal organizational structures. NetWatch models emergence of such informal networks via non-task-related social interactions between agents.

Many simulations that accentuate the fidelity of individual agents are computationally expensive and thus limited to a small number of agents in simple organizational structures. While each of the agents in NetWatch is autonomous and intelligent, the agents are implemented in a highly efficient manner - which allows users to set up large-scale simulations without sacrificing agent fidelity.

		NetWatch	Construct/DyNet	Multi-SOAR	Blanche	Cellular Automata
System Properties	Number of Agents	10-2000	10-5000	1-10	10-100	10-1000000
	Network topology	Arbitrary networks, including empirical	Implicit from interactions	Simple pre-set topology	Arbitrary pre-set networks	Pre-set grid topology
	Networks evolve	Yes (based on interactions)	Yes (based on interactions)	No (all-to-all connectivity assumed)	No (pre-set topology is constant)	Yes (if agents can move)
Agent Properties	Autonomy	Autonomous, asynchronous agents	Central control	Autonomous, asynchronous agents	Central control	Central control
	Knowledge	Local knowledge of self, others and environment, learning from experience	Global knowledge in the environment	Local knowledge of self and environment	Global numerical state	Local numerical state, perception of environment
	Intelligence	Strategic Reasoning, Planning, Behaviour switching	Behaviour switching	Strategic Reasoning, Planning	Computation, state switching	State switching
	Interaction	Direct interaction with other agents and environment	Simulated interaction with other agents	Direct interaction with other agents and environment	Simulated interaction with other agents and environment	Direct interaction with other agents

Table 2.2: Comparison of Techniques for Social Simulation

Chapter 3

Technical Description of NetWatch

3.1 Modeling Dynamic Networks

Based on the conceptual framework of multi-agent simulations, we have developed NetWatch, a multi-agent network model for reasoning about the destabilization of covert networks such as organized crime or terrorist organizations under conditions of uncertainty.

NetWatch is built to simulate the communication patterns, information and resource flows in a dynamic organizational network based on cognitive, technological and task based principles. In addition, the model is grounded using information about surveillance technologies and intelligence operations (e.g. [Alberts, Garstka, and Stein, 1999]) and the covert networks (e.g. [Berry, 2001]).

The process of gathering intelligence on an organization is simulated enabling the evaluation of diverse heuristics and technologies for data gathering. Using NetWatch, the user can conduct a vulnerability analysis and examine potential emergent reactions of covert networks in response to attacks as well as evaluate diverse destabilization strategies.

In this chapter, I introduce NetWatch and both technical and theoretical foundations that underlie its construction.

3.2 NetWatch Simulation of Covert Networks

The design of NetWatch simulation of covert networks and anti-terrorist activity is based on the concept of *red teaming*, a war-gaming approach in which a population of participants is divided into two or more adversarial teams. The teams then are empowered to use any techniques at their disposal to achieve their goals. The main objective of red-teaming is learning the mind-set and *modus operandi* of the adversary and development and testing of strategies fielded against said adversary. Real-world red-teaming is most common in the fields of military training, war-gaming and computer security and has recently received renewed attention as a useful counter-terrorist tool.

It is postulated that the true value of red teaming emerges with capture, documenting, and application of insights gained across individual events and

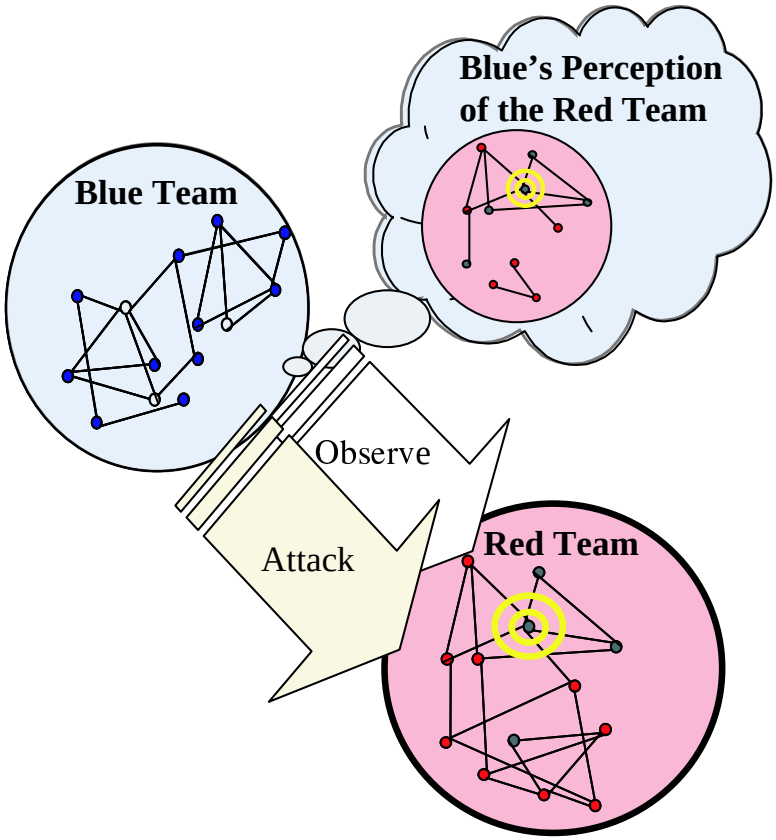


Figure 3.1: NetWatch Simulation Design

studies. A *comprehensive method* of red-teaming[Mateski, 2003] has been described as

> ...the ability to explore the conceptual space of all potential terrorist attacks -quickly, systematically, and thoroughly. Given the immediacy of the threat, the method employed should be simple, flexible, reusable, and scalable.

NetWatch is designed to follow the comprehensive method of red-team modeling by using high-fidelity representations of both the covert network and the entities responsible for prevention of terrorist activity.

For the purposes of simulation, the covert network is designated as the **Red Team** and the network of anti-terrorism forces as the **Blue Team** (see figure 3.1). Both of the teams consist of a number of autonomous, intelligent agents, designed to simulate with highest possible fidelity the activity of individuals and groups present in the subject networks.

The agents of the Red Team are intelligent, knowledge-driven planning agents that model the process of execution of a logistically complex terrorist attack - complete with gathering required resources, obtaining knowledge

and training, and tactical planning. Section 3.5 describes in detail the design and construction of Red Team agents including design of agent communication language (sec. 3.6), design of communication protocols for knowledge exchange and resource transfer (sec. 3.8). Task planning and execution by Red Team agents are described in section 3.9.

Red Team network is modelled upon organizational structure of a terrorist organization and constructed to fit a statistical profile of such an organization. The statistical profile mechanism (described in chapter 6) allows for manipulation of both social network topologies and distribution of information and resources, which leads to robust capabilities for testing of theories of organizational design in covert networks. Section 3.4 describes the construction of social networks within the Red Team and evolution of the networks due to group formation, homophyly and operational needs is described in section 3.7.

The Blue Team (described in section 4) represents a set of agencies engaging in anti-terrorist activity and pursues two interconnected goals. The blue team conducts signal intelligence-based information gathering and uses collected information to build a MetaMatrix representation (see sec. 2.6) of the Red Team. I describe multiple signal intelligence strategies in section 4.0.2. Needless to say, signal intelligence does not produce complete information on communication patterns and Blue Team agents must deal adequately with the uncertainty introduced by such incomplete information. A number of algorithms for improving the performance of signal intelligence strategies is described in section 4.4.

The second goal of the Blue Team is to destabilize or decrease performance of the Red Team. The Red Team performance metrics are described in section 3.10 and various destabilization strategies are outlined in section 4.12

The organizational structure of the Blue Team mimics the information sharing regimes defined by intelligence and law enforcement agencies. In such information sharing regimes, the agencies can divulge limited information to their peers but the rules for what information can be shared are defined by the agency's charter and rules of engagement. Construction of the Blue Team network and implementation of information sharing policies is described in section 4.11.

To conduct large-scale experiments within the NetWatch framework, I have designed the Observer Agent - a modular system for instrumentation of virtual experiments. The system uses a combination of communication traffic analysis and instrumentation taps within each agent to build a coherent "bird's-eye view" of the entire simulation, and compute a number of statistical metrics using this knowledge. The Observer Agent is described in detail in section 3.11.

3.3 Agent-based Modeling of Dynamic Networks

NetWatch agents are intelligent adaptive information processing systems, constrained and enabled by the networks in which they are embedded. These networks evolve as individuals interact, learn and perform tasks. The design of the NetWatch multi-agent model is based on the principles of agent-based models of complex adaptive systems outlined by Langton[Langton, 1989] (also see section 2.7 for a detailed discussion).

To reiterate, Langton's principles of agent-based modeling are:

1. The model consists of a population of simple agents.
2. There is no single agent that directs all of the other agents.
3. Each agent details the way in which a simple entity reacts to local situations in its environment, including encounters with other agents.
4. There is no rule in the system hat dictates global behaviours.
5. Any behaviour at levels higher then individual agents is therefore emergent.

However, I make an important distinction from Langton's ABM techniques. In NetWatch and related models, agents are not defined as simplistic automata following a small set of deterministic rules. Instead, an agent can be viewed as a representation of a human actor involved in the the simulated activities. Using artificial intelligence techniques, the agents can plan and reason about task completion and formation of their social networks and make strategic moves to maximize their utility.

In effect, each agent within NetWatch is built in the same manner as an autonomous robot (sans the hardware) designed to survive on its own in a hostile environment. In greater detail, the methodology of multi-agent network modeling is based on the following principles:

- Agents are independent, autonomous entities endowed with some intelligence, though cognitively limited and boundedly rational. Agents can utilize both deterministic or stochastic rules.
- Agents and the networks in which they are embedded co-evolve. While the initial topology of agent network can be used as an independent variable, the community of agents will create a very different topology at the end of a simulation.
- Agents do not have accurate information about the world or other agents and are limited by their perception.

- Agents can learn the state of the world through interaction. Note that while agents do not have access to a global world-view, they can learn about their non-immediate neighbors through communication and collaboration with other agents.

- Agents can be strategic about their communication. They can use rule-based, decision-theoretic, optimization or other techniques to maximize their utility.

- Agents do not use predefined geometrical locations or neighborhoods. Instead, their choice of communication partners depends on the topology of their social network and evolves over time.

Note the importance of networks in the design of the simulation. Agent-based models comprised of cellular-automata frequently overlook the fact network topology is as important to the fidelity of a model as the rules governing the behaviours of individual agents. The topology of a flat or toroid grid with communication through Manhattan or Minsky neighborhoods[Minsky, 1967], while providing a useful test-bed for cellular automata models, is not grounded in social theory and thus is not very suitable for modeling complex social systems.

3.4 Social Networks in NetWatch

In NetWatch, agents communicate on the basis of network membership. Agents learn of other agents outside their ego network via interaction with agents in their ego network and a process of introduction. Therefore, networks are represented as a directed graph structure representing the probability of communication (social proximity) p_{ij} between two agents a_i and a_j:

$$\begin{aligned} Net &= AG, P \\ AG &= a_i : Set\ of\ agents \\ P &= p_{ij} : \forall a_i, a_j \in AG \end{aligned} \tag{3.1}$$

The directed nature of the graph Net allows the user to specify one-way relationships and chain-of-command relationships. While the formal network is generally pre-specified at the start of the simulation, the informal network evolves through interaction.

The agents do not have access to full information about the network, but every agent a_k can only access a probability vector $P_k = p_{ki}$ where p_{ki} is a probability of agent a_k communicating with all agents $a_i \in A$. Hence, each agent may only know who it may interact with or is close to, but does not

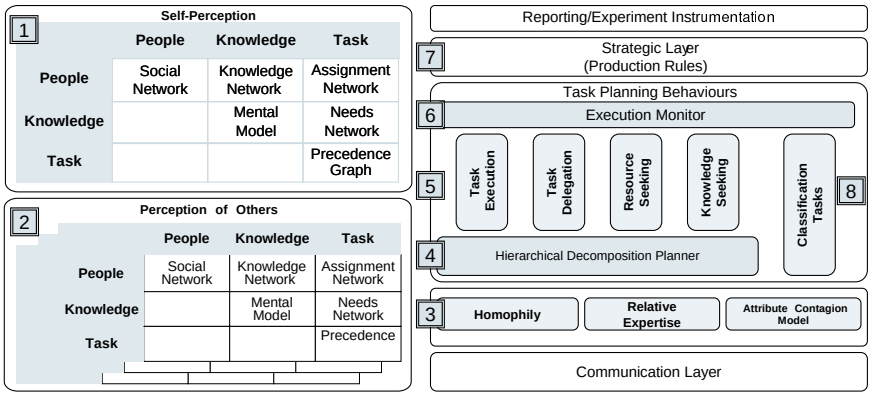


Figure 3.2: NetWatch Agent Architecture

know the complete interaction patterns of other agents. Each agent possesses a belief matrix that it uses to store any information it learns about interrelationships of other agents within the network. However, this information is typically incomplete and inaccurate.

Agents obtain information via interaction with other agents. The accuracy of an agent's perception of another decreases with the distance between them in the social network. This corresponds with the empirical reality where people's knowledge of each other decreases exponentially as the social distance between them [Krackhardt, 1990] increases.

3.5 Agents in NetWatch

In keeping with cognitive science research, NetWatch agents representing humans are both cognitively and socially constrained [Simon, 1955]. Thus, their decision-making ability, actions, and performance depend on their knowledge, structural position, procedures and abilities to manage and traverse these networks. Each agent's perception of the meta-matrix consists of the agent's ego network (the set of agents it is directly connected to), its own knowledge, resources and task assignments. It is augmented by the agent's perception of other agents' ego networks, knowledge, resources and task assignments (Fig. 3.2, 1). An agent also tries to form beliefs about the networks of other agents (Fig. 3.2, 2).

The agents implement a layered behavior model inspired by Rodney Brooks' *subsumption architecture* for robot control [Brooks, 1985]. On the lower levels of control lie primitive communication behaviors (Fig. 3.2, 3) ,

based on cognitive models of human communication.

Intelligent task-directed behaviors are facilitated by a hierarchical decomposition planner (Fig. 3.2, 4), adapted for goal-directed interaction and distributed task execution via delegation of subtasks. The planner is described in section 3.10.1.

Execution of distributed tasks is monitored by an independent execution monitoring process (Fig. 3.2, 6), which watches results of delegated tasks, handles exceptions and triggers replanning in case of failure.

Finally, production rules (Fig. 3.2, 7) governs the agent's strategic reasoning, triggering tasks or high-level behaviors.

Such a layered architecture allows a social modeler to isolate strategic and tactical performance of the agents from their lower-level interaction and build experiments where any one of the levels is manipulated.

3.6 Agent Communication

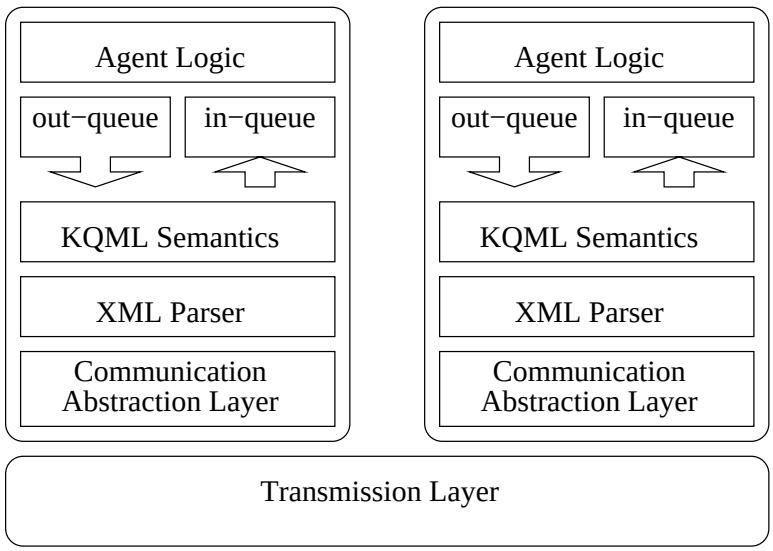


Figure 3.3: Communication Architecture of NetWatch Agents

Agent communication in NetWatch is designed as a layered construct (see figure 3.3). On the lowest level, the agents use a communication abstraction layer that isolates the internal agent logic from the details of actual network or shared memory communication. It also implements agent name lookup and provides robust message sending and receiving functions. A simple notion of time-outs helps prevent deadlocks between communicating processes.

After being received by the communications abstraction layer, the message is parsed using a SAX XML parser[XML.org, c]. This provides for both

rigorous grammar checking and simplicity in the syntax and parsing of the agent communication language.

3.6.1 Agent Communication Language (ACL) in NetWatch

NetWatch implements a simple ACL based on the simplified semantics of KQML[Finin, Fritzson, McKay, and McEntire, 1994]. Each mesage is constructed as follows:

$$\begin{aligned} message &:= \{performative(string), (NameValuePair)*\} \\ NameValuePair &:= \{name(string), value\} \\ value &:= [number|string|message] \end{aligned}$$

KQML was conceived as both a message format and a messagehandling protocol to support run time knowledge sharing among agents. KQML messages are opaque to the content they carry. KQML messages do not merely communicate pieces of information but communicate an attitude about the content (e.g. assertion, request, query).

The language's primitives are called performatives. As the term suggests, the concept is related to speech act theory. Performatives denote permissible actions that agents may attempt in communicating with each other. In NetWatch ACL, only a limited number of performatives are supported based on the design of the knowledge exchange and task execution protocols. A list of performatives and their descriptions can be found in table 3.1.

A performative is followed by a set of name-value pairs that contain the actual information transmitted in the message. A value of such name-value pair can be numeric, string or contain a nested message (which is similar to passing a C-language *struct*).

A canonical form for a NetWatch message is following:

```
<performative
  sender="name-of-sender-agent"
  receiver="name-of-receiving-agent">
  <content>
    <content_performative [content-specific fields]/>
  </content>
</performative>
```

3.7 Formation of Homophyly Groups via Communication

The first agent behaviour to be discussed is that of Idle Chatter. Despite the unassuming name, Chatter is one of the most important steps towards

Chatter	*say,ask*	used by agents in the idle state and represents social-psychological concepts of homophily-based group creation (see section 3.7)
Knowledge Transfer	*knowledge-query*, *knowlege-transmit*	used in the knowledge-seeking behavior.
Resource Transfer	*resource-query*, *resource-transmit*	used in the knowledge-seeking behavior.
Classification Task	*classification-task*, *classification-task-result*	used to transmit classification task assignments and results
Planning Task, Delegation	*task*, *task-result*	used by agents to transmit planning task assignments and results. **Note:** format for a delegated task is the same as the format for system-assigned task
Reporting and Instrumentation	*attribute-report,stats-report*	facilitate experiment instrumentation and logging through agent self-reporting of state

Table 3.1: Design of NetWatch ACL: Message Types and Performatives

forming of groups in the agent population, thus propagating information and facilitating task performance.

Agents engage in Chatter at their leisure - i.e. when they are not occupied with planning tasks and between regularly arriving classification tasks (described in sections 3.9 and 3.10.1, respectively). When the Chatter behaviour is enabled, an agent chooses a partner to communicate with and engages in a round of conversation. In this conversation, information is exchanged and strength of network ties between the two agents increased.

The choice of a communication partner at every time period is based on two factors: *social proximity* of the agents and their *motivation to communicate*. Social proximity is defined as closeness of a relationship between two agents, scaled between 0 and 1 where 0 means "no relationship" and 1 is "very close relationship".

Motivation to communicate is computed on the basis of *homophyly* (relative similarity) and *need* (relative expertise). Empirical studies of human communication behavior suggest that, without any external motivation, individuals will spend about 60% of the time interacting on the basis of homophyly and 40% on the basis of need.

We defined homophyly to be based on a measure of relative similarity RS between agent i and agent j: the amount of knowledge that i and j have in common divided by the amount i shares with all other agents (including

self), or

$$RS_{i,j} = \frac{\sum_{k=0}^{K}(S_{ik} * S_{jk})}{\sum_{l=0}^{I}\sum_{k=0}^{K}(S_{ik} * S_{lk})} \tag{3.2}$$

where S_{ik} is 1 if agent i knows fact k and 0 otherwise.

In contrast, we defined need from a purely knowledge perspective. Relative expertise RE_{ij} defined as how much agent i thinks j knows that i does not know divided by how much i thinks all others know that i does not know, or

$$RE_{ij} = \frac{\sum_{k=0}^{K}((1 - S_{ik}) * S_{jk})}{\sum_{l=0}^{I}\sum_{k=0}^{K}((1 - S_{ik}) * S_{lk})} \tag{3.3}$$

Agents operate on their beliefs about what the other agents know. Thus, their calculations can be inaccurate. However, as interaction progresses and agents learn more and more about each other, the accuracy of the agents' perception of the world increases.

3.8 Inter-agent Knowledge Exchange

Tracing back its roots with the Construct [Carley, 1999] model, the NetWatch model is a cognitive model focusing on knowledge manipulation and learning. Each agent's knowledge is represented by a bit string. A value of 1 in the position n means that the agent knows fact n and the value of 0 means that it does not.

Both homophyly and need for information, as used in the knowledge exchange protocol, are abstract measures and do not weigh facts in regards to their importance to a task. More complex behaviors, such as task-directed information seeking, is accomplished using the planner (see section 3.9).

At the start of the simulation, the agents are endowed with some initial knowledge (typically within 2%-10% range), distributed randomly between agents or based on empirical profile of the organization.

To learn new facts, the agents execute the **Construct Knowledge Exchange Protocol**. For ease of description, we shall refer to the parties in knowledge exchange as Alice (agent $a_a \in AG$) and Bob (agent $a_b \in AG$).

1. **Determine who to communicate with:** Alice does this by evaluating *relative similarity* (Eqn. 3.2) or relative expertise (Eqn. 3.3) of every agent accessible through its social network (i.e. $p_{a,i} > 0 \; \forall a_i \in AG$ (Eqn. 3.1)). Then, a_a throws a dice that reflects the computed probability vector and picks an agent to communicate with (e.g. a_b).

2. **Determine what to communicate:** This is done by weighing information seeking vs. similarity-driven communication. If a_a is in information seeking mode, it chooses at random a part of the knowledge string that is not known and queries the agent chosen in step 1. In similarity-based communication, a_a chooses a part of the known knowledge string and sends it to that agent. Two example queries follow:

```
<ask sender="Alice" receiver="Bob">
 <content>
   <knowledge index="12"/>
 </content>
</ask>

<say sender="Alice" receiver="Bob">
 <content>
   <knowledge index="12" value="128"/>
 </content>
</ask>
```

3. **Determine proper response:** On receipt of a query, a_b determines if it should answer by checking whether the sender of the query is part of its network. If yes, a_b sends a reply - otherwise, he discards the message. If a_b does not know the facts requested, he may respond to a_a with a name of another agent ("Clare") that may be better suited to answer the question, known as referential data. On receipt of knowledge, a_b determines if the knowledge is useful and whether it came from one of the agents in its network (and thus can be trusted), chooses some knowledge from its knowledge base and send it in return. Two example replies follow:

```
<!-- If Bob knows the answer to Alices question -->
<say sender="Bob" receiver="Alice">
 <content>
   <knowledge index="12" value="253"/>
 </content>
</say>

<!-- If Bob does not know the answer -->
<!-- but can refer to another agent -->
<say sender="Bob" receiver="Alice">
 <content>
   <knowledge index="12" value="unknown"/>
   <refer-to receiver="Clare"/>
 </content>
</say>
```

4. **Update internal knowledge base:** On receipt of the reply, a_a determines the usefulness of the answer and uses that to update its internal

knowledge of a_b ("Bob knows fact n") as well as its knowledge base ("I now also know fact n").

If a_a received referential data, found it useful and it came from one of the agents in its network (and thus can be trusted), a_a chooses some knowledge from its knowledge base and sends it in return. He then uses the referential data to update its knowledge of a_b ("Bob does not know n" and "Bob knows Clare") and a_c ("Clare may know n"). This may be followed by a query to Clare (a_c), which may or may not be answered, depending on the strategic position of a_c.

Clare may not have been originally a part of Alice's network - but now through Bob, Alice has learned about her existence. Thus, agents within the organization use referential data about each other to form an informal network.

3.9 Planning and Execution of Complex Tasks

The design of task structures in NetWatch is based on the premise that organizations are fundamentally information-processing entities. In this view, voiced in Max Weber's work in the early 1900s and elaborated by March and Simon[March, 1988][March and Simon, 1958], an organization is an information-processing and communication system structured to achieve a specific set of tasks and comprised of limited individual information processors.

In such organizations, tasks are described as sequences of interdependent communications and actions[Thomson, 1967], meaning that the output of a given task is the input for a succeeding tasks. Such tasks are distributed across the individuals within the organization according to the organizational structure. Thus, the subtasks can be represented in a precedence network.

This view of the organization coincides with the work in the Artificial Intelligence community on creating algorithmic representations of planning tasks. In a 1977 paper, A. Tate[Tate, 1977] describes the organizational task completion as being composed of two stages:

1. The consitutent "jobs" of a plan are specified together with their precedence relationships ... This information defines a graph, termed a *project network*.
2. Various operations are performed on the project network to establish schedules and allocate resources.

The project network thusly generated can be used not only for predictions of how the project will be done but also as a tool to aid in monitoring the

progress of tasks and identification of bottlenecks. However, in the Operations Research literature, as well as in recent organizational modeling tools (e.g. VDT[Levitt, Cohen, Kunz, Nass, Christiansen, and Jin, 1994], MAGNET[Collins, Tsvetovat, Mobasher, and Gini, 1998]), the task structures are manually constructed, at a considerable effort.

A formalism referred to as a Hierarchical Task Network (*HTN* has been developed in the AI planning community[Erol, Hendler, and Nau, 1994][Young, Pollack, and Moore, 1994] to encapsulate the process of automatic decomposition of hierarchical task-subtask structures into an executable precedence networks.

A task network in HTN representation is a collection of tasks that need to be carried out, together with constraints on the order in which the tasks can be performed and resources required. A task network that contains only *primitive tasks* - tasks which can be immediately carried out if their resource constraints are met -is called a *primitive network*. Such a network might occur, for example, in a scheduling problem.

In the more general case, a task network can contain *non-primitive tasks*, which cannot be executed directly because they represent activities that may involve performing several other tasks.

HTN Planning works by expanding non-primitive tasks and resolving conflicts iteratively until a conflict-free plan consisting of primitive tasks only can be found. Erol,Hendler and Nau generalize the algorithm of HTN planning[Erol, Hendler, and Nau, 1994] as:

```
1. Input a planning problem $P$
2. IF $P$ contains only primitive tasks,
 THEN Resolve the conflicts in $P$ and return RESULT
 ELSE if the conflicts can be resolved, return FAILURE

3. Choose a non-primitive task $t$ in $P$
4. Choose an expansion for $t$
5. Replace $t$ with the expansion
6. Check plan for constraint satisfaction and conflicts
7. Resolve the conflicts that can be resolved immediately
8. Go to 2
```

In NetWatch, agents follow the generalized algorithm above. However, the planning problem is further complicated by the fact that planning is done in distributed fashion and none of the agents involved in task completion have access to the complete HTN specification. Furthermore, agents may not have access to the knowledge and resources required for accomplishing even the primitive tasks. All of the above requirements must be satisfied purely by interaction with other agents.

In the meta-matrix representation, the agents possess a definition of their hierarchical task structures — an acyclic directed graph specifying the precedence of tasks. The skill requirements and the resource requirements are also

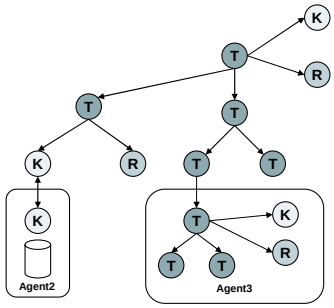


Figure 3.4: Planning in NetWatch Agents

specified in the meta-matrix.

The planner operates with five basic behaviours:

- **EXPAND**: Find subtasks of a given non-primitive task
- **EXECUTE**: Executes a subtask if its knowledge and resource constraints are satisfied. Return RESULT or FAILURE
- **KNOWLEDGE-SEEKING**: Sends series of queries to agents requesting information that would satisfy a knowledge constraint of a primitive task
- **RESOURCE-SEEKING**: Sends series of queries to agents requesting information that would satisfy a resource constraint of a primitive task. There is a cost associated with transferring resources
- **DELEGATION**: If an agent lacks sufficient information plan a non-primitive task expansion, it will attempt to delegate the planning and execution of the task to other agents.

The planner starts with a top-level task and performs an expansion by finding its subtasks, resource and knowledge constraints. If one of the subtasks requires knowledge or resource that the agent doesn't possess, the agent sends out request messages. The requests are handled by the Knowledge Exchange protocol (see section 3.8).

Similarly, if the agent does not have the ability to execute a task, a task delegation message would be sent to an agent determined likely to execute the task.

3.9.1 Execution Monitoring

In traditional multi-agent planning systems, an assumption is made that agents are cooperative[Wagner, 2000]. This assumption cannot be true in simulation of adversarial networks. Thus, failure of delegated subtasks or

Failure Mode	Action
Subtask Failure (not executable)	Record failure (decrease capabilities of delegee) and re-plan subtask
Subtask Failure (rejection)	Record rejection (decrease relationship strength to the delegee) and delegate to another agent
Knowledge or Resource Request Failure (not available)	Record failure and attempt another request
Knowledge of Resource Request Failure (rejection)	Record rejection (decrease relationship strength to the delegee) and query another agent
Communication Timeout (no response)	Query another agent or re-plan
Number of retries exceeded	Return failure, abort non-primitive task

Table 3.2: Reactions of Execution Monitor to Failure Modes

knowledge requests is not only possible, but expected. Delegated operations can fail due to lack of appropriate information at the receiver or level of busyness of the receiver. Alternatively, delegated operations can be strategically rejected by the receiver due to its internal constraints and rules[Wooldridge].

NetWatch addresses this problem via the use of an Execution Monitor process within each agent. Execution Monitor consists of a stack which contains delegated subtasks and requests and a set of rules covering the instances of subtask or request failures, timeout conditions and strategic rejections, as shown in table 3.2.

Performance in planning tasks is measured as (a) time that it takes, (b) amount of re-work or replanning needed to complete the task, and (c) percentage of tasks that have failed.

3.9.2 Task Triggering Mechanisms - When and Why Do the Agents Plan and Execute Tasks

The planner, in conjunction with the task specifications, and data structures accumulating knowledge and resources, govern **how** the agents execute their tasks. However, the questions of **when** and **why** to start planning and executing a task remain unanswered.

In the Autonomous Agnets/Artificial Intelligence communities it is assumed that agents need to only be given a capability to execute a task and a human user will command the agent to start. This condition is not realistic in the world of agent-based simulations, as the human user is absent from the loop. Moreover, the goal of the task itself is not to accomplish a concrete piece of work (i.e. retrieving information, negotiating a deal) but to model the way the work is performed in the real world.

Listing 3.1: "Resource-based Task Triggering"

```
FOR Every time period DO
 Run CHATTER to obtain new information and resources
 FOR every non-primitive task in task graph DO
   Evaluate feasibility of task, based on given knowledge and
      resources
   IF a task is feasible
     TRIGGER TASK EXECUTION
   END IF
 END FOR
END FOR
```

Periodic Task Triggering

The simplest strategy for triggering of planning-based tasks is to allow some agents (e.g. cell leaders) to trigger them periodically with period between tasks controlled as a parameter of the simulation. For example, it is known that Al Qaeda in the past has attempted a major terrorist attack approximately every two years.

It is then possible to vary the frequency of incoming complex tasks to test the robustness of the organizational network in response to increased workload and cognitive demand on leader agents (who bear most of the planning burden in their respective cells)

However, periodic task triggering is not a realistic way to model the functioning of an organization. The periodicity of attacks by Al Qaeda is an effect of the political climate, resource constraints, and morale of members - rather then a pre-set parameter.

Periodic task triggering is implemented in NetWatch as a baseline test strategy, as well as a way to hold a complex task planning mechanism as a constant when other variables are being evaluated.

Resource-driven Task Triggering

Resource-driven task triggering is an opportunistic strategy based on knowledge and resource exchange behaviours that are a part of the **CHATTER** protocol and follows the algorithm in listing 3.1.

While resource-driven task triggering is more realistic given the context of the simulation, it assumes that agents are purely opportunistic and do not have endogenous or exogenous factors (morale, doctrine, political situation) contributing to the decision to start the task.

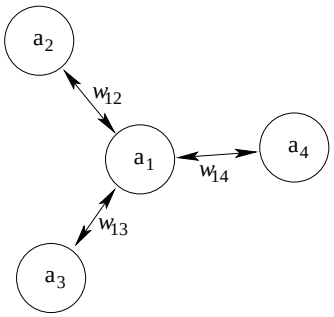


Figure 3.5: Attribute Contagion in Agent Networks

Attribute-driven Task Triggering

A multitude of theories on causes of terrorism have been proposed linking political situation, poverty, religious extremism and other factors with an individual's decision to become a terrorist. Many of these theories can be generalized in a following framework:

$$T = \sum (a_i * y_i)$$

where T is a tension function, a_i is a quantifiable attribute of an agent, and y_i is a weight or relative importance of attribute a_i. The weights can be derived from analysis of empirical data through regression-based methods. The attributes dynamically change as follows:

$$a_i^{(t+1)} = f(a_i^{(t)}, x)$$

where $a_i(t+1)$ is a value of the attribute at time $t+1$, computed as a function f of the previous value of attribute and x - a set of exogenous factors.

For example of a specific case of attribute change, let us consider an attribute contagion model. Social influence network theory[Friedkin, 1998] describes an attribute contagion process in a group of N persons in which members' attitudes and opinions (expressed by attribute values) change as they revise their positions by taking weighted averages of the influential positions of other members (see figure 3.5):

$$a_i^{(t+1)} = \alpha_i(w_{i1}a_1^{(t)} + w_{i2}a_2^{(t)} + w_{iN}a_N^{(t)}) + (1 - \alpha_i)a_i^{(t)}$$

for each of the N persons in the group ($i = 1, 2, ..., N$). The attribute values of persons at time t are $a_1^{(t)}, a_2^{(t)}, ..., a_N^{(t)}$. The set of influences of the group members on person i is $\{w_{i1}, w_{i2}, ..., w_{iN}\}$, where $0 \leq w_{ij} \leq 1$, and $\sum_j w_{ij} = 1$. The susceptibility of a person i to the influence of others is a_i, where $0 \leq a_i \leq 1$.

In view of this analytical model, a task triggering algorithm would be quite simple:

```
FOR each time period t DO
 FOR every attribute a(i) DO
   compute new value of a(i)(t) based on the old value a(i)(t
       -1), internal and external factors
 END FOR

 Compute T(t): the current tension function
 IF T(t) > tension threshold DO
   TRIGGER TASK EXECUTION
 END IF
END FOR
```

While the attribute-based task triggering model presents possibly the most realistic view of when and why tasks may be triggered within a terrorist network, its accuracy depends on a number of parameters which may be difficult to estimate.

First of all, it is important to determine which attributes of the agent should be a part of the tension function, and what their relative weights should be. This can be done by building regression models of empirical data; however, in the context of terrorist organizations, a large number of such studies would make choosing a realistic set of attributes difficult.

In light of this observation, one can view NetWatch's task triggering mechanism as a way to test multiple empirically derived theories and compare their performance in simulation to real-world results.

A further difficulty arises from the need to set a tension threshold. There is virtually no empirical data that would provide a methodological way to reach a firm value for the threshold; thus it will remain a parameter of the simulation to be tested using parameter space exploration rather then empirical grounding.

3.10 Classification Tasks

Another measure of organizational performance in NetWatch is based on the binary classification task, a general representation of organizational performance involving elements of pattern matching and determination of statistical relationships.

The classification task is a generalized version of the RADAR Task[Ye and Carley, 1995]. In the RADAR task, the organization is presented with bit patterns representing a radar signature of an aircraft (such as position, altitude, speed, friend-or-foe identification, etc). The goal of the organization is to analyze the radar signature to determine whether the incoming aircraft is friendly, neutral or hostile.

The task is represented by a vector of binary values supplied by a task generator (described in 6.0.4) and an outcome. The goal for each agent is to

Listing 3.2: "Execution of Classification Task"

```
//Iterate through all subtask in the complex task
FOR EACH subtask node t(i) DO
 P(i)=0; k=0;
 //Find all edges between this subtask and knowledge required
 //to accomplish the subtask
 FOR EACH e = edge betweeen t(i) and the knowledge subgraph DO
   k=k+1;
   IF e.target!=0
   //If the agent knows the required fact
     THEN P(i)=P(i)+1; //increment the probability factor
   END IF
 END FOR

 //Probability of full access to the subtask is the ratio of
     the knowledge
 //that the agent already has and the knowledge required for
     the subtask
 Prob(i)=P(i)/k;
END FOR
```

determine which configuration of 1's and 0's in the vector represents which value of the outcome (friendly/neutral/hostile).

Task complexity[Carley, 1992b] is defined as the length of the task vector (N), or number of subtasks in a non-primitive task. For a given level of task complexity there are potentially 2^n possible configurations. Thus, as task complexity increases the likelihood of encountering identical task configuration in two tie periods decreases exponentially.

Furthermore, each agent can only access segments of the task vector that are determined by its knowledge string. The probability of access to elements of task vector is calculated using the algorithm in listing 3.2

The actual access to the subtasks in the vector is determined with a roll of uniformly distributed dice. The task is then decided by a "majority rule"". Each agent's decision accuracy is computed by taking a series of classification tasks and comparing the agent's decisions to "true answers" - computed by applying a majority rule to each task given complete information and access.

Task complexity[Carley, 1992a] is defined as the length of the task vector (N). For a given level of task complexity there are potentially 2^n possible configurations. Thus, as task complexity increases, the likelihood of encountering identical task configuration in two tie periods decreases exponentially.

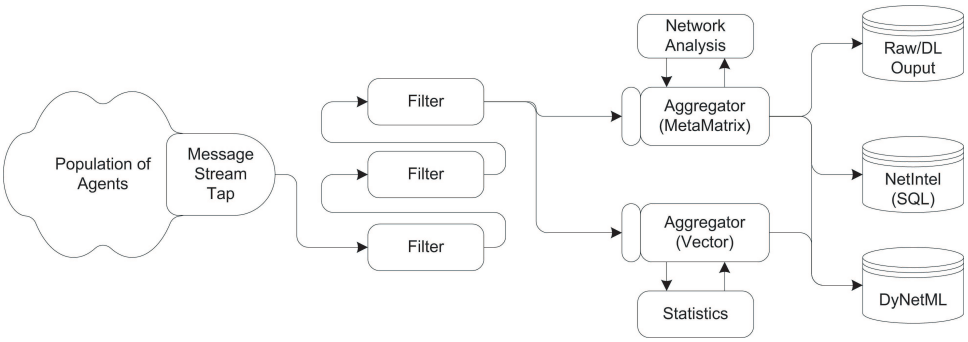


Figure 3.6: Experiment Instrumentation: Observer Agent

3.10.1 Performance Measurement

An agent can only access bits in the task vector that correspond to non-zero values in the its knowledge vector. The task is then decided by a "majority rule". An agent's decision accuracy is computed by taking a series of classification tasks and comparing the agent's decisions to "true answers" - computed by applying a majority rule to each task given complete information and access. Task performance is measured as a percentage of correctly decided tasks.

Overall performance of the organization is measured as the percentage of correct decisions made by the organization[Ye and Carley, 1995][Lin and Carley, 1997]. A second measure is the severity of the error. An organization can make an "off-by-one" error (e.g. classifying a neutral aircraft as friendly) or an "off-by-two" error (e.g. classifying a hostile aircraft as friendly). The severity of the error is defined as percentage of total errors that are severe ("off-by-two").

While appearing simplistic, performance in classification tasks have been shown [Lin and Carley, 2003] to correspond to organizational performance in real cases, thus making classification tasks a suitable substitute for more complex tasks for purposes of simulation modelling.

3.11 Observer Agent: Instrumentation for Virtual Experiments

Conducting useful experiments using a simulation as complex as NetWatch requires a fairly complex system for experiment instrumentation and observation.

In NetWatch, the instrumentation system is designed in a manner similar to IRIS Explorer[Numerical Algorithms Group, 2004] and other scientific visulialization applications. It also provides full integration with the CASOS

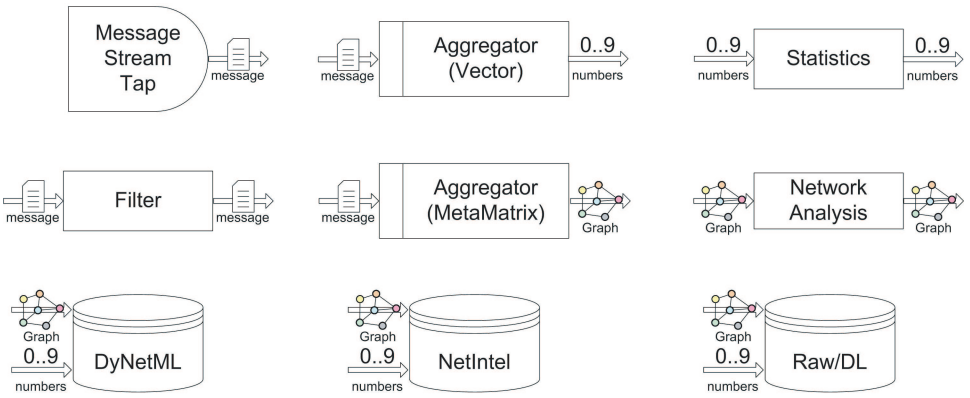


Figure 3.7: Instrumentation Modules

toolchain, described in chapter 7.

The Observer Agent (shown in figure 3.6) is designed using modular dataflow architecture and consists of a number of reusable modules that perform data collection, filtering, storage and analysis. Each module is defined as having input and output ports that allow modules to be connected to each other in analysis chains. Both input and output ports allow multiple connections - thus facilitating building complex analysis scripts.

3.11.1 Message Source

A message source module taps message traffic of any group of agents (either Red or Blue Team). It outputs a raw message stream that can be parsed and filtered by other modules.

3.11.2 Message Filtering

A message filter of the Observer Agent selectively intercepts messages from the message stream and routes them to the data accumulator or discards them.

The filter can be configured by the Strategy Module to intercept messages based on

- random criteria,
- message sender or recipient,
- message type (knowledge, task, resource, chatter),
- message content, or
- any combination of the above.

Both input and output of a message filter are message streams, thus multiple message filters can be used sequentially or in parallel

3.11.3 MetaMatrix Aggregator

A MetaMatrix aggregator receives a stream of messages and stores their header and contents in a MetaMatrix structure. It inputs a message stream and outputs a set of graph structures comprising the MetaMatrix.

- Header information (Sender and Recipient of the message) is stored as Person-to-Person edge
- Knowledge contents are stored as a Person-to-Knowledge edge
- Resource contents are stored as a Person-to-Resource edge
- Task contents are stored as Person-to-Task edge

3.11.4 Vector Aggregator

A Vector Aggregator receives a stream of messages and stores values of a field in a message in a vector of numbers. The output of the aggregator is a vector that can be analyzed with the statistics module

3.11.5 Network Analysis

The Social Network Analysis (SNA) module uses metrics implemented in ORA[Carley and Reminga, 2004]. These metrics include both standard SNA metrics (such as network density, degree centrality, betweenness and eigenvector centrality), as well as MetaMatrix measures such as cognitive demand and task and knowledge exclusivity.

Metrics computed on graphs are stored as node and edge attributes in the graph data structure or can be output as vectors.

3.11.6 Storage Modules

Storage modules allow the Observer Agent to output collected data in a variety of file formats, ranging from raw text files and UCINET's DL format, to DyNetML input/output streams (described in chapter 8).

A second storage module provides integration with the NetIntel databases (described in chapter 9).

3.11.7 Summary

As I have shown in this section, NetWatch is designed as a multi-agent system incorporating autonomous, asynchronous and intelligent agents embedded in complex social networks. By employing a full-featured agent communication language (ACL) and a combination of a set of basic behaviours and AI-based planning subsystems, each agent can produce a high-fidelity approximation of behaviour expected of a human actor in a similar organizational situation. The next section describes a virtual experiment aimed at demonstrating the potential of NetWatch as a model of organizational evolution and grounding it by comparing results of the simulation with these from previous models.

3.12 Virtual Experiment 1: Structural Evolution in Covert Network

The first virtual experiment showcases NetWatch's ability to model the processes of task-related and social communication among agents and their effect on the structural properties of the network. Of particular interest is a question of whether or not task-based activities preserve the original network topology - or result in creation of a new, different topology that is better suited to the task at hand.

A further purpose of this experiment is to ground NetWatch's performance as a model of organizational performance and evolution by docking its results with these of Construct[Schreiber and Carley, 2004].

3.12.1 Experimental Design

The experiment is framed using the following parameters:

Parameter	Value
Number of Agents	100
Total number of facts in knowledge network	200
Total number of available resources	20
Number of subtasks in task structure	20

The network of agents is built using the cellular network construction routine discussed in section 6.0.1, using the following profile:

Parameter	Value
Mean Cell Size	6
Cell Size St. Deviation	1.7
Internal Cell Density	0.9
Probability of Random Connections	0.01
Density of cell leaders	0.16
Probability of connection between leaders	0.6
Probability of triad closure within cells	0.9
Probability of triad closure outside cells	0.17

The knowledge network of agents was generated using a routine described in section 6.0.3, using the following parameters:

The task precedence network was generated using the Critical Path Method (section 6.0.4) with the Average Connectivity (sum of predecessors and successors) of 4.0 and Average Branching Factor of 1.7.

Parameter	Value
Proportion of shared knowledge	0.15
Proportion of specialist knowledge	0.65
Proportion of privileged knowledge	0.2

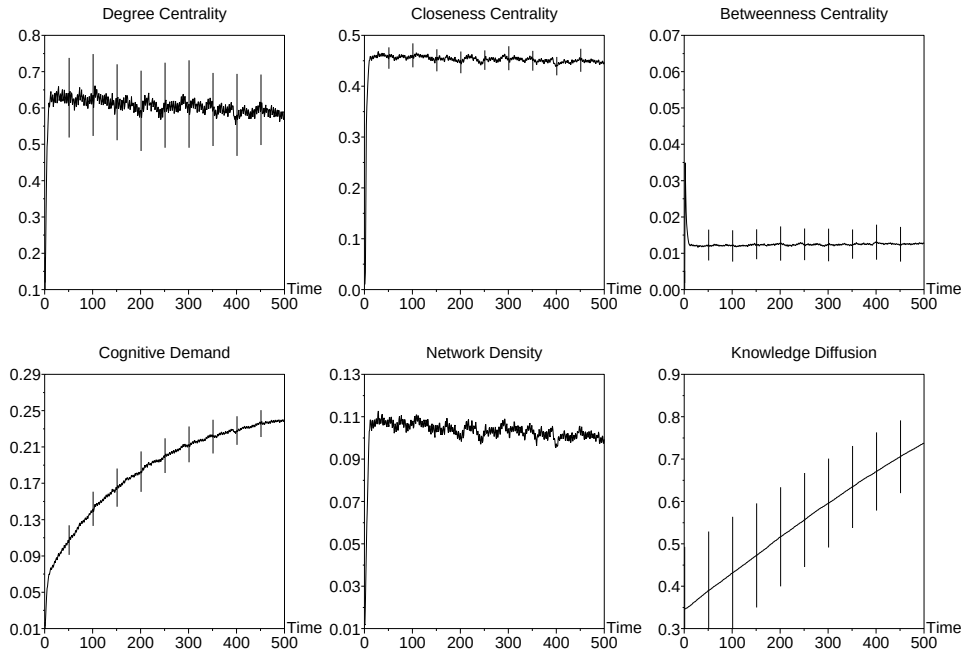


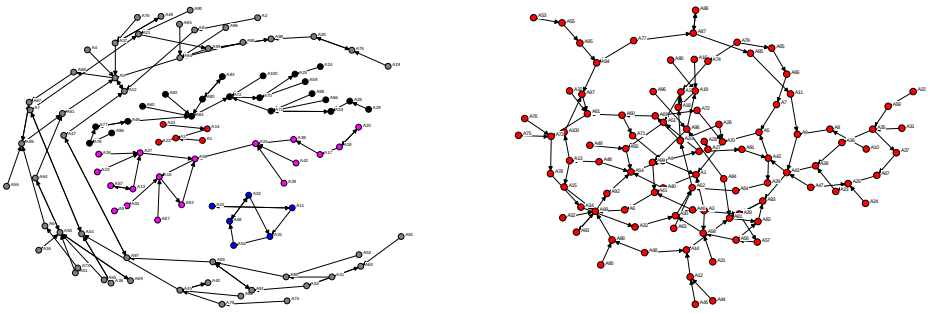
Figure 3.8: Time-series measurements: change in (a) average degree centrality, (b) average closeness centrality, (c) betweenness centrality, (d) cognitive demand, (e) network density and (f) knowledge diffusion;(Cellular network, 100 agents, 20 repetitions)

For each experimental configuration, I generated 20 initial network topologies and repeated the experimental run with each network. While the parameters given to the network generator stay the same, stochastic nature of network generation results in generation of different initial topologies. Differences in initial topologies thus result in larger variance in some of the dependent variables. All experiments are run for 500 time periods.

3.12.2 Observations

Change in Average Node Degree Over Time

Figure 3.8(a),(b) and (c) show change in agent centralities over the course of the simulation. Average betweenness and closeness centralities stay fairly constant throughout the run, while average degree centrality drops slightly as agents solidify their local networks. Network density (fig. 3.8(e)) also drops



Time Period 0 Time Period 499

Figure 3.9: Agent Communication Patterns at the beginning and end of simulation

slightly.

Change in Knowledge Diffusion

Knowledge diffusion[Carley, 1995] occurs when an idea or fact known by one individual becomes known by others in the society. Average level of diffusion is determined by tracking several facts that are originally known by one agent only, and assessing the number of agents that report knowing this fact at every time period.

In random networks, Carley[Carley, 1995] shows the knowledge diffusion function to monotonically increase with time with an asymptote at or near 100% diffusion. The functional form of knowledge diffusion is close to

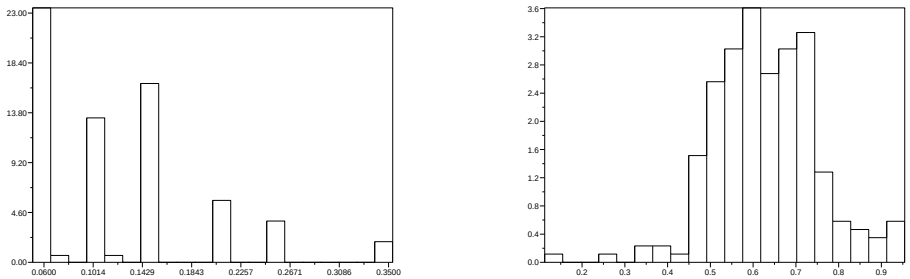
$$d = 1 - \frac{1}{t}$$

where d is the information diffusion ratio and t is a time period.

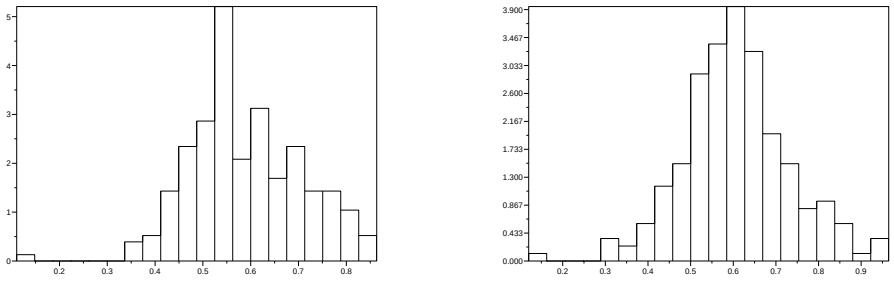
Experiments with cellular networks (figure 3.8(f)) show a slower knowledge diffusion ratio with mean diffusion showing near-linear progression. This is expected due to the fact that cellular networks are designed precisely for slowing information diffusion - a necessary precondition for maintaining secrecy of covert plans. The large standard deviation shows that despite linear growth of the mean diffusion, the precise configuration of the initial network and knowledge distribution affects speed of knowledge distribution as well.

3.12.3 Changes in Network Topology Over Time

Figure 3.9 shows that while the topology of the network changes through the simulation and overall network density increases, the cellular nature of the network is maintained through the interactions.



Time=0 Time=100



Time=200 Time=300

Figure 3.10: Change in degree distribution over time; (Cellular network, 100 agents, 20 repetitions)

However, average degree centrality of individual nodes does increase and its distribution changes from a sparse pattern corresponding to cells and leadership structures originally generated into a pattern more closely resembling a normal distribution, and thus signalling creation of a more egalitarian structure (figure 3.10.

3.12.4 Task Performance

Figure 3.11 shows that learning that occurs in agents as they communicate and accomplish tasks contributes to organization's average task performance. The graph shows performance on classification tasks only, which has been shown as a good estimate of organizational performance in knowledge-intensive tasks.

The monotonic increase of task performance over time demonstrates that the organization learns and optimizes its arrangements of knowledge and resources. However, the organization is faced with tasks of varying composition. Thus, until a clearer model of the task structure emerges, the performance on each individual task varies significantly – within 10% range

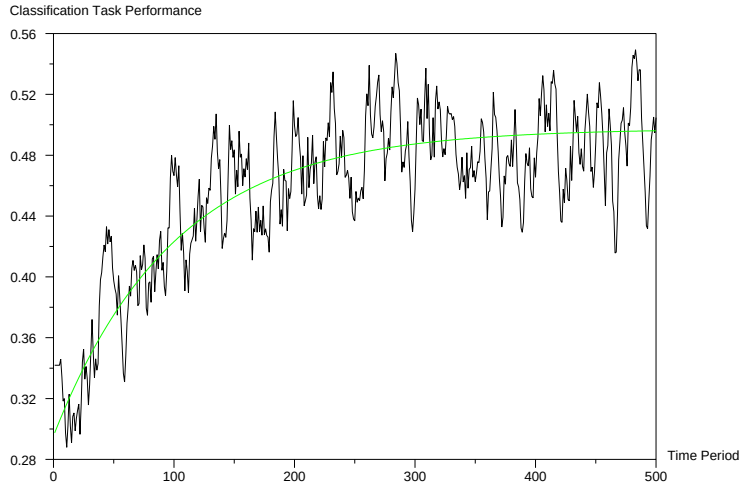


Figure 3.11: Time-series measurements of classification task performance (Cellular network, 100 agents, 20 repetitions)

of the mean.

3.12.5 Discussion

The results of this experiment indicate that, while overall structure of the organization remains the same, processes of information exchange and pressures of task performance force the organization to become more egalitarian. As knowledge becomes more distributed throughout the organization, the amount of knowledge exclusive to a small group of agents also decreases, providing for a structure that is more resilient to attack.

Essentially, the organization adapts to increase task performance. This optimization process is not without bounds, as the maximum task performance is shown to be limited by formal organizational structure and its power to prevent information and resources from crossing large distances through the network.

In order to provide face validity for studying performance of signal intelligence heuristics as well as the effects of destabilization strategies, a baseline model of organizational evolution and performance must be established.

The purpose of this experiment is to provide such a baseline by examining the performance of an organization constructed based on studies of covert networks and built of intelligent agents.

Results of this preliminary experiment can be summarized as follows:

- Organizational models constructed with use of intelligent agents evolve in patterns similar to these observed in empirical data[Carley, 2002c].

- Simulated organization exhibits patterns in growth of performance and knowledge diffusion as these established by Carley in [Carley and Hill, forthcoming], [Carley, 1995].
- Simulated organization evolves structurally by building a network topology that improves performance in knowledge-intensive tasks. However, this structural change preserves the character of a network as a cellular, covert organization. Organizational evolution happens through both link addition and link deletion.
- If left unchecked, the covert organization will gain efficiency in task performance by moving resources where they are needed, and disseminating knowledge useful to task completion.

3.13 Example: Four Days in The Life of a Terrorist Cell

To simulate evolution of social networks and organizations, NetWatch implements a multi-level complex system that involves a large number of simultaneous processes. These processes range from socializing in groups to strategic interactions to distributed planning and execution of tasks. To maintain realism of the system and a level of face validity, it is thus important to not only examine overall outcomes of simulation scenarios, but to also examine individual interactions and processes that occur within the simulated organization.

Doing so on a large-scale basis (e.g. an organization with hundreds of agents, over a long period of time) is impossible due to sheer quantity of generated data, but it is possible to trace interactions, tasks and information flow over a short period of time, with a confined group of agents engaged in a focused task.

The initial data for this example comes from encoding of news stories and legal documents related to the bombing of the U.S. Embassy in Tanzania in 1998. The data has been encoded in MetaMatrix form and includes a person-to-person network with 16 agents, 5 areas of knowledge (*religious extremism, weapons training, driving training, bomb preparation knowledge and media relations*), resources such as *truck, bomb material and building for bombmaking* and a precedence graph containing the main task (*bombing*) and a number of auxiliary tasks (*training exercises and bomb preparation*).

In this example, I have triggered the organization to plan and execute a terrorist attack using a truck bomb. Each of the steps described in the trace below is an agglomeration of activities that have occurred in 1 day (i.e. 4 time periods of the simulation). Thus, in a relatively short space,

the simulation can cover execution of a terrorist attack from beginning to completion.

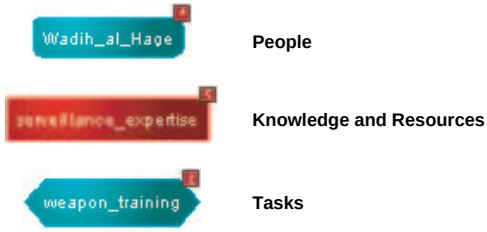


Figure 3.12: Legend for plots 3.13-3.16

Figure 3.13 shows the state of the organization before planning and execution of a terrorist attack starts[1]. Before planning phase was triggered, the agents were allowed to run freely for 60 time periods (i.e. 15 days). Thus, a certain level of information exchange and attribute contagion has been already achieved. Note, for example, the prominence of *religious extremism* and the number of people connected to it.

Triggering of the terrorist attack occurred in the beginning of this day, with a broadcast message to all agents. Only agents that were able to produce a hierarchical decomposition of the complex task and find out the subtasks and resource requirements were able to start planning the task. Some of the resources (*money and a building for bombmaking*) have been already located by the end of the day, and people with access to these resources will likely become anchors to the planning process of the terrorist attack.

Figure 3.14 shows the organization in the midst of planning. Note agents *Abdullah Ahmed Abdullah (Saleh)* and *Mohammed Rashed Daoud al-Owhali*. A day ago they had access to the building for bomb preparation and money. Now they also have purchased a truck from *Abouhalima* — i.e. a resource exchange process — with the money that they had during Day 1. At the end of the day, *Saleh* and *al-Owhali* engaged in weapons training and bomb preparation (subtasks on the critical path of execution of a bombing), and enlisted help of a weapons export *Khalfan Khalis Mohamed*.

During Day 2, preparations were well under way, but explosive material has not been yet procured. Transactive memory prompted *al-Owhali* to come back to *Abouhalima* as he was able to procure resources. *Abouhalima* was a familiar person to *al-Owhali* as *al-Owhali* bought a truck from him during Day 2.

However, he did not have explosives. The closest place to obtain them was from *Wadih al Hage*, at a network distance of 2. *al-Owhali* was introduced

[1]This example is fully simulated. While names of actors and initial distribution of skills and resources have been obtained from real-world information, the events described below are a product of a computer simulation and should not be construed to indicate the course of historical events or any legal implications thereof

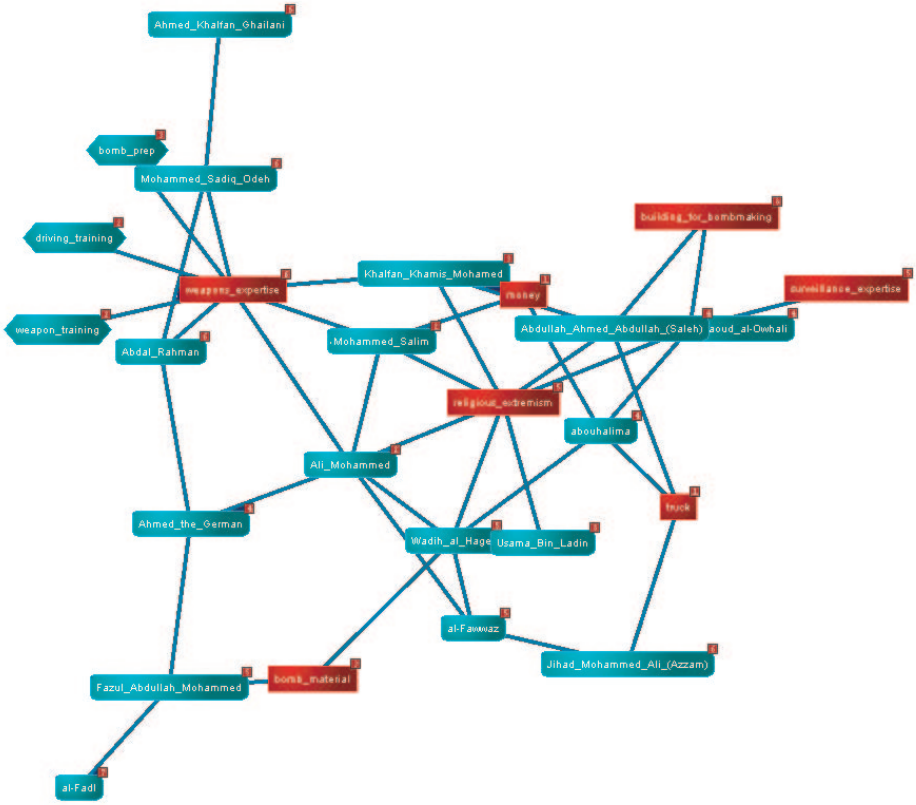


Figure 3.13: Day 1: Start of planning

Figure 3.14: Day 2: Planning and Resource Marshalling

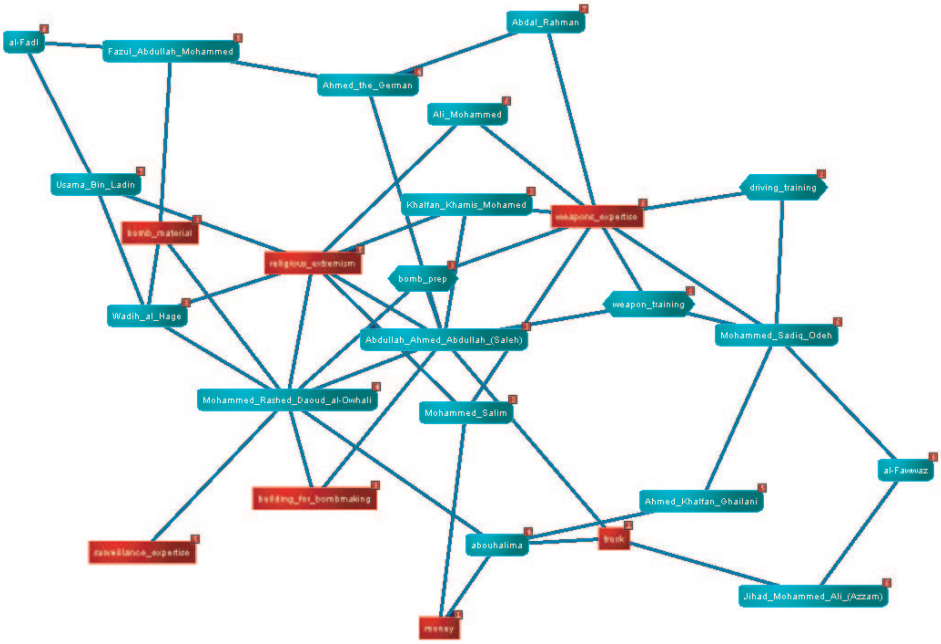


Figure 3.15: Day 3: Final Preparations

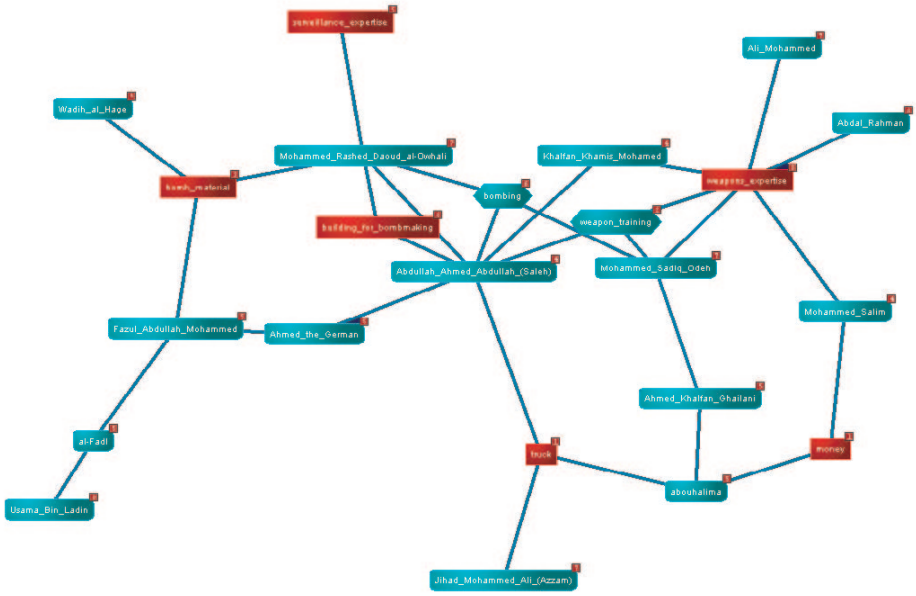


Figure 3.16: Day 4: The Bombing

to *al Hage* in the beginning of Day 3 (figure 3.15), and procured explosives by the end of the day.

Meanwhile, *Saleh* has been engaged in *weapons training*. There he met *Mohammed Sadiq Odeh* who has had *driving training* and recruited him to drive the truck bomb.

The bombing itself happened on Day 4 (figure 3.16). It was perpetrated by *al-Owhali*, *Saleh* and *Odeh*. The model used for NetWatch does not let us find out whether the perpetrators survived and got away or died in the blast.

This "dramatic reading" of the scenario illustrates in human terms how a set of intelligent agents goes about accomplishing a complex task - via planning, resource acquisition and knowledge exchange. While most of the simulation scenarios do not read like a paperback novel, the processes behind them stay as complex as the ones illustrated above. While we may only look at the top-level results, and condense dozens of runs into a single diagram, we must keep in ming this underlying complexity.

Chapter 4

Learning Network Structures from Observed Communications

As I have discussed in section 3.2, the Blue Team represents a set of agencies engaging in anti-terrorist activity, and pursues two interconnected goals. The blue team conducts signal intelligence-based informaiton gathering and uses collected information to build a MetaMatrix representation (see sec. 2.6) of the Red Team, e.g. learn the structure, task assignments and knowledge distribution of the Red Team.

However, the Blue Team has no access to the actual information about the Red Team. Its only source of information is a set of wiretaps on the communication network of the Red Team, which are (a) noisy and (b) incomplete sources of information. The Blue Team must therefore cope adequately with the uncertainty introduced by incomplete information obtained from wiretaps.

The second goal of the Blue Team is to destabilize or decrease performance of the Red Team by isolating agents on the Red Team through use of various destabilization strategies as outlined in section 4.12.

This section is organized as follows: after overview of design of blue team agents (sec. 4.0.1), I proceed to define and evaluate a number of wiretapping methods and heuristics. First, a baseline performance is established through implementation and testing of random sampling algorithms with control for signal-to-noise ratio (section 4.0.2). Section 4.2 defines metrics of wiretap performance and means for comparison of different algorithms and presents a virtual experiment to study performance of random sampling algorithms under various signal-to-noise ratio conditions.

Capture and analysis of each message has associated costs. Therefore, a signal-to-noise ratio — i.e. the amount of messages captured *vs.* the total amount of message traffic — can be established as a measure of cost of running a wiretap. The goal in this case is to achieve the maximum accuracy of estimating the nodes and edges in the subject network at the lowest possible cost.

Section 4.3 discusses snowball sampling - a common technique for obtaining social network data. Snowball sampling starts with a small initial population and follows the links of every actor to find who they commu-

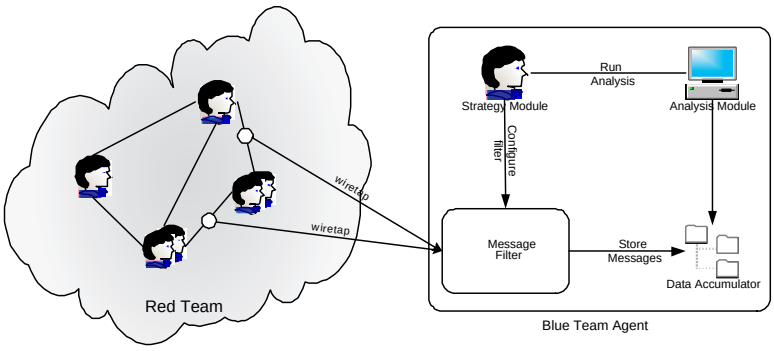


Figure 4.1: Design of the Blue Team Agent

nicate with. The population expands as new agents are found, thus search progressing radially from a small initial sample. In this section, I also discuss a common problem with snowball sampling and introduce a simple algorithmic solution, which is used to implement snowball sampling in NetWatch.

Section 4.4 discusses a set of sampling strategies that utilize social network analysis of known structure of the subject network to guide a targeted wiretap. These strategies, further referred to as *socially intelligent wiretaps*, improve upon snowball sampling by achieving a higher degree of learning of the subject network at a lesser signal-to-noise ratio (e.g. at a lower cost).

Section 4.5 puts snowball sampling and four socially intelligent sampling strategies to the test against the baseline random sampling and discusses its performance and limitations in study of dynamic social networks.

Sections 4.6 and 4.7 discuss and test a new approach to sampling dynamic social networks based on simulated annealing. By improving breadth of coverage of the search space and eliminating problems with local maxima, the annealing-based algorithms achieve the highest performance of all strategies discussed in this dissertation.

Section 4.11 discusses the means for sharing information among agents in the blue team and provisions for implementation of such sharing strategies.

4.0.1 Design of the Blue Team Agents

A Blue Team agent is built on a similar base as the Red Team agents but with a number of key differences regarding its tasks and behaviours (see figure 4.1). Blue Team agents have a capability of observing message traffic on the Red Team network while performing traffic filtering and traffic analysis. Signal intelligence thus obtained is accumulated to form a representation of the Red Team and analyze behaviours and structure of the Red Team.

Each Blue Team agent consists of the following modules:

- Message Filter module receives raw message traffic from wiretaps placed

by the Blue Team agent into the Red Team network.

- Data Accumulator is a MetaMatrix structure that is used to accumulate information on sources and destinations of messages, as well as their content (such as knowledge or resource transfer information or reports of task assignments and accomplishments). The Data Accumulator module also performs aging of network data while discounting old or infrequently activated nodes and links.

- Analysis Module performs MetaMatrix and Social Network analysis on data accumulated in the graph structures of Data Accumulator and reports to the Strategy Module.

- Strategy Module uses results of analysis of gathered social network data to reconfigure the message filter in ways that may increase the quantity or quality of captured information.

If a wiretap could capture all communication, the Blue Team would be able to create a full and accurate picture of the Red Team network. However, a wiretap is not able to capture all relevant messages due to its placement in the Red Team network or the signal-to-noise ratio. Thus, it is a goal of each Blue Team agent to maximize the effectiveness of its data gathering through capturing higher amounts of relevant messages. Since placement of wiretaps and capture of messages have costs, the Blue Team also needs to minimize costs by operating at lower signal-to-noise ratios.

4.0.2 Message Filtering

A message filter of a Blue Team agent selectively intercepts messages from the message stream and routes them to the data accumulator or discards them.

The filter can be configured by the Strategy Module to intercept messages based on

- random criteria,

- message sender or recipient,

- message type (knowledge, task, resource, chatter),

- message content (messages containing new information get higher priority), or

- any combination of the above.

4.1 Signal Intelligence Strategies

The strategy module of the Blue Team agent implements a number of configurable strategies for tapping and filtering message traffic data.

The baseline strategy is based on *random selection* of messages as they come through the the wiretap. Any message in the stream has an equal probability of being intercepted and recorded. Signal-to-noise ratio is directly related to the probability of interception and can be treated as an independent variable, ranging from 5% useful signal to 25% useful signal - thus enabling study of the effect of signal-to-noise ratio on the quality of collected data.

4.2 Learning Network Structure though Signal Intelligence: Random Sampling

Random sampling of communications can be considered a baseline against which performance metrics of other network sampling strategies can be compared. While random sampling of wire (or wireless) traffic is rarely used in the real world, in simulation it can be used to provide a robust notion of signal-to-noise ratio and its effect on acquisition of knowledge about network structure.

As the true goal of intelligent wiretapping heuristics is to *"do more with less"* — i.e. produce a maximally correct result given a certain amount of captured traffic — it is prudent to compare quality of network structure discovered via intelligent heuristics with results of similarly configured random sampling systems.

In this experiment, I establish such a baseline by comparing performance of random sampling techniques across networks of different size and topology, while controlling for signal-to-noise ratio of the sampling apparatus.

4.2.1 Experimental Design

This experiment is based on a matrix design and tests performance of a number of simple wiretapping strategies on simulated organizations of different size and initial topology:

	Network Topology		
Size	Uniform, 100 agents	Scale-Free, 100 agents	Cellular, 100 agents
	Uniform, 250 agents	Scale-Free, 250 agents	Cellular, 250 agents
	Uniform, 500 agents	Scale-Free, 500 agents	Cellular, 500 agents

The density of uniform random network is set at 0.2 - i.e. probability $P_{i,j}$ of an edge existing between agents i and j is 20%. The method for generating such networks is described in section 6.0.1

The scale-free network are grown using the Barabasi-Albert method of preferential attachment[Barabási and Albert, 1999], with parameters:

$$k = 0.25$$

$$\gamma = 2$$

More details on growing scale-free networks can be found in section 6.0.1

The cellular networks are generated using the mechanism outlined in section 6.0.1, using the following profile:

Parameter	Value
Mean Cell Size	6
Cell Size St. Deviation	1.7
Internal Cell Density	0.9
Probability of Random Connections	0.01
Density of cell leaders	0.16
Probability of connection between leaders	0.6
Probability of triad closure within cells	0.9
Probability of triad closure outside cells	0.17

Each cell of the experiment was repeated 20 times, generating 20 random networks with the same parametric signature. Results of the experimental repetitions were averaged.

4.2.2 Performance Measurement

Techniques for measurement of effectiveness of signal intelligence strategies can be broken down into two classes of metrics. Employment of one set of metrics vs. the other is a choice to be made depending on goals of the measurement. All of the below techniques assume the existence of two Meta-Matrix structures: one of the True Network — network collected through direct observation of the simulated organization and the Learned Network — network collected by the Blue Team agent in the course of application of the wiretapping strategy.

The first technique is based on the assumption that the goal of discovering nodes and connection patterns between them is to achieve a maximally complete picture of the overall network structure. The concern in this case is the surveillance technique needs to not only uncover highly visible actors in the network but also discover the breadth of the network structure and minimize the number of undiscovered nodes and edges.

If this assumption is true, the best simple measure of quality of network is *hammingdistance*[Sanfeliu and Fu, 1983], a sum of differences between two graph structures, in this case, the True and Learned Networks. The significance of hamming distance in this particular case is that it illustrates the overall number of errors made by the Blue Team agents. However, to compare performance of an algorithm on networks of different size, the raw hamming distance needs to be normalized:

$$D^{norm}_{hamming} = \frac{D_{hamming}}{numberOfPossibleEdges} = \frac{D_{hamming}}{numberOfNodes^2}$$

Lower normalized hamming distance signifies higher performance.

The second measurement technique is a strict assessment of an algorithm's ability to zero in on a core members of a network. The definition of a core member of the network is based on results of applying a number of social network measures to the network, including *degree* and *betwenness* centralities[Freeman, 1979], *cognitive demand*[Carley and Ren, 2001] (also section 2.6.1, *task* and *knowledge* exclusivity[Ashworth, 2003](also section 2.6.1.

Agents are thought to be a part of the core group in an organizational network if they exhibit high values in one or more of these social network metrics. Practically, one can think of these individuals as members of the "Top 10" actors in the network, however mathematically it would be more prudent to describe these actors as

$$centrality_i \geq max(centrality) - \sigma^2(centrality)$$

or actors whose centrality falls within 1 standard deviation of the top centrality value.

Performance of the wiretapping strategy can then be measured as the probability of learning correctly the members of the core group or cardinality of the intersection of learned core group with the true core group divided by the cardinality of the true core group:

$$P = \frac{cardinality(core_{true} \bigcap core_{learned})}{cardinality(core_{true})}$$

This measure of performance is most useful under the assumption that finding the core group through SNA metrics is an acceptable compromise to mapping the entire network.

This presents us with a trade-off. If an intelligence gathering strategy is designed to achieve optimal hamming distance performance, it will inevitably miss some of the structural properties related to the core structure. However, if a technique is optimized to achieve optimal core-group estimates, it may not be able to adequately map the peripheral structure of the network. In the context of terrorist groups, that would mean that the first technique would

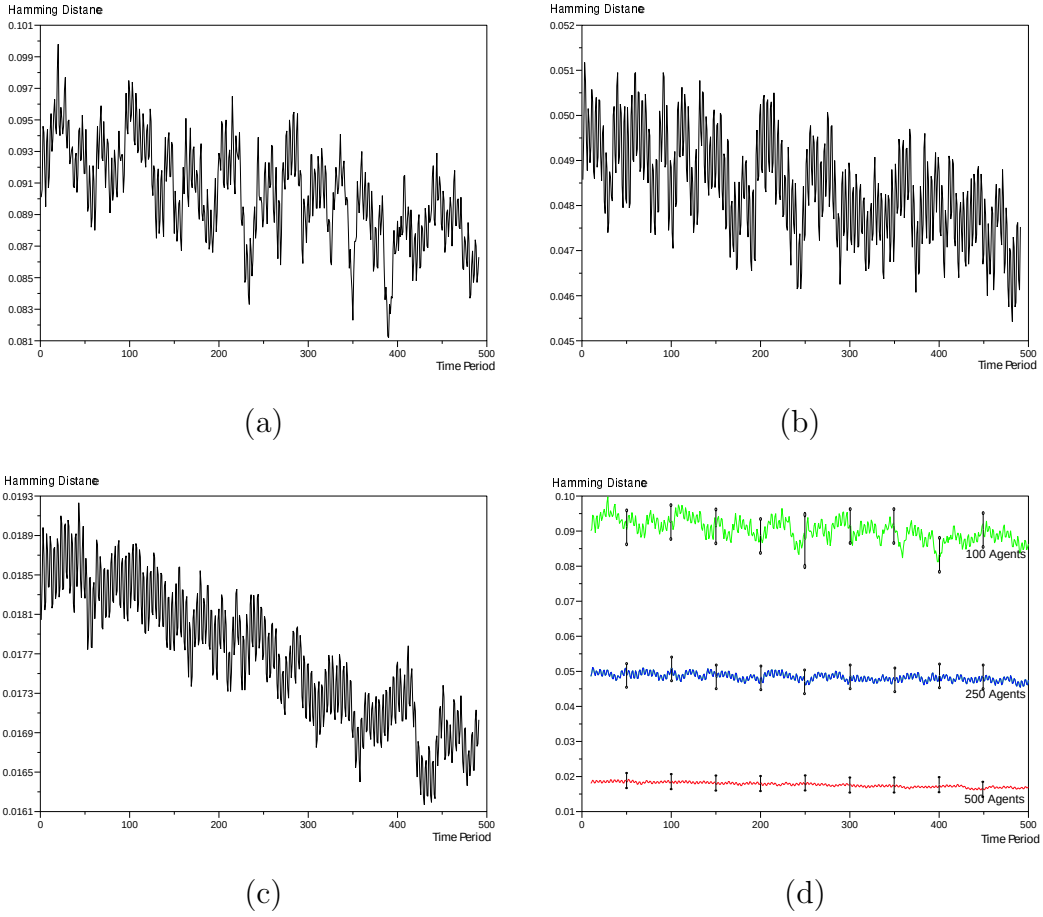


Figure 4.2: Time-series: Effect of network size on Hamming Distance with Random sampling (10% signal/noise ratio), Cellular Networks. (a) 100 agents, (b) 250 agents, (c) 500 Agents, (d) normalized hamming distance for 100, 200 and 500 agents; (all results averaged over 20 runs)

be bad at finding the leadership of the group. The second technique would not be able to find peripheral groups of operatives – which are more likely to become future perpetrators of terrorist attacks.

In this experiment, we study performance of a number of simple network observation strategies in terms of both normalized hamming distance and core-group metrics.

4.2.3 Observations and Discussion

Effects of Network Size on Hamming Distance

Figure 4.2 shows, as a time series, the process of learning the shape of the True Network through a simple uniform-probability wiretapping strategy (see

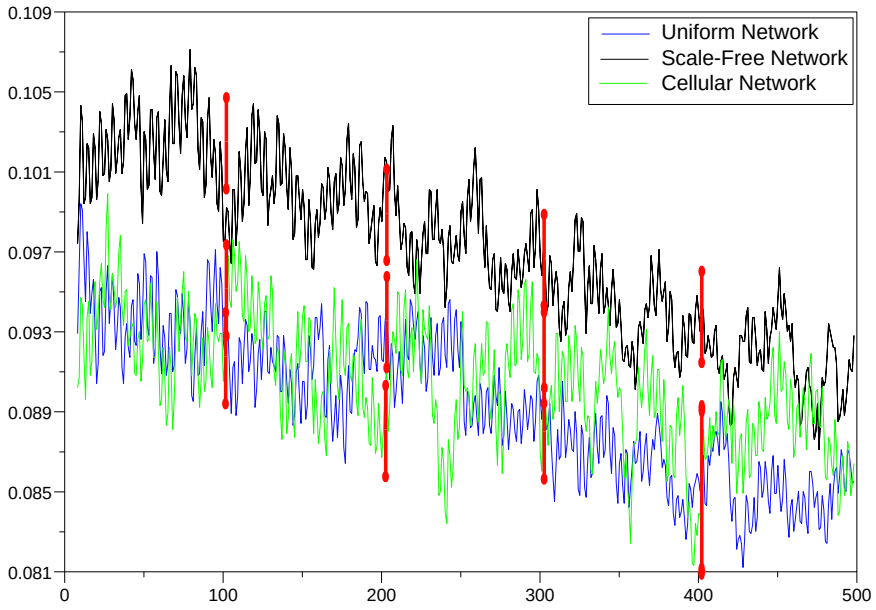


Figure 4.3: Effect of Initial Network Topology on Wiretap Performance, (100 agents; averaged over 20 runs)

section 4.0.2). In this portion of the experiment, the signal-to-noise ratio was set at 0.1 - i.e., any message exchanged by Red Team agents has a 10% chance of being intercepted by the Blue agents.

Plots 1,2,and 3 on figure 4.2 show that in each of the cases the amount of information collected by the Blue agents gradually improves the knowledge that the Blue Team possesses about the Red Team (i.e. the hamming distance decreases). However, plot 4 on the same figure shows that in context of the absolute scale of hamming distance such improvement is fairly small.

How could this occur? Agents in the Red Team communicate continuously, therefore releasing information about structure of the Red Team to the Blue agents. The Blue agent collects this information and continuously works on improving its picture of the Red Team.

The problem lies in the fact that the Red Team network evolves as agents create new connections and let old connections expire once they are no longer needed. Thus, as the Blue Team agents monitor the Red Team, they must force their representation of the Red Team network to follow the changing communication landscape.

In general, the Red Team will change faster then the Blue Team can follow due to the fact that not every message is captured. The hamming distance

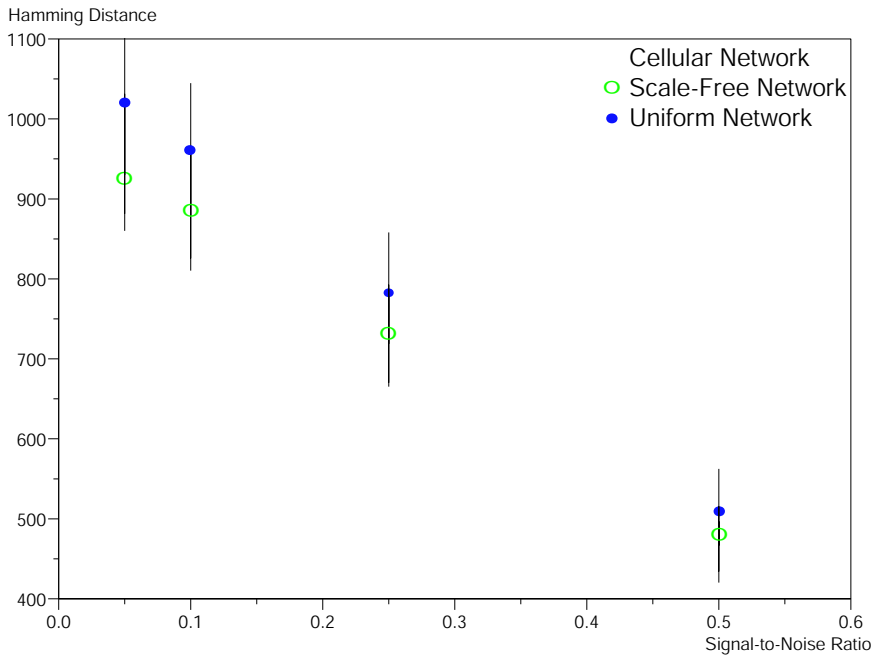


Figure 4.4: Effect of Signal-to-Noise Ratio Wiretap Performance; (averaged over 100, 200 and 500-agent networks, 20 runs in each configuration)

between the True network and the Learned network becomes a measure of how well the Blue Team can match the speed of evolution of the Red Team.

Effects of Network Topology on Wiretap Performance

Figure 4.3 shows effect of initial network topology on effectiveness of wiretaps. All of these tests were conducted on networks of 100 agents, 10% signal/noise ratio. The most pronounced gain in hamming distance is for Erdös random graphs with scale-free and cellular networks approximately equivalent.

4.2.4 Effects of Signal-to-Noise Ratio on Wiretap Performance

This portion of the experiment combines data collected for all initial topologies of the experimental design sampled at different levels of signal-to-noise ratio.

Figure 4.4 shows the effect of signal-to-noise ratio of the random sampling wiretap on the quality of acquisition of network data. The conclusion drawn from this figure on its own is fairly obvious — greater signal-to-noise ratio

has a significant impact on the quality of learned network. However, this dataset is used in the experiments that follow as a point of reference and a baseline that other results are compared to.

The baseline results show a near-linear dependence of accuracy of network mapping on signal-to-noise ration, in case of purely random sampling of communications. This is expected due to the fact that random sampling has an equal chance of discovering all edges of the network, whether they belong to a highly connected agent or to a near-isolate agent. The probability of discovery of an edge at any given time is thus proportional to the ratio of messages that are captured and overall message traffic - which comprises the effective signal-to-noise ratio.

4.3 Snowball Sampling

A *Snowball Sampling* strategy is based on work of Biernacki and Waldorf[Biernacki and Waldorf, 1981] and Granovetter[Granovetter, 1976]. A snowball sampling strategy captures traffic originating from one agent and targets every agent with which it communicated. This essentially is a breadth-first search of the network. In NetWatch the snowball sampling strategy targets agents sequentially, one at a time.

While snowball sampling can quickly map communication in smaller social networks, it exhibits a number of problems. First of all, it can only discover agents that reside inside a single component of the network. This problem is compounded by the fact that cellular networks consist of semi-isolated groups of agents where frequency of communication inside the group is much greater then frequency of communication outside the group. Mapping out multiple components of the network requires snowball strategy to use multiple, randomly selected entry points.

Further, as agents are targeted sequentially, a problem of oscillation arises. In a sub-network resembling a star topology, a snowball sampler has to return to the center of the star before it can continue to sample communications from other agents. This can be resolved by using a queue to manage a list of unexplored targets and a taboo list to prevent the search algorithm from revisiting the targets that it has already explored.

Figure 4.5 and listing 4.1 illustrates how the snowball sampling algorithm explores a simple graph.

As I have established experimentally in section 4.5, snowball sampling shows a considerable bias towards highly connected nodes. While it is effective at learning network structures at a higher efficiency than baseline methods, it does not discover the breadth of the network by avoiding nodes with low communication rates.

Next section presents a number of strategies that improve upon perfor-

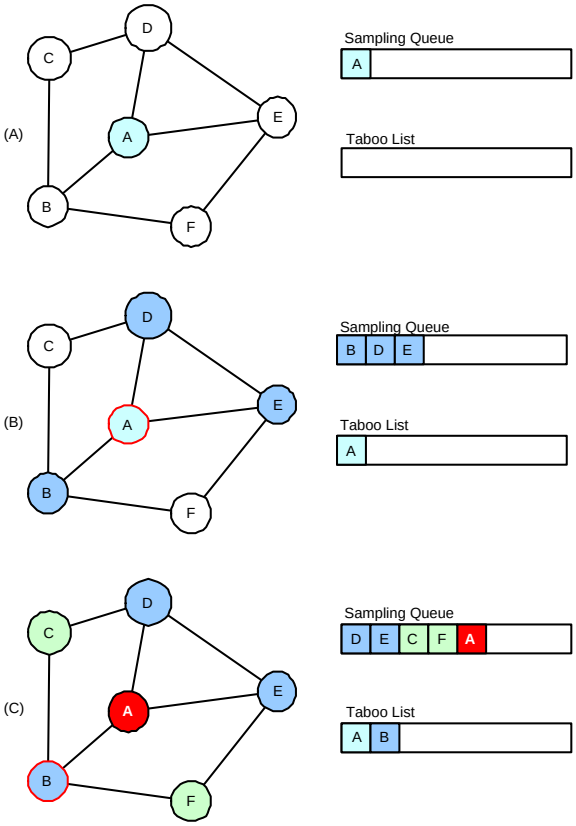


Figure 4.5: Snowball Sampling with Taboo List

mance of snowball sampling on dynamic networks via use of socially intelligent traffic analysis and multi-point sampling.

4.4 Socially Intelligent Traffic Analysis

As the Blue Team agents receive messages from the wiretap agents, they use their address information to build a representation of the network of the Red Team, or the *Learned Network*.

Thus, while Snowball sampling is myopic (i.e. can only see and survey small portions of the network at each time), a more intelligent Blue Team agent can use its accumulated knowledge of the target network to make intelligent decisions about locations of future wiretaps and configuration of their message filters.

The Analysis Module (see figure 4.1) of Blue Team agents implements an analysis toolkit containing a number of common social network analysis algorithms including *degree centrality*, *betweenness centrality* and *closeness centrality* [Freeman, 1979]. Also accessible to the agents are methods

Listing 4.1: Snowball Sampling Algorithm

```
CurrentTarget = a random starting point (A).

REPEAT:
  Add the CurrentTarget to the Taboo List (B)
  Add all agents that CurrentTarget communicates with to the
      Sampling Queue
  REPEAT
    NewTarget = deque from Sampling Queue
  UNTIL NewTarget is NOT on the Taboo List (C)
UNTIL Sampling Queue is empty
```

Listing 4.2: Simple Socially Intelligent Sampling

```
CurrentTarget = random starting point
REPEAT
  Add CurrentTarget to TabooList
  Capture all traffic TO and FROM Current Target for a period of
      time
  Store captured messages in MetaMatrix accumulator
  Run Measure of choice on MetaMatrix
  CurrentTarget = agent highest in Measure
FOREVER
```

of MetaMatrix analysis including *cognitive demand* (see section 2.6.1, [Carley and Ren, 2001]), and knowledge and task exclusivity metrics (see section 2.6.1,[Ashworth and Carley, 2002],[Ashworth, 2003]).

In its simplest implementation, the socially intelligent wiretap algorithm function is presented in listing 4.2.

In this form, the socially intelligent traffic analysis presents a number of problems.

The main problem is rooted in the nature of both traditional SNA metrics and most of MetaMatrix. To be exact, the root of the problem is the fact that centrality metrics such as degree, betweenness and cognitive demand exhibit poor robustness to incomplete data[Borgatti, Carley, and Krackhardt, 2004].

Figure 4.6 illustrates this effect. Let node A represent the node picked randomly in the beginning of the algorithm. After listening to message traffic to and from node A, the Blue Team agent will discover nodes B, D, and E. At this point, the Blue Team agent has accumulated enough information to assume that the topology of A's network is a star. Computing any of the centrality measures mentioned above will produce the same result: the top Red Team agent in all of the measures will invariably be A. Thus, the simple algorithm never has a chance to discover nodes C and F because they do not communicate to A directly.

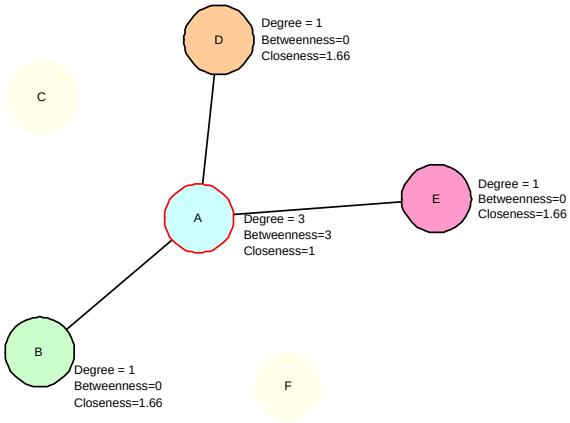


Figure 4.6: Simple Socially Intelligent Traffic Sampling Does Not Capture the Entire Network

4.5 Non-Random Signal Intelligence Strategies: First Approach

The goal of this experiment is explore performance of a number of different wiretapping strategies that utilize the knowledge of structure of the Red Team network to direct their efforts of data capture.

The wiretap heuristics that I evaluate in this experiment are:

Snowball Sampling strategy is based on work of Biernacki and Waldorf[Biernacki and Waldorf, 1981] and Granovetter[Granovetter, 1976]. A snowball sampling strategy captures traffic originating from one agent and targets every agent with which it communicated. This essentially is a breadth-first search of the network. In NetWatch, the snowball sampling strategy targets agents sequentially, one at a time, and switches targets periodically. Snowball sampling is described in detail in section 4.3.

Simple SNA-based Sampling uses a social network analysis toolkit containing a number of common social network analysis algorithms, including *degree centrality*, *betweenness centrality* and *closeness centrality* [Freeman, 1979], as well as methods of MetaMatrix analysis, including *cognitive demand* (see section 2.6.1, [Carley and Ren, 2001]), and knowledge and task exclusivity metrics (see section 2.6.1,[Ashworth and Carley, 2002],[Ashworth, 2003]). This experiment uses socially intelligent sampling algorithm as described in section 4.4.

I evaluate performance of each of these strategies based on normalized hamming distance metric defined in section 4.2.2. The evaluation is done

against the baseline of random sampling experiments established in section 4.2.4.

4.5.1 Experimental Design

The goal of this experiment is to evaluate performance of several simple algorithms and heuristics for learning network structure based on captured communications. For this initial evaluation, I conduct the experiment using cellular networks.

The size of networks is kept constant throughout the experiment:

Parameter	Value
Number of Agents	100
Total number of facts in knowledge network	200
Total number of available resources	20
Number of subtasks in task structure	20

The cellular networks are generated using the mechanism outlined in section 6.0.1, using the following profile:

Parameter	Value
Mean Cell Size	6
Cell Size St. Deviation	1.7
Internal Cell Density	0.9
Probability of Random Connections	0.01
Density of cell leaders	0.16
Probability of connection between leaders	0.6
Probability of triad closure within cells	0.9
Probability of triad closure outside cells	0.17

4.5.2 Observations

Snowball Sampling

Snowball sampling strategy (see figures 4.7,4.8) performs at a level comparable to the random sampling baseline at signal-to-noise ratio of 25%. However, the measured signal-to-noise ratio of snowball sampling strategy (i.e. the ratio of number of captured messages to number of rejected messages) is markedly lower - 16.3%.

However, as histogram on figure 4.8 shows, snowball sampling shows a considerable bias towards highly connected nodes. While it is effective at learning network structures at a higher efficiency then baseline methods, it

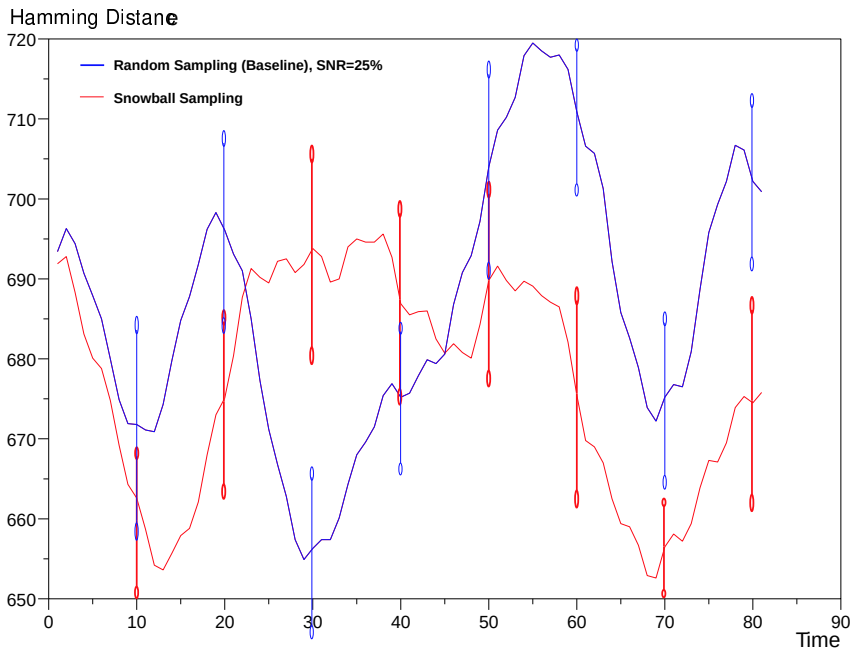


Figure 4.7: Snowball sampling performance: hamming distance (mean of 20 runs)

does not discover the breadth of the network by avoiding nodes with low communication rates.

Simple Socially Intelligent Methods

Socially intelligent sampling methods exploit the partial knowledge of network topology in a heuristic of guiding the sampling algorithms. The heuristics are based on well-known social network analysis metrics and can be summarized as the following conjecture:

> If one samples communication traffic of individuals considered well-connected or powerful, the sampling mechanism will produce a high-fidelity representation of the subject network.

The algorithm for socially-intelligent sampling is discussed in detail in section 4.4. In this experiment, I evaluated performance of metrics of degree centrality, betweenness centrality, cognitive demand and knowledge exclusivity.

Figure 4.9 demonstrates performance of socially intelligent wiretap methods in terms of hamming distance between learned network and true network.

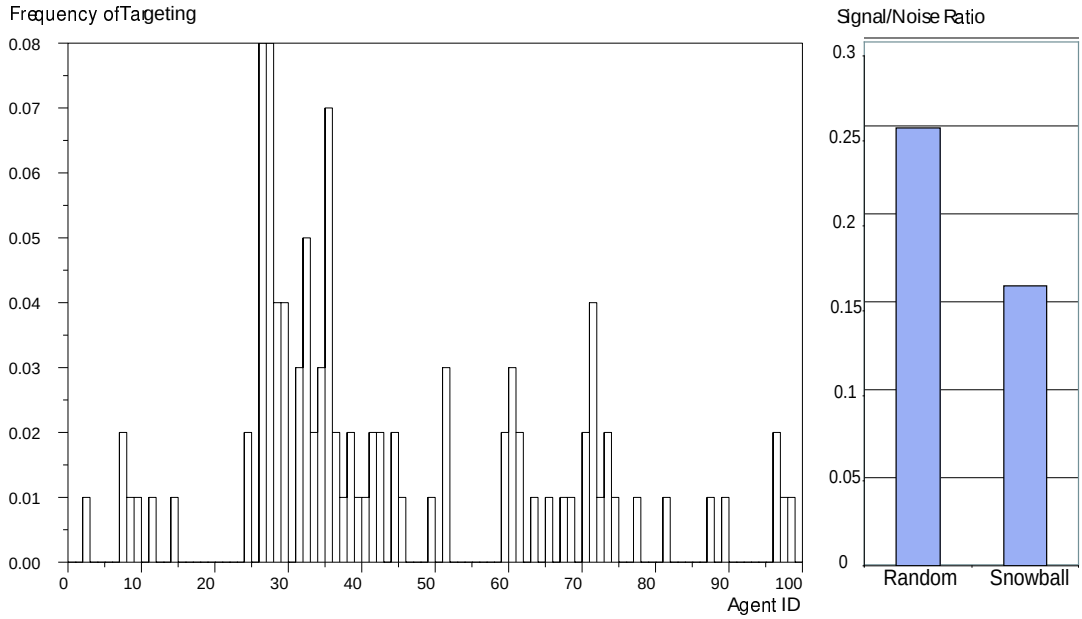


Figure 4.8: Snowball sampling performance; target choice histogram and signal-to-noise ratio of a single run (100 agents)

Heuristics based on pure SNA metrics of degree and betweenness centrality produce essentially identical performance over time and several times reach the best performance among all techniques. However, on average they do not perform as well.

MetaMatrix-based metrics of cognitive demand and knowledge exclusivity track closely to each other in the first half of the run but diverge as knowledge diffusion increases. The explanation of this divergence lies in the fact that with time, knowledge becomes diffused among the agents. When agents send knowledge requests to other agents, with knowledge diffusion these requests will be sent across a larger group of agents — and thus, cognitive demand at time t becomes a worse predictor of who will communicate at time $t + 1$.

Overall, socially intelligent strategies performed significantly better then baseline established for signal-to-noise ratio of 0.25 (see figure 4.10) and slightly worse then baseline at signal-to-noise ratio of 0.5 (figure 4.11. At the same time, the measured signal-to-noise ratios of socially intelligent strategies were significantly lower then those needed to achieve same performance in the baseline strategies.

Socially intelligent sampling strategies as a whole perform significantly better than baseline strategies at the same signal-to-noise ratio as well as

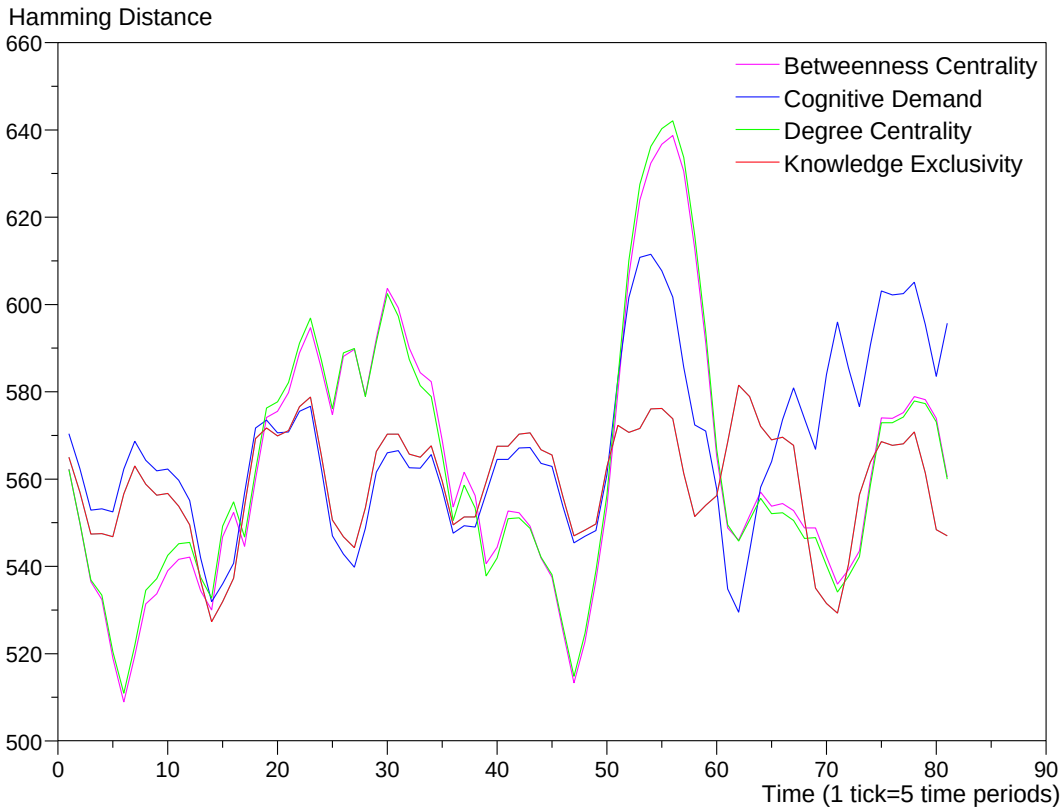


Figure 4.9: Performance of simple socially intelligent heuristics; single run (100 agents)

	rnd-0.05	rnd-0.1	rnd-0.25	rnd-0.5	snwbll	btwns	cog.dem.	deg.	kn.exc
mean	969.98	905.75	729.1	463.3	721.43	582.98	590.99	583.4	580.54
st.dev	166.24	153.165	125.58	77.51	120.48	158.83	157.53	159.12	152.66
s-n ratio	0.05	0.1	0.25	0.5	0.16	0.21	0.18	0.23	0.17

snowball sampling strategy. The highest performance on average comes from strategies that take advantage of knowledge content of communications — cognitive demand and knowledge exclusivity.

However, a significant problem remains in the design of the simple heuristics. Essentially, these heuristics can be described as a hill-climbing algorithm where the sampling point (i.e. the wiretap) moves in the direction of highest value of the metric. However, these algorithms are generally unable to discern local maxima from globally optimal solutions. Once such local maximum is discovered, the hill-climber is unlikely to sample any other area of the network.

Figure 4.12 illustrates the occurrence of local maximum in one of the experiments (cellular network, degree centrality heuristic). The histogram shows frequency with which each of the nodes was targeted by the wiretap.

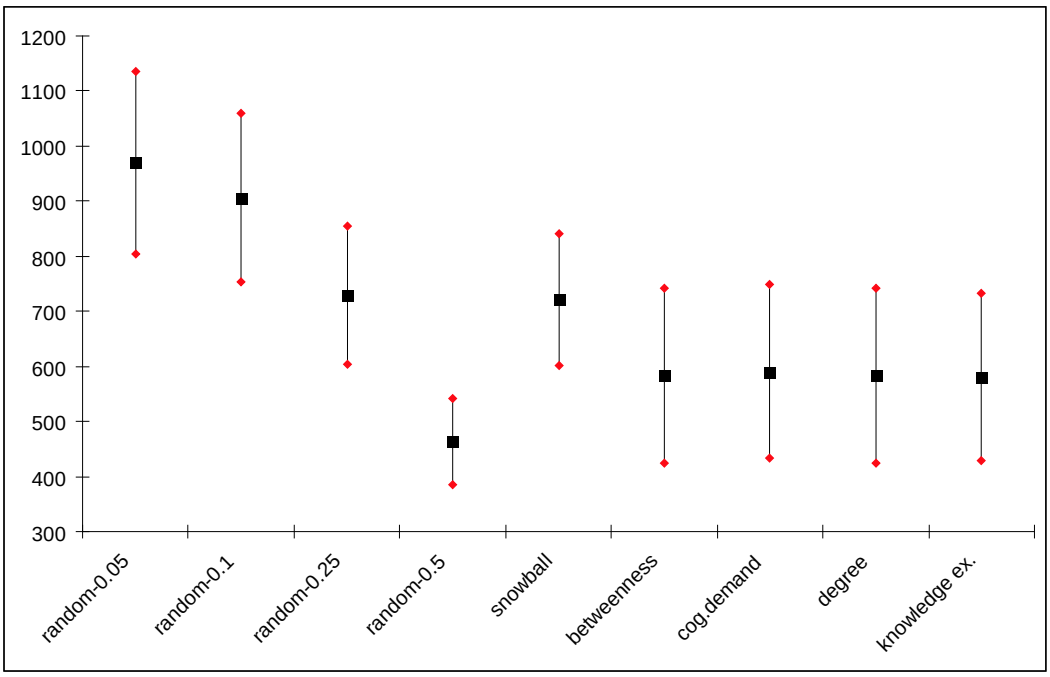


Figure 4.10: Mean Performance of Socially Intelligent Strategies (3x3 cells, 20 runs/cell)

In this particular case, the local maximum is located near Agents 92 and 93 - which together account for close to half of messages captured.

A further complication to the above problem is the fact that initially the Blue Team agents know very little about the Red Team — thus the accuracy of their estimations of centrality metrics is bound to be low[Borgatti, Carley, and Krackhardt, 2004],[Costenbader and Valente, 2003]. Therefore, the heuristic can fall into a local maximum within one or two time periods from the start.

While problems of local maxima are serious, they are not unsolvable. One of the best solutions for navigating parameter spaces with local maxima is to use an algorithm similar to simulated annealing with a measure of randomization in the beginning to serve as a bootstrapping mechanism. The next section describes such an algorithm, and shows its performance advantages over simple hill-climbing heuristics.

4.6 Socially Intelligent Traffic Sampling with Probabilistic Targeting

A notion of a *Taboo List* can be added to the algorithm to force it to sample agents that do not get top centrality measures.

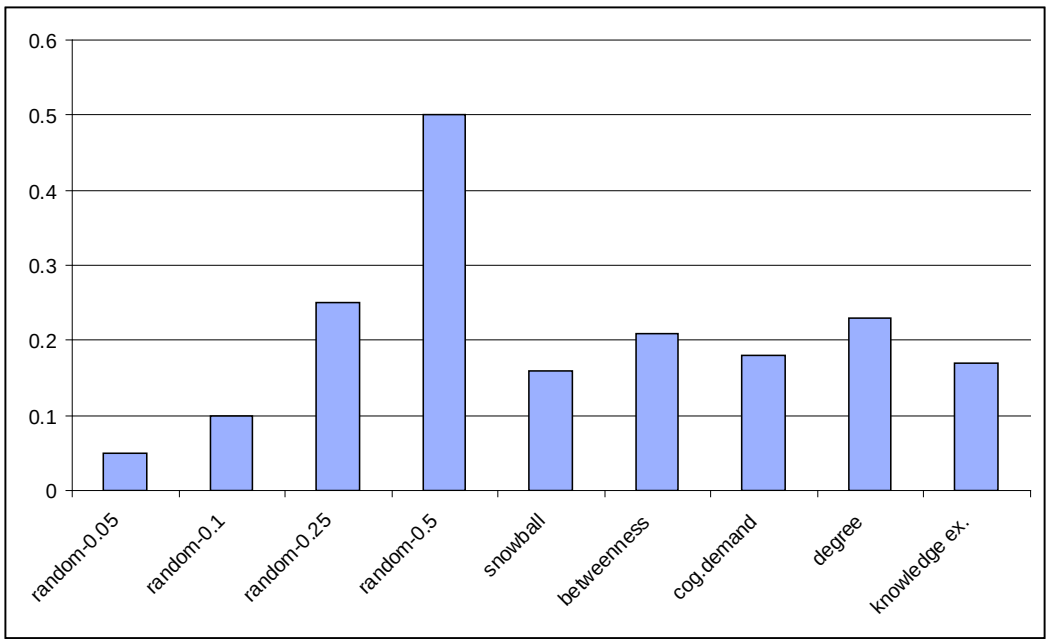


Figure 4.11: Mean Signal-Noise Ratio of Socially Intelligent Strategies (3x3 cells, 20 runs/cell)

However, I propose a more robust solution: a traffic sampling algorithm with probabilistic targeting. This algorithm allows the much greater coverage of the network and is less prone to finding local maxima. The algorithm maintains a set of nodes that comprise its region of interest (ROI). After random initialization and a period of traffic capture, the captured MetaMatrix is analyzed and a new ROI is constructed based on the results of this analysis. However, nodes to comprise the new ROI are not picked deterministically. Rather they are picked by an exponentially distributed random variable. The distribution is composed in a way such that the most prominent nodes (i.e. these highest in the measure of interest) have the highest probability of being included in the ROI - but there is non-zero probability of the least well-connected nodes included as well.

The algorithm is illustrated on figure 4.13 and listing 4.3.

The exponentially distributed random variable is initialized as follows:

$$P(x) = \lambda e^{-\lambda x}$$

Where parameter λ dictates the speed of fall-off of the probability distribution function. Thus, λ dictates how "adventurous" the algorithm would be in including little-known agents in the ROI.

Furthermore, manipulation of the λ parameter during the running of the algorithm results in behaviour similar to that of simulated annealing -

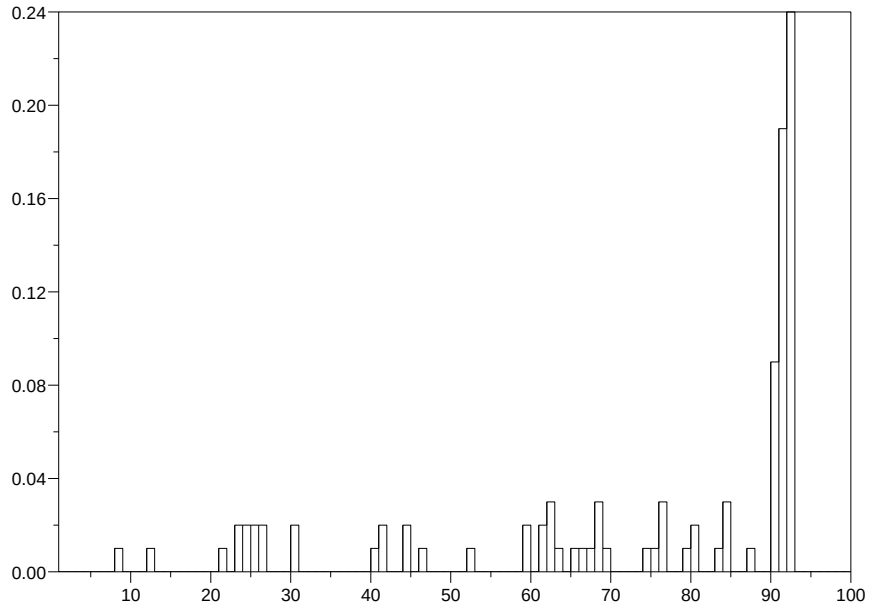


Figure 4.12: Histogram: frequency of capture of messages per agent; demonstrates adherence to local maximum in simple soc.int. heuristics (single run)

the ROI becomes more constrained as more information on the network is obtained.

Results of evaluation of this algorithm as well as a parameter sensitivity study are presented in the next section.

4.7 Intelligent Network Sampling Heuristics

In the previous experiment, I have shown that well-understood sampling mechanisms of snowball sampling and hill-climbing socially intelligent sampling outperform the baseline strategies but still perform sub-optimally. This sub-optimal performance is due to existence of local maxima and degrading accuracy of social network metrics in environments with incomplete knowledge of the network.

In this experiment, I use the probabilistic sampling algorithm described in the previous section in conjunction with a number of common social network analysis algorithms. These social network algorithms include *degree centrality*, *betweenness centrality* and *closeness centrality* [Freeman, 1979], and MetaMatrix metrics of *cognitive demand* (see section 2.6.1, [Carley and

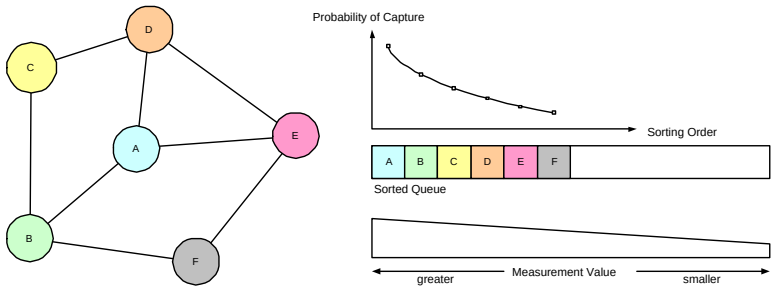


Figure 4.13: Socially Intelligent Traffic Sampling with Probabilistic Targeting

Listing 4.3: Socially Intelligent Sampling with Probabilistic Targeting

```
ROI = a small random set of nodes
Let Exp = Exponentially Distributed Random Variable (lambda)

REPEAT
 Capture traffic TO and FROM nodes in CurrentTargets
 Store captured messages in MetaMatrix accumulator

 Run Measure of choice on MetaMatrix
 Insert all nodes into an Array SORTED by value of Measures

 ROI = empty list
 FOR i = 0 to number of nodes
    Probability(i) = Exp(i)
    r = 0 < random number < 1
    IF (Probability(i) < r)
      ADD Array(i) to ROI
    END IF
 END FOR
FOREVER
```

Ren, 2001]) and knowledge exclusivity metrics (see section 2.6.1,[Ashworth and Carley, 2002],[Ashworth, 2003]).

I also test a heuristic targeted at achieving the maximum breadth of coverage of nodes on the network - perhaps at the expense of depth of knowledge or number of undiscovered edges.

Performance is evaluated on the basis of hamming distance between the learned network and true network and in terms of effective signal-to-noise ratio. I further introduce a wiretap effectiveness metric that embodies the *"get more from less"* philosophy by combining the two metrics to study incremental efficiency of each of the algorithms.

4.7.1 Experimental Design

The experiment follows a matrix design. The independent variables are initial network topology (uniform, scale-free and cellular) and wiretap heuristic. The table below defines experimental cells; each of the cells specifies a name of a data series that will be referred to in the observations and graphs.

	Uniform	Scale-Free	Cellular
Snowball	u-snowball	sf-snowball	c-snowball
Soc.Int: Degree Centrality	u-degree	sf-degree	c-degree
Soc.Int: Betweenness Centrality	u-btw	sf-btw	c-btw
Soc.Int: Cognitive Demand	u-cd	sf-cd	c-cd
Soc.Int: Knowledge Exclusivity	u-kex	sf-kex	c-kex

The size of networks is kept constant throughout the experiment:

Parameter	Value
Number of Agents	100
Total number of facts in knowledge network	200
Total number of available resources	20
Number of subtasks in task structure	20

The density of uniform random network is set at 0.2 - i.e. probability $P_{i,j}$ of an edge existing between agents i and j is 20%. The method for generating such networks is described in section 6.0.1

The scale-free network are grown using the Barabasi-Albert method of preferential attachment[Barabási and Albert, 1999], with parameters:

$$k = 0.25$$

$$\gamma = 2$$

More details on growing scale-free networks can be found in section 6.0.1

The cellular networks are generated using the mechanism outlined in section 6.0.1 using the following profile:

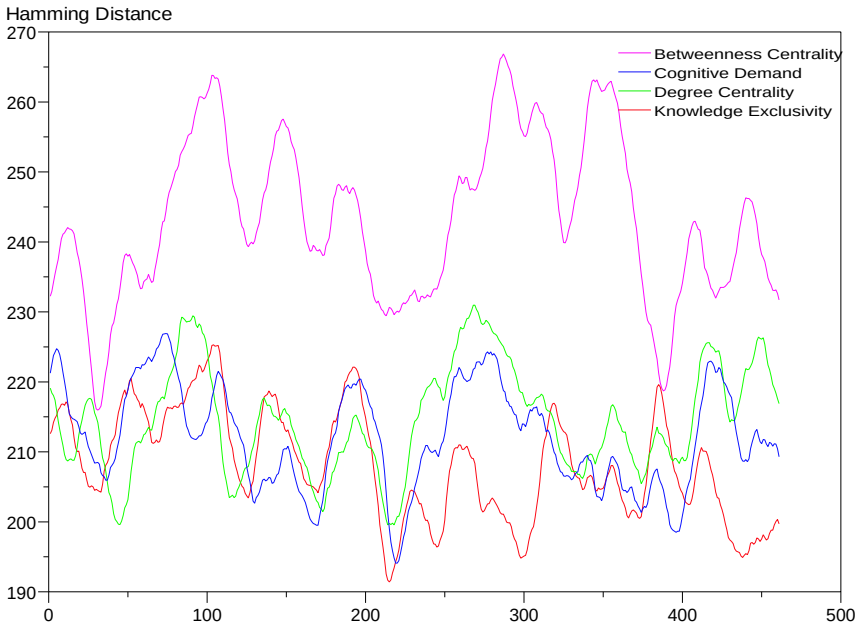


Figure 4.14: Performance of Probabilistic Sampling Socially Intelligent Strategies, cellular networks. (a)100 agents, single run

Parameter	Value
Mean Cell Size	6
Cell Size St. Deviation	1.7
Internal Cell Density	0.9
Probability of Random Connections	0.01
Density of cell leaders	0.16
Probability of connection between leaders	0.6
Probability of triad closure within cells	0.9
Probability of triad closure outside cells	0.17

4.7.2 Performance of Probabilistic Sampling

Figure 4.14 shows that annealing-based sampling strategy clearly outperforms the simpler SNA-based algorithms described in section 4.5. At signal-to-noise ratios similar to the simple strategies (figure 4.15) the annealing-based strategies achieve a mean hamming distance of about 50% of simple strategies.

Simulated Annealing algorithm as described above is a probabilistic method, where probability of capture varies depending on value of a cost function.

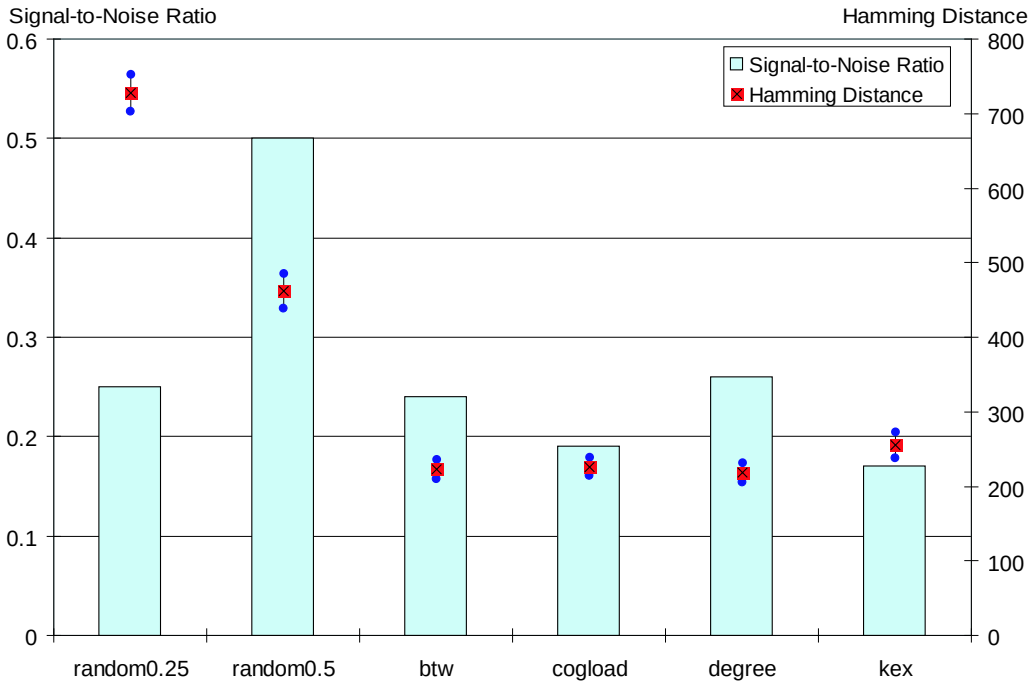


Figure 4.15: Performance of Probabilistic Sampling Socially Intelligent Strategies, cellular networks, mean of 100,200,500 agents, 20 runs per cell

This method achieves its low hamming distance metric by sampling more efficiently. Every agent that has been discovered has a chance of being sampled in a particular period as opposed to only agents with high levels of SNA metrics. The heuristic does not get stuck in local maxima and the randomization of search allows the heuristic to bootstrap itself efficiently. The level of randomization goes down slowly thus focusing the search and preventing waste of resources on agents that don't communicate a lot.

4.8 Estimating Cost-Effectiveness of Sampling Algorithms

In the course of experiments, I have observed that effective signal/noise ratio (i.e. the ratio of number of messages captured to the overall amount of traffic) is indicative of the cost of running a particular sampling algorithm. To elaborate, each captured message must be evaluated by the Blue agents and stored within its data structures. When a message is captured, and proves to provide duplicate data or data that is not within the current scope of interest, the processing time and storage space is wasted.

This leads me to a design of a combined incremental-cost metric that

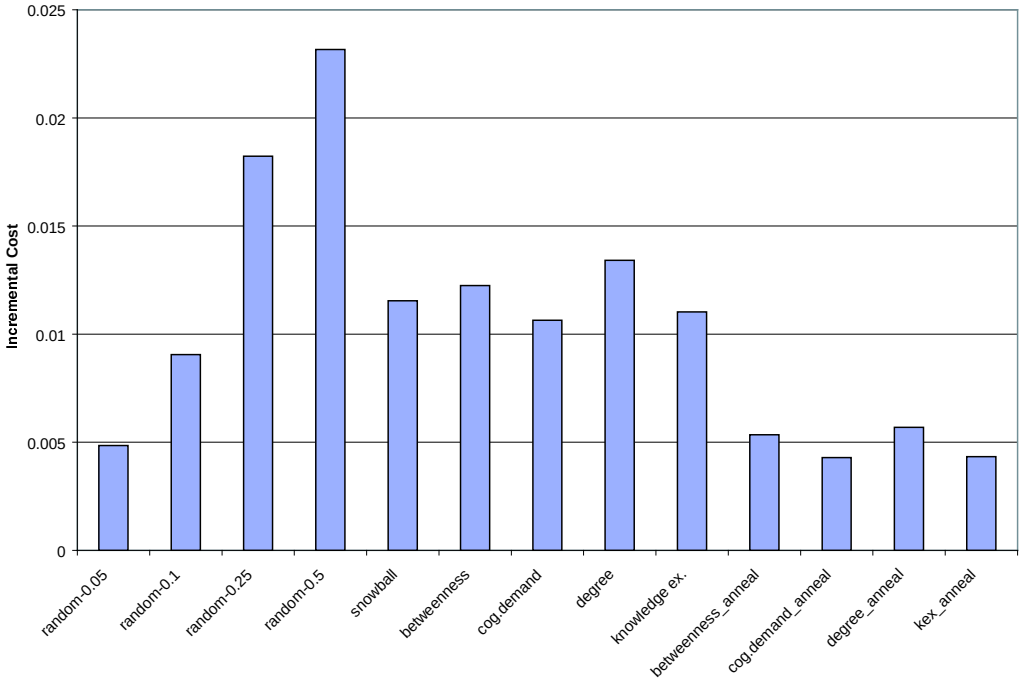


Figure 4.16: Incremental Costs of Sampling Strategies (mean/st.dev. of 180 runs)

evaluates every strategy and algorithm no only on the basis of its raw effect (i.e. the hamming distance produced as a result of running the algorithm), but also in terms of the cost of providing the needed level of accuracy. I have defined such incremental cost metric as:

$$CE = norm(D_{hamming}) * SNR = \frac{D_{hamming}}{n^2} * SNR$$

or a multiplication of a normalized hamming distance (e.g. hamming distance divided by the maximum number of edges in the graph) by the effective signal-to-noise ratio.

Values of CE should be interpreted as a cost of incremental improvements in the hamming distance, and thus lower values indicate better performance.

Figure 4.16 illustrates how different strategies reviewed in this document compare on the basis of cost effectiveness. The winners in terms of incremental cost are strategies based on simulated annealing. These algorithms have been designed from the start in terms of lowest incremental costs and "doing more with less", and thus produce a very low incremental cost value.

A non-obvious result on figure 4.16 describes a very low-fidelity random sampling strategy as having an incremental cost as advantageous as the most sophisticated algorithms. However, while the incremental cost is low, the

best possible performance from this strategy is also very low (see sec. 4.0.2) - which severely limits the usefulness of this technique.

4.9 How to Improve the Accuracy of Hamming Distance Metric

In experiments presented throughout this chapter, I have evaluated performance of intelligence gathering strategies in terms of Hamming distances between "True Network" - a network structure created by a perfect observer, and a "Learned Network" - a network structure created by a cognitively limited observer utilizing one of the many sampling strategies. Hamming distance provides a good estimate of accuracy of the mapping at a given point of time.

However, in context of dynamic networks this metric is limited due to the fact that it does not discriminate between Type I and Type II errors (i.e. *false negatives* and *false positives*), and provides no means for finding other possible error states. Meanwhile, a number of different error conditions occur, and each carries different significance in terms of the intelligence gathering capability. The classic error conditions are:

Error of Omission (Type I Error) : an edge that exists in the True Network is omitted from the Learned Network

False Positive (Type II Error) : an edge that does not exist in the True Network, but exists in the Learned Network

However, in a dynamic network, we can classify a number of other errors as well:

Late Removal : an edge that has been removed from the True Network due to its age (e.g. a connection that has not been activated for a long time), but has not been removed from the Learned Network.

The **Late Removal** error is commonly treated as a False Positive - as mechanism to detect this type of errors is unclear.

Incorrect Weighting : a mismatch in edge weights, in a valued network.

A future research scenario in evaluation of dynamic network data gathering techniques would include creation of ROC curves specifying the trade-off between false-positive and false-negative rates. As a first approach to the topic, I would like to propose a metric of estimating accuracy of network predictions that would be more descriptive then Hamming distance.

To define the alternative distance metric, let us first revisit the notion of edges and edge weights. In NetWatch, the edges of both True Network and

Learned Network should be viewed as a product of agent communication, rather then a separate entity. This is operationalized as follows: when communication between two agents i and j occurs at time t (and is observed), the edge $E_{i,j}$ is set to 1. At time $t+1$, the value of $E_{i,j}$ decays. The decay function in NetWatch is given as $E_{i,j} = \frac{1}{\alpha * \triangle t}$ where α is the decay rate and $\triangle t$ is the amount of time elapsed since last communication before i and j.

In short, the value of $E_{i,j}$ is not discrete - rather, it is continuous in the range of 0 to 1. Normally, if $E_{i,j}$ reached a value less then a threshold parameter $\epsilon \approx 0.05$, the edge was considered extinct and removed from the network.

To instantiate the new metric of network matching, the threshold operation should be excluded, in favour of treating all edge values as continuous variables.

Then, the metric can be written as follows. Let G_1 and G_2 be two graphs under comparison. Let us assume that both graphs contain the same collection of nodes. Let us also adopt a notation where E_1 is an edge in G_1 and E_2 is an edge in G_2:

$$NetDistance(G_1, G_2) = \sum_{i,j \in G_1, G_2} \begin{cases} E_1^{i,j} < E_2^{i,j} : w_{type1} * E_2^{i,j} - E_1^{i,j} \\ E_2^{i,j} < E_1^{i,j} : w_{type2} * E_1^{i,j} - E_2^{i,j} \end{cases}$$

where w_type1 and w_type2 are weights of Type I and Type II errors, respectively.

To restate, type I and type II errors are thus quantified as continuous functions and weighed separately. Setting of the weights then emphasizes priority of the evaluator to minimize false positives in a trade-off against decreasing the errors of omission. Furthermore, the error conditions of **Late Removal** and **Incorrect Weighting** thus become special cases of the existing error conditions and present a less severely penalized instances of Type I and Type II errors.

Due to treatment of errors as continuous variables, it also becomes possible replace constant weights of Type I and Type II errors with non-linear functions reflecting the true response of the experimenter to severity of errors. For example, if the weighting function of the error conditions are set to quadratic or higher degree polynomial functions, large errors will be severely penalized while smaller errors will receive lesser penalties.

While it was not possible to test this algorithm within the scope of this dissertation, it is a viable way to increase accuracy of error measurement in comparison of two networks and will be evaluated in future research.

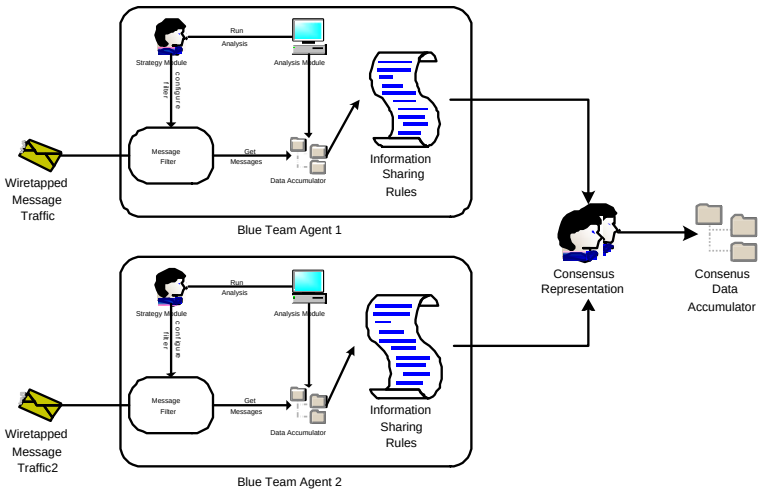


Figure 4.17: Information Sharing in the Blue Team

4.10 Summary

In this set of experiments (sections 4.2, 4.5, 4.7), I have discussed various algorithms and heuristics for learning a network representation of a set of communicating agents by observing their communication patterns.

As a main performance metric, I have used hamming distance. However, hamming distance alone does not capture the philosophy of maximizing effectiveness of network mapping or *doing more with less*. Thus, to estimate costs of running each of the strategies, I evaluated effective signal-to-noise ratios of Blue agents — ratios of messages captured to overall message traffic.

4.11 Consensus and Information Sharing in the Blue Team Network

While Blue Team agents collect and store their private representations of the observed Red Team network, it is also important to model how information sharing among intelligence agencies affects the accuracy of collected data.

NetWatch implements a capability (shown in figure 4.17) of modeling information sharing among Blue Team agents by means of information sharing rules and common databases for accumulating consensus knowledge.

The Blue Team agents implement information sharing by means of a filter object similar to the Message Filter described in section 4.0.2. The filter is configured using a set of rules that govern what information can be released and to whom:

Full Sharing: All data collected by the agent is shared,

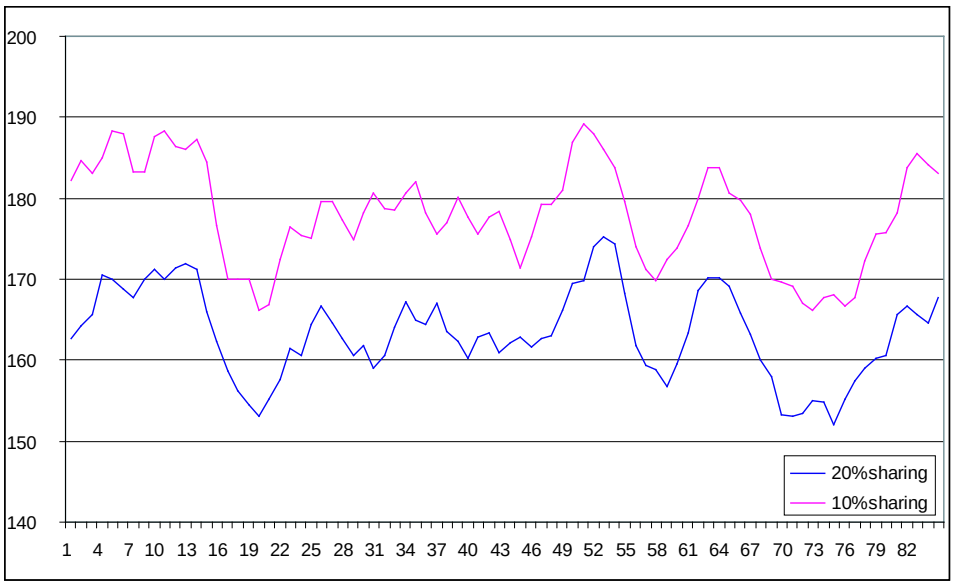


Figure 4.18: Performance of information sharing regimes

Recipient Filtering: Data can be shared only with a small set of agents

Content Filtering: Only data of certain type (e.g. person-to-person links) can be shared; data can be further filtered by attribute values.

Furhtermore, the data sharing rules can combined to form complex data sharing regimes, for example:

> Share only **Person-To-Person** data where **Source or Target is U.S. Citizen** with **FBI** or **ATF**
>
> Share only **Task** data with **Local Police**

Information sharing regimes enable modeling of complex intelligence and law enforcement community infrastructure as it pertains to gathering and analyzing information on covert and terrorist networks.

4.11.1 Preliminary Experiment in Information Sharing among Blue Agencies

In this experiment, I tested the effect of sharing some of the information obtained through wiretapping among four agents of the Blue Team. Each of the four agents used an annealing message capture strategy based on (a) degree centrality, (b) betweenness centrality, (c) cognitive demand and (d) knowledge exclusivity.

In the experiment, each of the agents shared between 10% and 20% of captured messages picked randomly among all captured messages and passed

them along to the shared data accumulator, which in turn was accessible to the rest of the group.

Figure 4.18 shows that amount of shared information only slightly affected the overall value of hamming distance. In comparison with performance of a single agent (see 4.7), the mean hamming distance dropped by ≈ 30 points, or 10%. However, information sharing drastically reduces volatility of the network predictions.

4.11.2 Future Studies

The main objective of future study of information sharing among intelligence agency is to realistically model the information sharing regimes that exist in the intelligence and law enforcement communities. While the facilities to implement such regime are already in place, data regarding information sharing rules among members of the intelligence community is difficult to obtain in public domain literature.

4.12 Network Destabilization Tactics

In this experiment, the Blue Team agents not only collect information about the Red Team but also attempt to influence the performance of the Red Team by finding vulnerabilities in the covert network and attacking them by isolating or terminating agents within the covert network.

However, in the real world isolation of an agent is very difficult. Arrests of individuals within the United States require significant legal proceedings before a court order can be obtained and isolations that span national borders are even more difficult because they require cooperation of a country where an individual is located. To simulate this in NetWatch, we have introduced significant cost incurred by a Blue Team agent every time it attempts to isolate an agent on the Red Team.

Removal of key nodes of a covert network, arguably, is the only effective mode of combating terrorist organizations. Thus, agents on the Blue Team must do a significant cost-benefit analysis before attempting isolation and be as precise as possible in locating the Red Team individual to be isolated.

4.12.1 Experimental Design

In this experiment, I simulated a set of 20 cellular networks allowing members of the Blue Team to isolate one of the agents within the Red Team network. Each network was used 4 times using different destabilization strategies. The four strategies examined are:

- no attacks on the Red Team.

- isolate a member of the Red Team at random.
- isolate the Red Team agent with the highest degree centrality.
- isolate the Red Team agent with the highest cognitive load.

The Blue Team attempted to isolate one agent after wiretapping the Red Team for 100 time periods.

The size of networks is kept constant throughout the experiment:

Parameter	Value
Number of Agents	100
Total number of facts in knowledge network	200
Total number of available resources	20
Number of subtasks in task structure	20

The density of uniform random network is set at 0.2 - i.e. probability $P_{i,j}$ of an edge existing between agents i and j is 20%. The method for generating such networks is described in section 6.0.1

The effectiveness of isolation strategy is measured as the difference between baseline performance of the Red Team (i.e. without any action by the Blue Team) and performance of the Red Team in presence of an anti-terrorist task force of the Blue Team.

One must note, however, that the networks in question are dynamic and there can not be an absolute and static performance metric for any of the teams outside the time series data.

4.12.2 Observations

		wiretapping strategies			
		Random	Snowball	Degree	Cog.Demand
attacks	Random	-5.4%	-13.4%	-3.9%	-18.3%
	Degree	-21.2%	-24.0%	-21.5%	-21.1%
	Cog.Demand	-5.7%	-11.0%	-13.5%	-3.0%

Table 4.1: Reduction in Organizational Performance of the Red Team due to Anti-Terrorist Activity

Table 4.1 presents results indicating the change in performance for each of the analyzed strategies. The performance of the Red Team is measured as a ratio of numbers of successfully completed tasks to the number of assigned tasks. Each cell in this table shows the percentage difference in performance from the 50 time periods prior to when the first agent is isolated and 50 time periods after the isolation. This shows the immediate impact of the various

destabilization strategies. Note, in general, that any of these strategies does lead to a performance reduction indicating that there has been some destabilization. Second, there is an interaction between the type of wire-tapping strategy and the type of destabilization strategy.

4.13 Emergent Network Recovery Behaviour

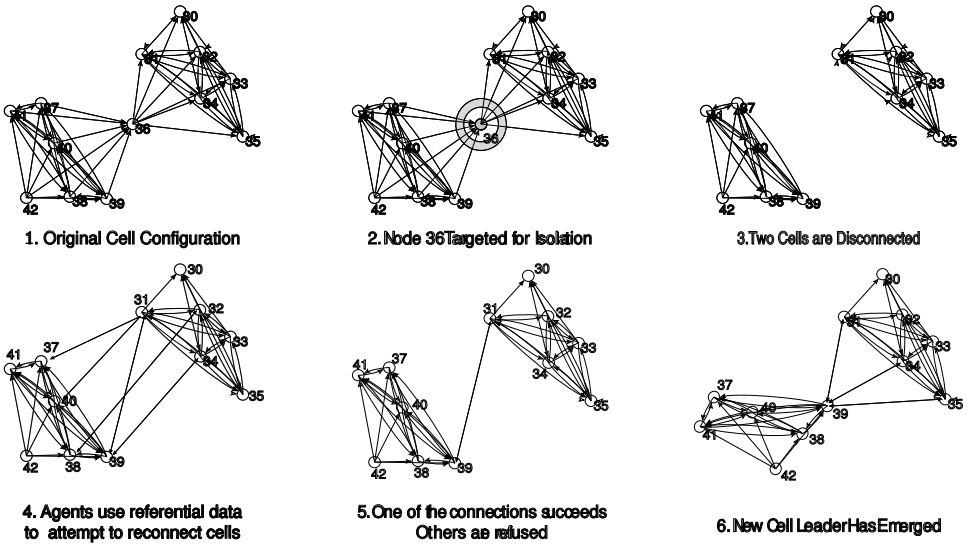


Figure 4.19: Recovery of a Cellular Network After Disconnection of a Gatekeeper Node

One of the notable results of this experiment has been the discovery of the mechanism that the network uses to recover after removal of one of its key members. Figure 4.19 demonstrates this process on a small cellular network

1. In the original configuration, the network consists of 2 fully interconnected cells and one gatekeeper agent. As the organization goes about its business, information flows from cell 1 (agents 30-35) to cell 2 (agents 37-42) through the gatekeeper agent 36. In process of passing knowledge and requests, small amount of referential data (such as "Agent X knows fact Y") are passed from one cell to another and stored by the agents. Relevancy or immediate usefulness of this information is low due to the fact that all needed information can be easily obtained from querying the gatekeeper agent.

2. Agent 36 is identified by the Blue Team as being important to the network due to its degree centrality, betweenness centrality and large amount of messages that it processes (cognitive demand).

3. The Blue Team proceeds to remove Agent 36 disconnecting the two cells. Due to the cellular structure of the organization, information transfer between cells is impossible and the performance of the organization is greatly degraded.

4. As information becomes unavailable from the central source, agents use the referential data accumulated in previous transactions to attempt to find information in the other cell. For the connection from Agent X to Agent Y to succeed, both agents have to have knowledge of each other. However, referential data is asymmetric and most of these connection attempts fail.

5. One of the connection attempts (Agent 31 to Agent 39) succeeds, thus opening a single pathway between cells

6. Referential information about Agent 39 spreads throughout the cell and more connections between agents in Cell 1 and Agent 39 are created. Within a short period of time, Agent 39 emerges as the new gatekeeper between the two cells.

   Information flows easily between the two cells and organizational performance is restored to levels similar to these before the removal of Agent 36.

A priority in research on destabilization of covert networks has been finding key individual's removal of which will separate a cellular network into subparts. However, the recovery process we demonstrate illustrates that even if the Blue Team achieves separation of the covert network into disconnected cells, the network will use its latent resources and quickly recover from damage.

The goal of network destabilization techniques should be to cause permanent damage to the covert network and not allow it to recover from the attack.

This finding also prompts us to not just study the effectiveness of removal of certain individuals but to look at the performance of these measures over time.

4.14 Scalability

NetWatch simulation is a complex software system. It consists of hundreds of independent processes, each storing large amounts of information and running multiple computations, planning and reasoning algorithms. Thus, to make the system useable in real-world scenarios, attention must be paid to scalability of all subsystems.

Low-level Subsystems Low-level subsystems of NetWatch consist of a custom lightweight threading library and a message passing subsystem.

The threading library (MThreads) is designed to be independent of the underlying operating system, and allows for a large number of simultaneous threads to reside within one system-level process. While MThreads was designed primarily to assure fully randomized time-slice allocation (which is required to run asynchronous agents within one process image) and to bypass kernel-level limitations on number of simultaneous threads, an important side effect of the design is its ultra-low overhead. The memory overhead of running a C++ object as MThread is $\approx$ 16 bytes, and processor overhead is limited to speed of two function calls - which can be considered negligible. Due to its independence of operating system peculiarities (e.g. Posix compliance), MThreads library has been compiled on all major operating systems. MThread library has been tested by running up to 500,000 simultaneous threads on a single-core Pentium IV processor.

The messaging subsystem can be thought of as a collection of queues. Each agent has two queues - its in-box and its out-box. The queues store pointers to messages, which are located in shared heap memory. As a result, no time or memory is wasted copying and sending around large message structures - a 4-byte pointer is all that is required to pass an entire message structure between two agents.

While appearing as XML to the outside world, internally the messages are stored as hash-tables. Thus, no extra parsing is required and access of data members inside messages is near-constant-time.

Agent Mental State The mental state of each agent consists of a Meta-Matrix structure, essentially a set of sparse graphs representing the social network, knowledge network and transactive memory of each agent. The sparse representation of the graphs is economical in terms of memory requirements, but presents a number of bottlenecks in the processing stages.

The single most computationally complex part of NetWatch is the computation of relative similarity and relative expertise. Both of these metrics require matrix multiplication, and have a computational complexity of $O(n^3)$. This computational complexity previously was a major factor limiting the maximum size of a system that could be simulated with NetWatch.

However, I have observed that each agent's individual network does not change very fast. Thus, recomputing the entire metric is not necessary at every time period. Only affected rows and columns are recomputed, thus reducing the average case complexity of the computation to $O(n^2)$.

The relative similarity/expertise calculations have to be repeated by every agent, every time period. Thus, I can estimate average computational complexity to $O(n^3 * t)$ where t is the number of time periods.

Agent Planning The hierarchical decomposition planner utilized by NetWatch agents is fairly efficient. Its average-case computational complexity can be estimated as

$$O(log(t) * (\frac{1}{\sigma_k} * k + \frac{1}{\sigma_r} * r))$$

where t is the number of tasks, k is the number of facts in the knowledge structure, r is the number of resources in the system, and σ_k and σ_r are densities of knowledge and resource networks, respectively. The complexity of planning grows as the density of constituent networks falls, as it becomes increasingly more difficult to for agents to locate and acquire needed resources.

It should be noted that planning is not done in one step, but is divided into iterations. Each iteration of the planner can do one of the following:

- Execute an atomic task
- Seek and obtain one fact or resource
- Do one round of decomposition.

Thus, the planning process is distributed in time and occurs simultaneously with other processes, including socializing ("Chatter") and execution of classification tasks. Complexity of a single iteration of planning thus is negligible, in comparison with SNA metrics discussed above.

NetWatch is fairly efficient in terms of usage of computer resources and time, given the complexities involved. The real and projected timings, per time period, can be found in figure 4.20.

4.15 Summary

The results presented here show the potential power of multi-agent network simulations for addressing real world issues. Agent-based simulation frameworks enable a more realistic and extendable architecture for addressing policy issues in a manner comparable to human behavior. Such work moves us from the realm of building agents to act more or less independently on

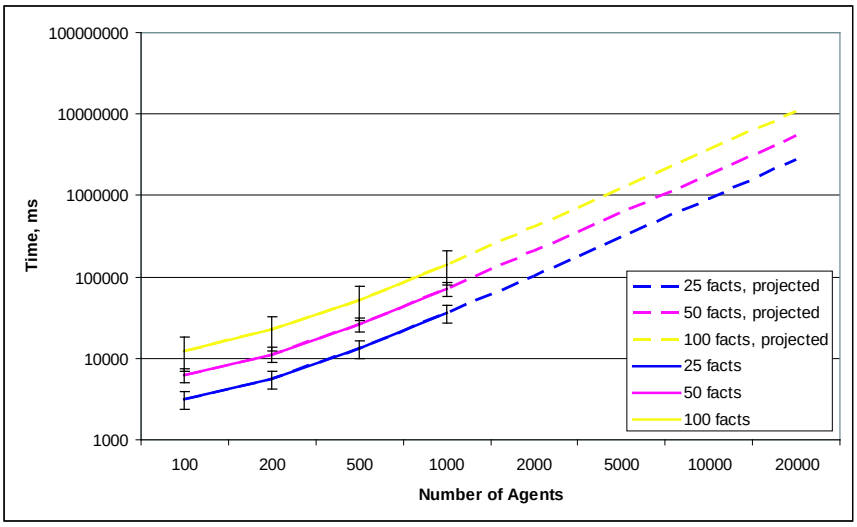


Figure 4.20: Measured and Projected timings of NetWatch runs on a 3GHz Pentium IV processor

behalf of people to the realm of using collections of agents to reason about how people as groups behave.

The advantage of an agent approach is that it enables the simulated actors to behave like humans — they are cognitively and socially bounded with knowledge of themselves and others dependent on their personal history. When such agents are embedded in dynamically evolving networks, the entire simulated system takes on the social and technological constraints consistent with empirical findings. The advantage of using AI planning, reasoning and decision making techniques is that complex intelligent agents are extensible to multiple tasks and scenarios.

Such models enable the researcher to examine the nature, not just of cognition but of social cognition, and to explore policy and managerial issues. In doing so, the goal is not to "predict" specific events, but to decrease uncertainty in detecting trends. As such, these tools are valuable assets to decision makers.

4.15.1 Objects of Further Study

Additional work planned on NetWatch includes a number of new experiments using the existing system as well as additional model components to enhance realism of NetWatch models.

Efficacy of Agent Isolation

The literature on network-centric warfare frequently concentrates on destabilization and dismantling of terrorist networks via isolation (arrest or liquidation) of key agents. However, history of targeted elimination raids conducted against Hamas[Wimisberg, 2004] shows that this tactics may not always be effective, and their costs (human, financial and political) may not always justify the achieved results. Moreover, history of operations against high-level operatives shows that elimination of one key actor is generally not enough.

Future studies involving NetWatch should consider alternate strategies of destabilization of terrorist operations, including cutting of resource pathways and financial channels, disrupting training and recruiting of new members. Furthermore, studies dealing with agent elimination should concentrate on locating and isolating entire groups of actors that engage in planning of attacks, or are close to executing an attack.

It has been observed that information warfare attacks on certain key actors in terrorist networks may be as effective as physical attack on the same[Alberts, Garstka, and Stein, 1999]. Thus, a model of information warfare included as part of NetWatch would be a useful tool for researching effects of information warfare on growth and effectiveness of terrorist organizations.

Second-Order Effects of Anti-Terrorism Policies

In the simulated world, the agents of the Red Team learn, obtain knowledge and resources, and execute tasks. Strategies applied against them by the Blue Team will succeed or not succeed over time, and thus hypotheses will be accepted or rejected. The real world is much more complex. Any action taken in an adversarial world is bound to force a counter-action from the the opponent. If strategy of the Blue Team (e.g. its use of simulated annealing) is known to the Red Team, in the real world this would result in change of strategies on Red Team's behalf that would diminish the utility of the Blue Team. In an adversarial situation, this cycle of actions and reactions can occur indefinitely, prompting innovation and adaptation on behalf of both adversaries.

While full study of second-order adaptation is beyond the scope of this thesis, a number of models addressing this very problem have been proposed. Dalvi and Domingos[Dalvi, Domingos, Mausam, Sanghai, and Verma, 2004] have addressed the notion of second-order adaptation in the realm of spam detection and classification. They have reduced the adaptation cycle to a game-theoretic construct, where optimization of adversarial strategy allows the "good guys" to increase quality of their spam classification and automatically adapt the classifier to to the adversary's evolving manipulations. In

the context of terrorist network, a similar problem has been approached by Jensen[Jensen, Rattigan, and Blau, 2001].

Additional Experiments

Response to Annealing Parameters The simulated annealing algorithm presented in section 4.6 produces results that are significantly improved in comparison to those of simpler algorithms. However, tuning parameters of a simulated annealer still remains somewhat of a black art. To fully understand the response of the system to change in annealing parameters an additional set of experiments needs to be performed.

Configuration Space Exploration NetWatch simulation accepts a large number parameters as its input. While experiments outlined in this dissertation explore response to parameters most important in the simulation domain, the full configuration space of NetWatch is largely unexplored. However, the configuration space of a complex multi-agent simulation is very large and full exploration may not be feasible due to high dimensionality of the configuration space. Using a tool such as WIZER[Yahja, 2003], a multi-dimensional configuration space can be explored using intelligent heuristics, which should cut down significantly on the computational cost of exploring the configuration space.

Model Enhancement

A number of additional features have been planned for NetWatch:

Human Intelligence NetWatch at this point solely focuses on gathering and processing of signal intelligence. However, human intelligence (HUMINT) is generally considered a much more valuable and reliable source of information on terrorist networks. Human Intelligence models would include recruitment of double agents from within the Red Team and infiltration of clandestine agents from the outside. The clandestine agents would have to possess a very different set rules governing their actions and may be difficult to model reliably without resorting to either classified data or paperback fiction.

Recruitment Model Terrorist organizations are not static. Their memberships grow and wane depending on many factors ranging from political situation to design of the educational system to economic conditions. Building a realistic model of terrorist recruitment would present NetWatch with a number of challenges, including ways to monitor who gets recruited, where

and by whom. The Blue Team will be further challenged to keep its representation of the Red Team synchronized with its growth.

Part II

Symbolic Reasoning about Social Structure

Chapter 5

Reasoning about social networks: a robust semantic language

> We used to think that if we knew one, we knew two, because one and one are two. We are finding that we must learn a great deal more about 'and.'
>
> Sir Arthur Eddington
> (1882-1944)

5.1 Introduction

The study of complex social and technological systems, such as organizations, requires a sophisticated approach that accounts for the underlying psychological and sociological principles, communication patterns and the technologies within these systems.

Since inception, Social Network Analysis and link analysis have operated on the cutting edge of bringing together mathematical analysis of social structures and qualitative reasoning and interpretation.

However, one facet of research methodology has largely remained unaddressed. Most mathematical analysis and social simulation tools operate on abstract numerical representations of social structures, such as graphs, matrices and time series. However, the concrete semantics behind these numbers frequently was only a part of the researcher's mental model. Its communication to the outside world was largely a function of the researcher's writing skills. This and the level of abstraction required by early computer models, have resulted in datasets and models that are very difficult to interpret, especially by non-specialists.

This chapter examines the past approaches of creating interpretable and semantically consistent models of social structure and social networks, as well as social simulation tools. I further propose the creation of a robust and scalable social structure semantic that facilitates interpretable reasoning about evolution of social structure.

The chief advantage of taking a semantic approach to reasoning about social networks is the ability to consistently describe the interactions of nodes and edges in multi-modal and multi-plex networks. Since the differences of node edge types in such networks are semantic in nature, the semantic encoding of the network structure allows the user to resolve combinatorial closures across edge types, and decompose complex interactions into sets or sequences of simpler ones.

For example, a long-standing problem in the field of social network analysis concerns integration of data in a dual-mode network, incorporating friendship and advice links. At this point, the notion of centrality in such a network is undefined both mathematically and semantically. However, by decomposing the notions of friendship and advice into more basic semantic notions of information transfer, affinity, respect and authority, a NetInference ontology can be constructed to analyze the dual-mode network as a unit.

This chapter is organized as follows: after discussion of issues and shortfalls of several graph-based models of social interactions, I proceed to construct a semantic of social interaction starting from the basic constructs of the human language (section 5.4), and review appropriate techniques borrowed from the fields of expert systems (sec. 5.8) and object-oriented modeling (sec. 5.9).

In section 5.12, I discuss the design of the language for specifying ontologies for social structure data. I further discuss representation of graph structures (sec. 5.14), design and execution of graph rules (sec. 5.15). The devices presented to this point are sufficient to address the issue of integration of friendship and advice networks, which is discussed in detail in section 5.16.

Section 5.17 introduces a facility for querying and manipulating graphs as a database, and builds upon this capability to recreate a set of traditional SNA metrics through the language of NetInference (sec. 5.18). Finally, I present an example of a complex ontology for reasoning about terrorist networks utilizing all capabilities of NetInference to facilitate question answering and inference of edges in a semantic context.

5.2 Issues in Representation of Social Network Knowledge

Traditional social network analysis operates on a simple set of concepts: nodes of a social network are a homogenous set of people or groups of people and links between nodes represent connections or relationships between these people. Semantically, the existence of an edge signifies that a relationship of some sort exists between two constituent nodes:

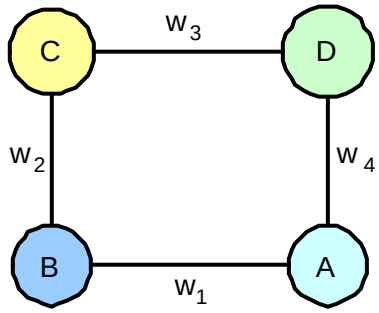


Figure 5.1: Simple Social Network

$$G = \begin{cases} \{A, B, C, D\} \in People \\ w_{1..n} = ConsistentWeightMetric \end{cases} \tag{5.1}$$

In a weighted network (figure 5.1), the weight of an edge takes a variety of semantics, including frequency of communication, distance between nodes or closeness of relationships.

$$w_i = \begin{cases} \{0, 1\}\ in\ a\ binary\ network \\ \in \mathbb{R}\ in\ a\ weighted\ network \end{cases} \tag{5.2}$$

While weight w_i can be any real number, in hand-collected social network data[Krackhardt, 2003] these values are often limited to a 5-point scale, rating the strength of a relationship between two nodes:

$$w_i = \begin{cases} 1: & A\ has\ never\ met\ B \\ 2: & A\ and\ B\ have\ met\ once \\ 3: & A\ and\ B\ meet\ occasionally \\ 4: & A\ and\ B\ meet\ regularly \\ 5: & A\ and\ B\ are\ close\ friends \end{cases} \tag{5.3}$$

Thus, computing simple graph-theoretic measures upon the resultant graph produces interpretable results that allow detection of powerful or important nodes, communication gatekeepers, etc.

It is important to note that the semantics of edge weights must be firmly set before any data collection efforts are undertaken and kept consistent throughout the entire life cycle of the resultant dataset. This presents few problems if the constituent population is fairly small and the study itself is restricted in time. However, growth of the scale of the study makes it significantly more difficult to maintain this consistency.

While the simple and consistent semantics of binary and homogenous valued edges allows for fairly easy data collection, it does not capture the full

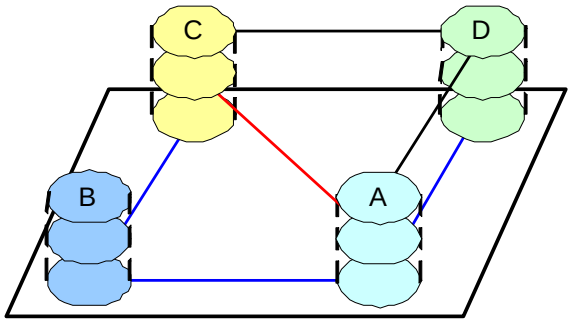


Figure 5.2: Stacked Graphs Metaphor

richness of human experience. The fact that real-world social networks do not generally fit into a mold of a valued graph was acknowledged by Krackhardt[Krackhardt, 1990] in studying overlaps and connections that cross the boundary between different person-to-person networks (i.e. friendship, advice, money-lending, subordination, etc).

We can view analysis of multiple overlapping networks through a metaphor of stacked graphs (figure 5.2). A stacked graph metaphor consists of a single, consistent set of nodes N, connected with multiple sets of edges E_i. Each of the edge sets represents a semantically distinct notion of an edge (e.g. $friendship$, $advice$, or $money - lending$) and (in hand-collection methods) is collected with a separate set of questions on the questionnaire.

Cross-layer distance between actors can be calculated by three approaches. In a cross-layer graph traversal, the stacked graphs are viewed as a single n-dimensional adjacency matrix, which allows one to use any standard graph-traversal algorithm to find the shortest path between two actors, even if such path crosses boundaries of a single layer[Breiger and Mohr, 2004] [White, Boorman, and Breiger, 1976]. In this case, a hop between two layers of a stacked graph is considered a costless operation. However, that may not be the case in every situation. For example, crossing the boundary between professional and personal network layers may entail conflict of interest - in which case such a path may be shortest in distance but not lowest-cost.

Another approach to analyzing stacked graphs aggregates the stacked graphs into a single graph via weighted edge summation - at which point any standard SNA algorithm can be applied. However, values of individual edges will be lost, as they are impossible to infer from the resulting dataset. Thus, when a lowest-cost path is found through a summed graph, it may not be possible to recreate such a path in a traversal of the stacked networks.

Finally, one can define distances across multiple layers in terms of Euclidean geometry. This approach is useful for calculating closeness of actors but does not result in sets of useable paths for a graph traversal.

Unfortunately, the notion of centrality is undefined for stacked graphs.

	Person	Knowledge	Task
Person	Person *interactsWith* Person	Person *knows* Fact	Person *participatesIn* Task
Knowledge		Fact *relatedTo* Fact	Fact *requiredFor* Task
Task			Task *precedes* Task

Table 5.1: Semantics of MetaMatrix Edges

Given the fact that centrality is a major component of many social network analysis studies, this is a major detriment to use of a stacked-graph model.

Due to purely operational difficulties of analyzing a stacked-graph dataset with more then a small number of layers, this approach only allows for analysis of a limited number of datasets.

5.3 Expansion of SNA Node Types

The stacked-graph approach to SNA presumes that there is a multitude of edge types — but only one type of node (i.e. a person). However, the real world networks consist of number of actors of different types — e.g., people, organizations, resources, information, events.

Krackhardt and Carley [Krackhardt and Carley, 1998] proposed concentrating knowledge about an organization in a format that could be analyzed using an expansion of standard network methods called the MetaMatrix (described in detail in section 2.6). The MetaMatrix expanded the notion of a node in a network to include a large number of possible entities. In its original shape, the MetaMatrix encompassed nodes of types $Person$, $Knowledge$, and $Task$ and introduced the concept of semantically loaded edges.

A semantically loaded edge such as those used in the MetaMatrix do not merely imply that a connection exists between two nodes. Depending on the types of source and target nodes, each edge carried a meaning. For example, an edge between $Person_i$ and $Knowledge_j$ carries the implied semantics of $Person \rightarrow_{KNOWS} \rightarrow Fact_j$. A full MetaMatrix semantics is illustrated in table 5.1.

At a later time, the MetaMatrix semantics has been expanded to accommodate more node types including $Resource$ and $Organization$, and corresponding edge semantics.

However, with the expansion of the number of node types that take part in the analysis, the concept of a node or an edge increasingly becomes overloaded with a plethora of meanings. For example: an edge between nodes $A_3 \in Agent$ and $O_1 \in Organization$ has an ambiguous meaning of "agent is

connected to organization" or "agent is a part of the organization". In this particular case, it becomes impossible to distinguish between a customer and an employee of an organization - which are significantly different relationships.

Of even more importance is a semantic ambiguity of edge direction, i.e. - does it mean the same thing if an *Agent* is connected to an *Organization* or an *Organization* is connected to an *Agent*?

Since the MetaMatrix contains many sub-matrices with heterogeneous node types (e.g. $Person \rightarrow Task$) and the MetaMatrix model traditionally only includes edges in the Upper Triangular portion of the matrix, the directionality of heterogeneous edges was either lost completely or potentially misinterpreted (e.g. does reversal of an edge's direction change the meaning of an edge when the source and target are of a different node type?).

A further problem with the ever-expanding MetaMatrix is the fact that the expansion process does not scale well. Let us suppose that a MetaMatrix includes N types of nodes; expansion of the model to deal with $N+1$ types of nodes will require definition of meaning and measures upon $(N+1)^2 - N^2 = 2N+1$ sub-matrices, which places an unnecessary burden upon the implementer.

The expansion trend has resulted in the creation of *semantic holes* - areas in datasets where an edge may exist but there is no adequate explanation for the meaning or any measures that may take that edge into account.

We also cannot overlook the fact that some edges have different properties than others. For example, subordination edges ("isSuperiorTo") for people or inclusion edges ("isPartOf") for organizations (as well as a number of others) are transitive — i.e., the boss of my boss is also my boss. A number of other special properties can be defined as well. Graph models to date do not offer sufficient reasoning capabilities to resolve transitive closure of edges, especially in a context where nodes and edges of multiple types coexist.

As a result, graph-theoretic analysis of the resultant networks is no longer a sufficient means of reasoning about social structure and must be supplemented with exogenous domain knowledge in order to derive meaning behind numeric measures. However, the graph semantics does not readily allow for machine-interpretable encoding of such domain knowledge.

The purpose of this chapter is to offer an alternative to graph-based models of social networks. This alternative structure offers an increase in expressive power over stacked-graph and MetaMatrix network models, enables automated reasoning and inference of network properties. In the same time, it is backward-compatible with existing SNA models and thus allows cross-validation using well-researched datasets.

5.4 Mapping Structures of Meaning

By structure-mapping, DiMaggio[DiMaggio, 1998] refers to as "the existence of some form of content-related domain specificity". Structure exists not only as sets of ties between actors but as networks among cognitive and cultural entities and study of these entities by means of network analysis is just as important as study of interpersonal relations.

Carley[Carley, 1994] uses network analysis techniques to map the structure of relations between conceptual items used in science fiction narratives and maps these structural representations of meaning to compare cultural phenomena through times. Mohr[Mohr, 1998] elaborates a framework for uncovering semantic structures that emphasizes relations among lexical and semantic terms in a classification system and application of formal network analysis models or pattern matching techniques.

The implications of these lines of research is creation of a link between network analysis and "understanding of the relationship between culture and social structure built upon careful integration of micro and macro, and of cognitive and material perspectives"[DiMaggio, 1998].

The further implication of this work is that semantically sound network analysis of human relations has entered a phase of maturity where sets of well-known techniques can be applied to reason on not only the structure implied in the network datasets, but also on *the meaning of this structure*[Breiger, 2004].

Further in this chapter, I introduce a concept of qualitative machine reasoning on social structures and show how integration of semantic reasoning with network analysis produce a powerful combination to create a new understanding of interplay of cultural and societal concepts.

5.5 Networks and English Grammar

To begin the process of definition of a regular ontology for expressing social network knowledge, let us first look a how interpersonal relationships might be described in English:

Alice **likes** *Bob very much.*

Bob and *Carol* **study together**.

Carol **fights** with *Alice* in school.

The high-school love triangle described above could be a classic example of the way in which social network representation of relational knowledge fails to address the complexities of human relationships. It is impossible to represent the three relationships within one graph - and it would be impossible to trace the love triangle if they were represented in a stacked-graph model.

However, humans can readily express and understand such relationships with the aid of language.

Let us then de-construct the above phrases. The phrase

Alice **likes** *Bob very much*.

consists of:

- Two nouns (*Alice* and *Bob*) representing the actors - or nodes of a social network
- A verb (**likes**) signifying (a) the existence of an edge between *Alice* and *Bob* and (b) the semantics of this edge.
- A value qualifier (*very much*) that adds emphasis to the verb

Thus, we can view a social network as a collection of English phrases consisting of nouns, verbs and adjectives and describing the relationships within such a network.

To elaborate, *Alice* **likes** *Bob* can be thought of as a representation of a directed edge $Alice \rightarrow_{likes} Bob$. *Bob* and $Carol$ **study together** would represent an undirected edge $Bob \leftarrow_{studiesWith} \rightarrow Carol$. $Carol \leftarrow_{studiesWith} \rightarrow$ *Alice* has a different implication - it is an edge that denotes an active conflict. Encoding the $fightsWith$ edge as a negative value is one way to do so — but then there is no way to distinguish between a passive dislike and an active act of conflict. Edges like these need to be addressed on a set-by-set basis, as there is not universal semantics that would adequately describe negative edges.

Further elaborations would allow complex qualifying statements, attribute strings (adjectives), time and precedence constraints, and so on, up to the full expressiveness of English or any other human language.

However, full machine understanding on human languages is an unsolved (and possibly unsolvable) problem. Thus, let us only use English grammar as a trampoline towards mapping semantics of human relationships via a machine-understandable mechanism.

5.6 Taxonomies and Social Networks

Perhaps, a combination of linguistic constructs (i.e. nouns and verbs) and graph-theoretic analysis would present a sufficient boost to expressiveness of social network data? A number of approaches have been proposed to address this question.

For example, RELATIONSHIP[Ian Davis, 2004] is an RDF[Eric Miller, 2004] schema that defines a vocabulary for describing social interactions and

friendOf	acquaintanceOf	parentOf	siblingOf
childOf	grandchildOf	spouseOf	enemyOf
antagonistOf	ambivalentOf	lostContactWith	knowsOf
wouldLikeToKnow	knowsInPassing	knowsByReputation	closeFriendOf
hasMet	worksWith	colleagueOf	collaboratesWit
employerOf	employedBy	mentorOf	apprenticeTo
livesWith	neighborOf	grandparentOf	lifePartnerOf
engagedTo	ancestorOf	descendantOf	participantIn

Figure 5.3: Vocabulary of the RELATIONSHIP taxonomy

relationships between people (see figure 5.3). However, definition of a vocabulary falls short of rigorously specified social relationship semantics.

While a vocabulary set can be negotiated and agreed to by a community of researchers, it will remain incomplete as the richness of human relationships presents more nuances that is possible to express in a finite vocabulary.

However, a more serious complication of a purely vocabulary-based specification of relationships is that a social network defined using this vocabulary is merely a labelled graph. While such graphs are widely used to communicate relationship information to human users, it is not possible for computers to reason about such labelled graphs without an understanding of natural language.

Thus, a further extension of language-based paradigm for expression of social structure data needs to be developed.

5.6.1 A Grammar for Expressing Relationships Between Entities

Use of a regular grammar provides us with a mechanism for expressing complex language-based concepts in a machine-understandable way. At first glance, such a regular grammar might be defined as:

```
network := (relationship)*
relationship := (relationshipVerb, fromNode, toNode)
relationshipVerb isA ontologyVerb
ontologyVerb := (verb, (attributeName, attributeValue)*)
fromNode, toNode isA ontologyNoun
ontologyNoun := (noun, (attributeName, attributeValue)*)
```

Thus, a network is defined as a collection of relationships which consist of two nodes (represented by nouns) and a verb. The concept of **isA** specifies that an entity is an instance of another entity or class of entities already present in the ontology - thus allowing inheritance of properties from entity to entity.

Shapiro[Shapiro, 1982] has published a grammar-based means for interpretation of semantic networks. The methodology uses Augmented Transition Networks (ATNs)[A.Woods, 1970] to interpret a tree or a graph of the semantic network and transform it into natural language sentences.

The advantages of the grammar notation have been summarized as "(1) clearness and perspicuity, (2) generative powers, (3) efficiency of representation, (4) ability to capture representation regularities and generalizations, and (5) efficiency of operation"[Shapiro, 1982]. ATN Grammars have been used as both parsing and generative tools for machine understanding in relational contexts[Woods, 1980].

5.7 From Grammars to Object Systems

A regular grammar for specification of network data is necessary but by no means sufficient for reasoning about social structure. While the grammar provides a structured framework for specifying entities and relationships between them, it does nothing to specify the meaning of these relationships - as well as rules or procedures that affect these relationships.

This mechanism is provided by the isA construct specified as a part of the grammar above. isA allows a simple yet robust inheritance of object properties and enables construction of robust hierarchical taxonomies of social structure objects.

The notions of inheritance and object orientation first appeared in Simula[Dahl and Nygaard, 1966]. In Simula, objects are grouped into classes and classes can be organized into a subclass hierarchy and elements of a class can appear wherever elements of the respective superclasses are expected — thus implementing notions of inheritance and polymorphism.

Cardelli[Cardelli, 1988] formalized the semantics of inheritance and multiple inheritance by building a grammar structure that united the requirements of building a strong typing system with requirements of keeping a consistent semantics that allows object orientation and inheritance on all levels, including ability to nest objects and compute polymorphic operations on sets of objects.

NetInference derives its hierarchical inference model from object-oriented semantics while superimposing them with reasoning mechanisms of symbolic reasoning.

5.7.1 Object-oriented Semantic Networks and Social Network Data

The main idea behind semantic networks is that the meaning of a concept comes from the ways in which it is connected to other concepts. In a semantic

net, information is represented as a set of nodes connected to each other by a set of labelled arcs which represent relationships between nodes.

This view is consistent with the views of traditional social network analysis, i.e., the idea that attributes of an individual actor are less important then the structural properties of the graph structure that the actor is embedded in.

However, semantic networks do not reduce the network structure to a binary graph - instead edge values matter as they convey the semantics of relationships between nodes. Semantic networks only treat structures as labelled graphs and do not allow consistent derivation of more complex node and relation types from basic types. Further, semantic networks are built with a top-down view of the network and thus do not allow for mixing of heterogenous edge semantics within the same network.

5.8 Ontology Languages and Social Networks

An ontology is an explicit specification of a body of formally represented knowledge[Gruber, 1993]: the objects and concepts in an area of interest and relationships that hold them together. The term is borrowed from philosophy, where an ontology is a systematic account of Existence. The ontology is defined on a finite domain by defining a set of representational terms. These definitions include entities in the subject of discourse (e.g., classes, relations, functions or other objects) and the formal axioms and rules that constrain the interpretation and well-formed use of data.

5.8.1 DAML and OWL

At this point of time, only a few ontological tools are developed explicitly for representing and reasoning with social relational data. Semantic Web efforts such as DARPA Agent Modeling Language(DAML)[Ankolekar, Burstein, Hobbs, Lassila, Martin, McIlraith, Narayanan, Paolucci, Payne, Sycara, and Zeng, 2001] are intended for representation of network resources and their interdependencies as well as organizational structures. Each node in DAML representation is an active agent, human or computational, and DAML records are used to locate and reason about resources needed for planning and accomplishing a task. However, DAML is designed for operations centered at vertices of the network, and is poorly suited for analysis of network topologies and rules that govern communication among nodes from a top-level perspective.

OWL is a general-purpose language for expression of ontologies. While not designed specifically for expression of social network-related knowledge, it is general enough to be adapted to express some of the information in

the social network context. Deaton[Deaton, Shepard, Klein, Mayans, Summers, Brusseau, Witbrock, and Lenat, 2005] has adapted OWL for use as an interchange language between end-user knowledge acquisition tools, analysis tools and Cyc - a large repository of knowledge and reasoning system. However, OWL does not have a ready facility for expression of executable rules and is not Turing-complete, thus having inherent limitations in expression of object-oriented ontologies. Further, in Deaton's adaptation of OWL, relationships between nodes are not first-class objects but methods that connect objects together. Expression of semantics of edges and nodes are done through significantly different mechanisms and thus results in an awkward implementation.

5.8.2 Cyc

CYC corporation[Lenat, 1995] is in the midst of an effort toward building an ontology for reasoning on social network concepts and data, particularly in the field of modelling terrorist networks and organizations. CYC has contracted a team of subject matter experts (SMEs) in the field of terrorism. Using a specialized fact entry tool, the experts construct facts and rules that govern construction and evolution of terrorist networks[Deaton, Shepard, Klein, Mayans, Summers, Brusseau, Witbrock, and Lenat, 2005]. These facts are interpreted by CYC reasoning engine and stored as assertions in CycL. CycL is CYC's representation language which is a form of higher order predicate calculus[Reed and Lenat, 2002].

CYC can reason over these assertions by using a variety of special purpose reasoning modules as well as by using general theorem proving. For instance, if someone enters the fact that $Person-1$ is a member of terrorist group A, then CYC can conclude that $Person-1$ is a terrorist, based on a domain rule that states that anyone who is a member of a terrorist organization is a terrorist.

CYC can also do a variety of subsumption based reasoning. For instance, $\#\$hasLeaders$ represents a two place relation that relates an organization to one of its leaders. CycL allows to assert that $\#\$hasLeaders$ is a specialization of the $\#\$hasMembers$ relation thereby licensing the inference from "X is a leader of Y" to "X is a member of Y".

One of the main benefits of CYC's is the fact that over about 15 years, CYCorp has accumulated a large knowledge base of facts about human behaviour and relationships, thus using its knowledge bases to further supplement the knowledge entered by the SMEs.

However, CYCorp's approach has a number of shortcomings based on the history of CYC's knowledge base. The knowledge base developed by CYCorp is essentially a "flat" expert system with facts and relationships about facts given approximately the same priority in reasoning. CycL language is thus

not attuned to the idea of building object-oriented taxonomies. CYC treats relations between objects as rules rather then first-class objects - thus limiting its potential for implementation of social network analysis techniques.

5.8.3 Knowledge Engineering and Ontology Design

CYCorp has approached an important and difficult problem is working with data in ontological form. The problem relates to the fact that much of what people know about the semantics of the world is in the realm of "common sense" — which is at best difficult to describe in formal terms, and at worst could be even difficult to describe in a human language, thus passing into the realm of poetry. Donald Knuth has once said "...anything we can think of can be automated. The things we do without thinking are much more difficult".

CYCorp's approach to describing common-sense knowledge — about networks as well as other aspects of human life — is centered on creation of a massive knowledge base of facts, rules and inferences. This knowledge base is populated by sets of human subject-matter experts in fields from English literature to military strategy. This approach essentially creates islands of knowledge within the knowledge base that are described in high detail, potentially with large gaps between them. It is not clear whether such unified approach will eventually lead to creation of a true artificial intelligence, but CYCorp currently possesses the largest knowledge base in existence.

A different approach to creation of ontologies is rooted in the realization that creation of a knowledge base that encompasses all knowledge in the universe is probably impossible. Thus, an ontology writer must create a reduced subset of this knowledge — based on subject matter or data at hand — and create a set of ontological objects to adequately describe this subset.

In NetInference, I take the second approach, and resolve to define minimum sufficient amount of semantics needed given the domain or data at hand. As I show in the next several sections, even the minimal approach brings powerful results in reasoning about social systems and relationships.

5.9 Requirements for Representing Relational Knowledge

The Artificial Intelligence research community has developed a number of approaches for representing entities and relationships between them in the way that could be extended to represent social network data.

A good system for the representation of knowledge in a given domain should possess the following four properties[Frakes and Gandel, 1989]:

- Representational Adequacy: the ability to represent all of the kinds of knowledge that are needed in this domain.
- Inferential Adequacy: the ability to manipulate the representational structures in such a way as to derive new structures corresponding to new knowledge inferred from items already known.
- Inferential Efficiency: the ability to incorporate into the knowledge structure additional information that can be used to focus the attention of the inference mechanisms in the most promising directions. Techniques for enhancing inferential efficiency may include search heuristics, dataset pruning and definition of Regions of Interest (ROI).
- Acquisitional Efficiency: the ability to acquire new information easily. Ideally, the inference mechanism should also guide machine acquisition of new knowledge.

5.10 Representational Adequacy - Atomic Semantic Units

Objects and object-oriented modeling are commonplace in modeling complex data flows and business processes. However, the current dominant object oriented modeling technique — the Unified Modeling Language(UML), is ill equipped for modeling of organizational modeling[Castro, Mylopoulos, Alencar, and Filho, 2001]. Instead, UML is suitable for later phases of model refinement which usually focus on completeness, consistency, and verification.

This problem derives from the fact that object-oriented modeling relies on the assumption that concepts, entities and relationships among them can be arranged in a hierarchical taxonomy and concrete concepts deriving from abstract. Entity relationships in a social network context are much more complex and encompass a graph-oriented, non-hierarchical view of the world.

While hierarchical taxonomies are too restrictive for representation of social network knowledge, an elaboration of the object-oriented paradigm can serve well as a meta-language for representation of concepts present in a network structure.

The goal of such a meta-language is not to express and describe nodes and edges that comprise a social-network model but rather to facilitate machine derivation of such definitions by means of semantic inference. Thus, a meta language is a structure of meaning that underlies every object of a model and facilitates descriptive, qualitative reasoning about objects and their relationships.

Figure 5.4 illustrates the role of ontological or meta-structures in describing a network. In this illustration person A is a $friendOf$ person B

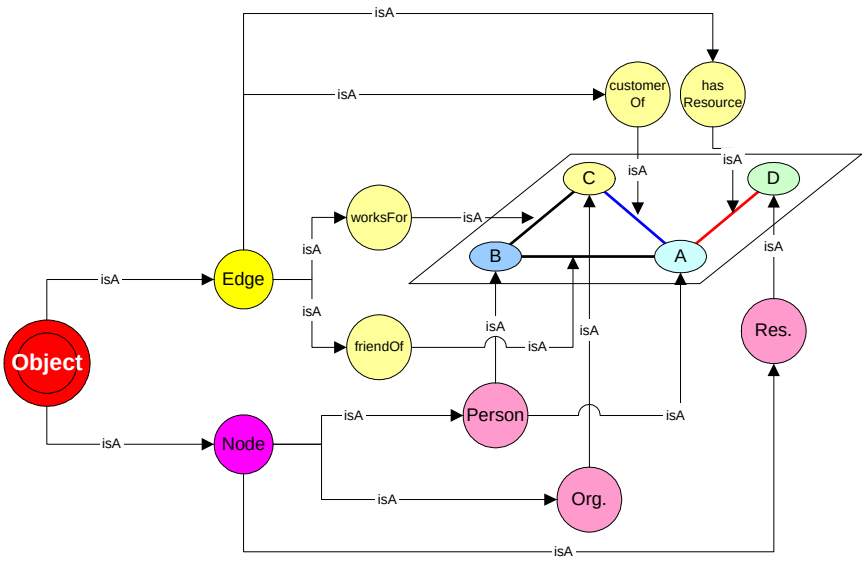


Figure 5.4: Network is just the tip of the iceberg: from complex taxonomy to simple networks

and *hasResource D*. Person *B worksFor* company *C* and person *A* is a *customerOf C*. While the network itself only consists of 4 nodes and 4 edges, it is only the tip of the iceberg.

To understand the meaning behind the nodes and edges while being able to infer knowledge about the network, a complex hierarchy of ontological entities must be set up. In this simple example, we only define three layers of ontological hierarchy. Abstract nodes (pink on the figure) and abstract edges (beige) are derived from obejcts Node and Edge (yellow and purple), which, in turn, inherit properties from a universal *Object* type. In studying real datasets, such hierarchies will be much more complex and involve many layers of inheritance.

5.10.1 Object Orientation - Mandatory Feature Set

Atkinson, et al[Atkinson, Bancilhon, DeWitt, Dittrich, Maier, and Zdonik, 1989] outline a set of basic features for manipulating and storing object-oriented data. The minimal feature set consists of eight qualities: complex objects, object identity, encapsulation, types or classes, inheritance, overriding combined with late binding, extensibility and computational completeness.

Complex Objects

Complex objects are built from simple objects through use of collection patterns — *set*, *list* and *tuple*. Collections must be orthogonal and applicable to both atomic units (e.g. integers, strings) as well as other objects.

NetInference implements objects through the use of associative sets, or sets of property-value pairs. An associative set can substitute for any of the collection patterns: a set is a simple collection of values, while a list uses property identifiers to specify an ordering and a tuple identifies individual values with a semantically significant handle. A value portion of the associative pair can be any of basic variable types, or another object.

Object Identity

The concept of object identity is the following: in a model that implements object identity, an object has an existence that is independent of its value. NetInference implements object identity through a global $object - store$ database, and ($gensym$) - a mechanism for generating globally unique object handles.

Encapsulation

NetInference objects encapsulate data and methods that act on data in a consistent manner. Moreover, due to Scheme's ability to manipulate executable objects ($\#PROCEDURE$ objects) as data, the storage mechanism for data and methods is exactly identical. NetInference objects encapsulate resolution rules that maintain object's self-consistency (e.g. value ranges of parameters) and its consistency with its surroundings (e.g. maintaining reference lists).

Classes

NetInference classes are defined implicitly: any object can serve as a concrete instance and then be converted into an abstract node and used as a parent for derivation of child objects through inheritance. This corresponds to the dynamic nature of NetInference objects - where the objets may be manipulated through their parent objects at runtime of the system. This corresponds to the view of classes taken in the data-warehousing community[Schaffert, Cooper, Bullis, Killian, and Wilpolt, 1986].

Inheritance and Overriding

NetInference implements object inheritance by specification of parent object at the time of object definition. Moreover, multiple inheritance is supported

as well. Properties and methods can be overridden by child objects both at the time of object creation as well as at runtime.

Extensibility and Computational Completeness

NetInference is based on Scheme, which is a Turing-complete language. Thus, any function that is not a part of NetInference may be implemented within the Scheme environment and processed by NetInference at runtime without further modifications to the code.

Extensibility of the object system can come from two approaches. New data types or data structures may be defined within the Scheme environment and manipulated by NetInference routines through their Scheme interfaces. Data types whose implementation for efficiency or other reasons cannot be done inside Scheme (e.g. images and multimedia data) can be implemented in C or C++ using a specialized API to create an interface between the NetInference environment and the custom objects.

Implementation details of the NetInference object system are described in section 5.13.

5.11 Summary

In this chapter, we introduce NetInference - a language designed for representation, searching and reasoning about network data. A concise comparison of NetInference and other languages for semantic data interchange and inferencing is given in table 5.2.

The rest of the chapter is organized as follows: section 5.13 describes the design and properties of object-oriented metaphor that underlies design of NetInference. Section 5.14 defines semantics of a graph or network based on the design of the object system. A system for declaring and resolving graph-based semantic rules is described in section 5.15. A graph query language based on rules and constructs of NetInference is specified in section 5.17

Section 5.16 illustrates the power of NetInference by adding semantics to existing social network data. In this example, I derive semantics of friendship and advice networks through decomposition of their edges into components of information flow, authority relationships and affinity connections and define a set of centrality measures that take this semantic decomposition into account to arrive at a new set of conclusions based on the combination of friendship and advice networks.

Section 5.19 introduces more complex semantics for reasoning about terrorist networks. The terrorist network ontology is based on a social network dataset compiled from investigation of the bombing of U.S. embassy in Tanzania.

	RDF	CyCL	OWL	Clips	NetInference
Semantic Structure	Object-Oriented	Flat	Object-Oriented	Flat	Object-Oriented
Object Orientation	Single Inheritance	Simulated with collections	Single Inheritance	Simulated with rules	Multiple Inheritance
Turing-Complete	No	Yes	No	Yes	Yes
Extensibility	External Logic	Fully Extensible	External Logic	Fully Extensible	Fully Extensible
Graph representation	Intrinsic	Simulated with collections	Intrinsic	Implemented with extensions	Intrinsic
Object query capability	External	Fully implemented	External	Implemented with extensions	Fully implemented
Graph query capability	External	External	External	External	Fully implemented
Reasoning on graph structures	External	Fully implemented	External	Implemented with extensions	Fully implemented
Regions of Interest	No	Yes (collections)	No	No	Yes (intrinsic)

Table 5.2: Comparison of features of inference and semantic representation languages

5.12 Design of the Social Network Semantic Language

Social Network Semantic Language — NetInference — is designed as an extension of Scheme[Abelson, Sussman, and Sussman, 1985]. There are a number of essential reasons for choosing a Scheme-based interpreted language for implementation:

- Scheme is a relatively simple yet Turing-complete programming language. Scheme interpreters are small and easily embeddable in other software tools. This feature can be used to create multi-agent simulations where each agent uses semantic reasoning to infer and plan its next action.
- Scheme implements the *lambda-calculus* paradigm that allows manipulation of procedures and code segments as data, and storage thereof in

data structures.

- Scheme is simpler and faster then Common Lisp, while providing all of the essential features required for implementation of the semantic reasoning systems.

Despite all of the advantages of using Scheme for implementation of NetInference, there was a significant obstacle: Scheme lacks means for an efficient implementation of hash tables as it does not support direct memory block manipulation. However, efficient hash table implementation is essential for building a fast object manipulation system. This obstacle was removed by implementing a fast hash table in C and linking it with a modified *Guile* Scheme interpreter. The hash table has been exposed to the Scheme interpreter as a data type and hash table objects can be stored and manipulated in the same way as any Scheme objects.

The following sections document the challenges and solutions present in design of NetInference as well as present examples and API documentation for usage of NetInference.

5.13 Object System

NetInference implementation is based on a custom, class-less object system built on top of Guile[Foundation, 2005]. The principles of object-orientation are derived from these specified by Atkinson[Atkinson, Bancilhon, DeWitt, Dittrich, Maier, and Zdonik, 1989] and Cardelli[Cardelli, 1988].

An object is defined as follows:

$$Object \quad := \quad \begin{cases} ObjectID : \\ ParentID : \\ Attributes := ListOf\{attributeID, value\} \\ Methods := ListOf\{methodID,\ value\} \\ Rules := ListOf\{rule\} \end{cases} \tag{5.4}$$

where $ObjectID$ is a globally unique identifier (with the exception of temporary objects (see section 5.13.1)), $ParentID$ is an identifier of an object that the current object inherits attributes and rules from (see section 5.13.3), *Attributes* is a list of attribute-value pairs, $Methods$ is a list of method-procedure pairs, and *Rules* is a set of rules to be executed during object-resolution procedure (see section 5.15).

Objects are defined by using the *defobject* macro:

```
(defobject <object-id>     //optional; defaults to a random
    unique id
```

```
#:as #$ParentObject
#:with-attr '((<name> <value>) (<name> <value>))
#:with-methods '((<name> <procedure>) (<name> <value>))
#:with-rules (list (defrule ...) (defrange ...)
```

Standard Object Methods

Objects include a number of standard methods, including:

getID : Return the current object ID

getField/setField : return or set the value of a named object property

getMethod : return an executable object (function) that can be called

resolve : trigger local rule resolution (see section 5.15)

These functions are discussed in detail in section 5.13.2.

5.13.1 Object Storage

By default, objects are stored in an *object-store* collection. *Object-store* is implemented as an instance of *Object* as well, which allows for greater code reuse. For example, the graph query operators (section 5.17) can operate on the entire database of objects as well as on individual objects or user-defined collections of objects — in exactly the same manner.

For ease of use, direct operations on the $object - store$ are hidden from the user. To allow easy access to objects inside the $object - store$, I have defined a special object-reference notation, from now on referred to as the $hash - buck$ notation. The use of $hash - buck$ notation is illustrated below:

```
;; Retrieve an object in standard Scheme notation
 (getObject object-store objectID)

;; Retrieve an object in hash-buck notation
 #$objectID
```

5.13.2 Object Properties, Methods and Rules

Attributes of an object are defined as a list of *attributeID-value* pairs where $attributeID$ is a locally unique identifier, and *value* is an instance of a simple type (*number*, *string*, *symbol*, *list* or any other type allowed in Scheme), or a reference to another *Object*.

To get and set values of *Attributes*, one should use the following API:

```
;;Get value of a field
(getField object attributeID)

;;Set value of a field
(setField object attributeID value)
```

To further simplify access to object fields and methods, a *dot − notation* is implemented

```
;; Retrieve a field of an object in standard Scheme notation
(getField (getObject object-store objectID) fieldID)

;; Retrieve a field of an object in hash-buck and dot notations
#$objectID.fieldID
```

Methods are defined similarly to *Attributes* as a list of name-value pairs. However, the value portion of method's name-value pair must be a Scheme procedure:

```
;;Get and execute a method
((getMethod object methodID) <arguements>)
```

Similarly, *Methods* can be accessed using the *dot − notation*:

```
;; Run a method of an object using standard Scheme notation
((getMethod (getObject object-store objectID) methodID)
   argument1 argument2)

;; Run a method of an object using hash-buck and dot notations
(#$objectID.methodID argument1 argument2)
```

Rules of an object is a list of Scheme expressions that compile into a *Resolve* function. A detailed description of the *Resolve* functions can be found in section 5.15.

5.13.3 Inheritance

An object system must implement property and method inheritance as a means of defining one object in terms of another.

NetInference does not make a distinction between class and instance. Every object in the system can be manipulated as a unit of data, or used as a prototype for creation of new objects.

```
(defobject "person"
 #:with-attr '(("name" "") ("age" 0)("location" "unknown"))
 #:with-methods '(("print" (method '(begin (display a) (newline)
    ))))
)

(defobject "jonny"
```

Listing 5.1: "Multiple inheritance in NetInference"

```
 (defobject "child"
  #:with-attr '(("name" "") ("age" 10)
                ("shoe-size" "5")("location" "unknown"))
  #:with-methods '(("print"
                    (method '(begin (display a) (newline)))))
 )

(defobject "student"
 #:with-attr '(("studentID" "")("location" "school"))
)

(defobject "jonny"
 #:as '((#$child #$student))
 #:with-attr '(("name" "jonny")("age" 10))
)

>#$jonny
(object
  id="jonny"
  parent=(#$child #$student)
  name="jonny"
  age="10"
  location="school"
  studentID=""
)
```

```
 #:as #$person
 #:with-attr '(("name" "Jonny")("age" 10))
)
```

In this example, *jonny* is defined as an instance of object *person*. *Jonny* overrides the *name* and *age* properties of *person*, but inherits the *location* property and *print* method of the parent object.

Multiple inheritance is allowed by specifying a list of objects in the `#:as` clause of the object definition. In multiple inheritance, order in which parent objects are specified matters: the objects specified later get priority over objects specified earlier, thus earlier values in overlapping fields (e.g. location field in the next example) will be overwritten by later values (see listing 5.1).

In this example, *jonny* is defined as instance of both *child* and *student*. Object *jonny* overrides the *name* and *age* properties of *child* but inherits $shoe-size$ and *location* from *child*, and $studentID$ from *student*. However, the multiple inheritance rule overrides the value of *location* field of *child* with the value of *location* field of *student*.

Listing 5.2: "Typing Mechanism"

```
(defobject "child"
  #:with-attr '(("name" "")("location" #$location)))

(defobject "school" as #$location)
(defobject "apple" as #$fruit)

 ;; Only values that are derived from #$location are allowed
 ;; as valued of the 'location' property

;; This is a legal object
 (defobject "jonny"
  #:as #$child
  #:with-attr '(("location" #$school)))

;; This statement will produce an error due to type system
    violation

(defobject jimmy"
␣␣#:as␣#$child
␣␣#:with-attr␣'(("location"␣#$apple)))
```

5.13.4 Typing Mechanism

NetInference uses a "medium-strength" typing mechanism: "simple types" (number, string, etc) are weakly typed, Lisp style; objects are strongly typed with an enforced parent hierarchy. An example of typing mechanism at work can be found in listing 5.2.

5.13.5 Abstract and Concrete Objects

In NetInference, there are no classes - any object can play a role of a concrete instance and in the same time serve as a parent object for a set of other objects. This allows for rapid propagation of changes throughout the object system without having to iterate over large sets of instances.

However, because of the ability to easily modify a significant object (e.g. *Person* or even *Object* itself), care must be taken by the user to consider all implications of such wide-ranging changes.

To ease the modeling process and allow for immutable truths to become a part of the system, the NetInference object system allows the user to specify some objects as *abstract*. In this case, the properties and methods of the object are considered read-only and cannot be modified without redefining the entire object. The only permitted operations on abstract are $getField$ and $getMethod$, and derivation of a child object. Note that the child object may override a property of an abstract object through polymorphic inheritance:

```
> (defobject "person" #:with-attr
    '(("disposition" "kind-spirited")))
> (setField #$person "abstract" true)

>#$person.disposition
kind-spirited

> (setField #$person "disposition" "evil")
 **writing to abstract objects not permitted**
 NIL

> (defobject "evil-person" #:as #$person
             #:with-attr '(("disposition" "evil")))

>#$person.disposition
evil
```

The main role of abstract objects is to convey a small set of ground truths about the ontology or model under development. They should not be used as a part of any executable structure other then as parent objects for the purpose of inheritance.

In the design spirit of NetInference, abstract objects are designated by setting an *abstract* property to *true*. After this point, no other modifications to the object will be permitted, including unsetting the *abstract* property. By default, all objects in NetInference except *Object* are created as concrete (i.e. lacking the *abstract* property), with full permissions on all fields.

As higher-level models are loaded into NetInference, the number of defined abstract objects will increase, but the general philosophy is still the same: abstract objects represent the basic instances that the model operates on and derives from. The graph model, described in the next section is a primary example of such a higher-level model.

5.14 Graph Representation

The object system of NetInference provides a foundation upon which a robust graph representation scheme is designed.

A graph is traditionally defined as a set of nodes or vertices V connected by a set of edges E. NetInference adapts this definition to fit the constraints of the object system by creating two abstract objects - *Node* and *Edge*. Nodes and edges of each graph represented in NetInference are defined as child objects of these abstract objects.

A $Node$ object is defined as:

$$Node = \begin{cases} nodeID : uniqueID \\ edges : setOf(E \Rightarrow Edge) \\ properties,\ methods\ and\ rules : inherited\ from\ Object \end{cases}$$

An $Edge$ is defined as:

$$Edge = \begin{cases} edgeID : uniqueID \\ from : N1 \Rightarrow Node \\ to : N2 \Rightarrow Nodeproperties,\ methods\ and\ rules : inherited\ from\ Object \end{cases}$$

Note that the definition of $Node$ contains a list of edges emanating from it and the definition of $Edge$ contains two objects of type $Node$. Such circular definition is permissible in NetInference due to delayed instantiation of properties (i.e., abstract object $Edge$ will not be referenced until a concrete edge emanating from a concrete $Node$ needs to be instantiated). However, it affords a remarkable convenience: in a graph traversal or calculation of graph-based metrics all required information for each node and edge is contained directly within the object and is accessible in constant time.

Of course, such savings of time come with a memory penalty, as each of the object references requires storage. The implementation of object references is quite memory-efficient and the penalty thus is fairly small. Moreover, such representation of graph elements does not require an external data structure other then what is provided by the object system - which counteracts or at least partially counteracts the memory penalty of maintaining circular data structures.

NetInference provides convenience macros $defnode$ and $defedge$ to ease creation and derivation of graph nodes (see listing 5.3).

5.14.1 Subgraphs

At this point, the object system and graph representation of NetInference resembles a primordial soup. It is a giant collection of various objects, such as nodes and edges, loosely tied together by means of references between objects — but with no overarching structure. The only mechanisms for manipulating objects provided in the object system are these of robust retrieval of an object by its handle, and of object inheritance.

Such chaotic representation of knowledge is not extremely useful, as users need mechanisms to effectively prune datasets and control the focus of search or reasoning algorithms. However, installing a permanent overarching structure is too restrictive since it forces the objects into a configuration that may or may not be efficient for the problem at hand.

Listing 5.3: "Node and Edge Definition"

```
(defnode <nodeID> #:as <parent_node>
 #:with-attr '(<list of attribute-value pairs>)
 #:with-edges '(
   (<edge-name> <edge-object>)
   ... list of edges ...
 )
)

(defedge <edgeID> #:as <parent-edge>
 #:with-attr '(<list of attribute-value pairs>)
 #:from <source-node>
 #:to <target-node>
)
```

However, properties of NetInference objects can contain any data type supported by the language, including other objects. The implication of this is that one can trivially define compound objects that contain nodes and edges of a graph.

This yields to a ready definition of such important and useful notions such as subgraphs and regions of interest. These notions serve as a lens through which views of large datasets can be focused. Moreover, as objects can be members of multiple subgraphs, this allows for an unlimited number of views to be defined based on structure of the model and conditions presented by the data.

A subgraph is defined as

$$G_{sub} = v_{sub} \in V, e_{sub} \in E$$

where v_{sub} and e_{sub} are subsets of the vertex and edge sets, respectively. In NetInference, a subgraph is defined as a compound object whose properties contain all objects in G_{sub}:

```
> #$subgraph1
(object id="subgraph1"
 n137=(object id="n137" parent=#$Node ...)
 ...
 n42=(object id="n42" parent=#$Node ...)

 e1= (object id="e1" parent=#$Edge ...)
 ...
 e24= (object id="e24" parent=#$Edge ...)
```

If one wishes to manipulate multiple graphs at the same time within NetInference, he should define them using this method - as the format of a subgraph is the same as the format of the object-store and elements of

a subgraph can be manipulated in the exact same manner as elements of the object-store. Moreover, the object-store itself can be thought of as a subgraph that encompasses every object defined in the system.

5.14.2 Regions of Interest (ROIs)

A region of interest (ROI) represents a set of nodes and edges selected in process of - or for the purpose of - running a data manipulation routine. While an ROI can be created by hand, most commonly the ROIs are returned as a result of running a query or applying a set of conditions to graphs or subgraphs - e.g. $select$ and $select-subgraphs$ operations described in detail in section 5.17.

An ROI is defined as an object that contains a collection of subgraphs:

$$ROI = (v_1 \in V, e_1 \in E, v_2 \in V, e_2 \in E ... v_n \in V, e_n \in E)$$

The subgraphs are indexed by randomly generated unique IDs. A $map-ROI$ function iterates over ROI's and performs user's choice of operation (or any $\lambda-expression$ allowed in Scheme). ROIs can be also used as arguments to the *select* function - which allows for nested queries.

5.14.3 Hypergraphs

Formally, a hypergraph G can be defined as a pair (V, E), where V is a set of vertices and E is a set of *hyperedges* between the vertices. Each hyperedge connects a set of vertices: $E = u, v, ... \in V$. (Hyperedges are undirected.)

An $HyperEdge$ is defined as:

$$HyperEdge = \begin{cases} edgeID : uniqueID \\ nodes : setOf(N \Rightarrow Node) \\ properties,\ methods\ and\ rules : inherited\ from\ Object \end{cases}$$

Hyperedges are child objects of *Edge* and can be manipulated using the same routines as other edges and freely mixed with edges of other types. However, hyperedges should be treated differently for a number of graph-theoretic concepts.

5.15 Inference in Social Networks

NetInference implements two types of inference on social network data. Object structure (sec. 5.13) and system of property inheritance facilitate hierarchical inference, where properties of objects within the social network can

be determined by inspecting their semantic predecessors within the object structure.

The second inference mechanism is that of graph-oriented rule resolution. In this mechanism, every object has a set of routines (rules) for checking its own self-consistency - i.e. the consistency of the internal state of the object with its external connections. The rules also provide for computing closures upon objects in the network (e.g. the transitive closure) and inferring existence of nodes and edges based on generalizable domain rules.

5.15.1 Rule Definition

A NetInference rule is specified using *defrule* macro:

```
(defrule <rule name>
  (if [conditional expression or query]
      (then [a sequence of statements to be executed if
         condition is true])
      (else [a sequence of statements to be executed otherwise])
  )
)
```

Rules can be placed into an object at the time of object creation through inheritance or at arbitrary times as specified by the user, e.g.:

```
;; rules are specified at the time of object creation
(defobject obj1
 .....
 #:with-rules '((defrule rule_1 <rule specification>)
                (defrule rule_2 <rule specification>)))

;;rules are inherited from obj1
 (defobject obj2 #:as obj1
   .....
 )

;;rules are added to the object at runtime
 (addRule obj2 (defrule rule_3 <rule specification>))
```

5.15.2 Example

As a simple example, let me define a set of rules on domain of locations and places. The rules specify information about possible ranges of parameters, and a transitive closure of nested locations.

First, let us define the notion of *location* as a node with two attributes — *latitude* and *longitude* — and some constraints on the values of the attributes:

$$-90 \leq latitude \leq 90$$

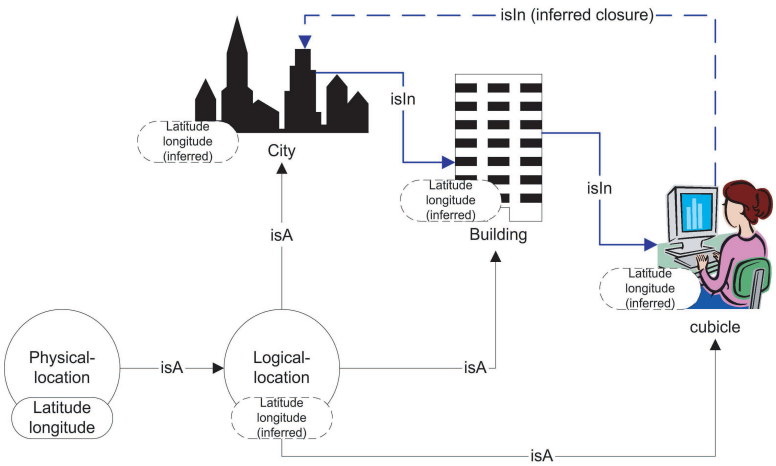


Figure 5.5: Inference of inherited properties and transitive closure

$$-180 \leq longitude \leq 180$$

```
(defnode "location"
 #:with-attr '(("name" "") ("latitude" 0) ("longitude" 0))
 #:with-rules '(
  (defrule lat_range
    (if (and (< this.latitude -90) (> this.latitude 90)) (then (
        error))))
  (defrule lon_range
    (if (and (< this.longitude -180) (> this.longitude 180)) (
       then (error)))))
 )
```

This definition provides us an ability to define a multitude of physical locations which can be placed on a map. However, of more interest is the notion of a logical location. We can define a logical location with arbitrary granularity - it can be a table in a cafe, a street corner, a city, or the entire planet:

```
(defnode "logical-location" #:as #$location)
(defedge "isIn" (from #$logical-location) (to #$logical-
    location)
  #:with-rules '(
   (defrule coords (and (setField from.latitude to.latitude)
                        (setField from.longitude to.longitude)))))$
```

The definition of *isIn* edge type allows for inference of a physical location (i.e. latitude and longitude) from a nesting series of logical locations. (see figure 5.5)

However, nested location hierarchy exhibits one more property - transitivity. For example, if a table *isIn* a cafe *isIn* Pittsburgh it can be inferred that the table also *isIn* Pittsburgh, i.e., there is not only an edge from a node

Listing 5.4: "Object Resolution Algorithm"

```
DO
DEQUEUE object O from the evaluation queue
RUN resolution rules of O and its parent objects
 IF an object other then O is altered, created or deleted by the
     rules
  THEN set the altered object's dirty bit to TRUE;
       add object to the evaluation queue
 END IF
 SET O's dirty bit to FALSE
UNTIL evaluation queue is empty
```

table to node *cafe* to node *Pittsburgh* but also an edge from node *table* to node *Pittsburgh*. Computing transitive closure can be done by defining *isIn* as a transitive edge:

```
(defedge "transitive-edge"
  #:with-rules '(
    (defrule transitive-closure
      (if (exists (this.to) (isParent? this.to.edge transitive-
          edge))
       (then (defedge #:as transitive-edge
                      (from this.from) (to this.to.edge)))))
)
```

The edge transitivity rule examines the edges connected to the target of the edge and if one of these edges is transitive (i.e. is a child object of a transitive edge) the transitive closure is computed by adding another edge. An edge can then be declared to be transitive by inheriting the rule from the $transitive - edge$ object.

While a set of rules declaring transitive closure relations between locations is fairly simple, it illustrates the mechanism by which NetInference can infer values of attributes of nodes and edges, and create new entities based on sets of rules.

5.15.3 Rule Resolution Mechanism

The rule resolution mechanism in NetInference is decentralized and based on the idea of lazy evaluation – the notion that at every moment of time only objects that change need to be watched and re-evaluated by the rule resolver.

Every NetInference object is tagged with a *dirty bit* – a flag that signals whether an object's attributes have been changed recently or whether edges that connect the object to other objects have been added or dropped. If a change has been made, the object's *dirty bit* is set to *true* and the object is added to a queue of unresolved objects.

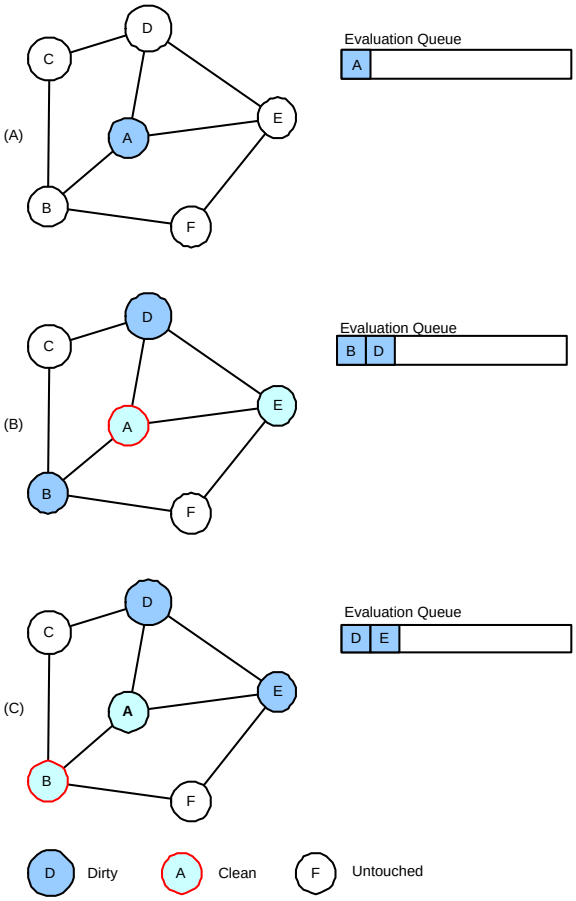


Figure 5.6: Inference of inherited properties and transitive closure

The rule resolution algorithm (see figure 5.6) can be described in the pseudo-code in listing 5.4.

Note that a node that has been already resolved by the algorithm may be changed again by another node's rules and thus will be re-added to the evaluation queue. This produces the distinct possibility that this algorithm may not converge for all graphs but instead may produce an oscillating behaviour.

Oscillations are undesirable if the goal is to produce a stable state where every object has been resolved and marked as *clean*. However, if the intention is to model dynamic systems, oscillation is a valid behaviour and must be studied as such - instead of being considered a property of the evaluation algorithm.

The algorithm then should be modified to achieve a double goal: within each iteration of the algorithm, convergence should be achieved. In the same time, oscillations should be possible in space of multiple iterations. This dual goal can be achieved by maintaining a $taboo - list$ of nodes that have been visited. If a node on the taboo list is referenced by another node, it does not

Listing 5.5: "Object Resolution Algorithm with Taboo List"

```
FOR desired number of iterations DO
 REPEAT
  DEQUEUE object O from the evaluation queue
  ADD O to the taboo list
  RUN resolution rules of O and its parent objects
  IF an object O1 is altered, created or deleted by the rules
  THEN set O1's dirty bit to TRUE;
    IF O1 is not on taboo list
    THEN ADD O1 to evaluation queue
    END IF
  END IF
  SET O's dirty bit to FALSE
 UNTIL evaluation queue is empty

/* Next iteration of resolving will start with nodes marked
   dirty
   but not resolved int he previous iteration */
 FOR all nodes on the taboo list
  IF node's dirty bit is set to TRUE
  THEN ADD node to the evaluation queue
  END IF
 END FOR
END FOR
```

get inserted into the evaluation queue but is still marked as *dirty*; it will be re-evaluated at the next iteration of the algorithm in listing 5.5.

If the system is convergent, this algorithm will converge after a finite number of iterations. However, if the system is not convergent, breaking the process into a set of iterations allows the users to capture dynamic state of the system and trace it through the use of time series.

5.15.4 Rule Resolution and Discrete Event Simulation

The multi-iteration property of the taboo-list object resolution algorithm (listing 5.5) lends itself to an application of NetInference as an engine for running discrete-event simulations of dynamic social systems.

Doing so requires an addition of an event queue, where events are represented as changes in state of objects (e.g. change of an object property, addition of a node or edge), occurring in time. At each time-step a set of events is removed from the event queue and applied to an object in the simulated network. Then one iteration of the rule resolution algorithm is run, propagating the state changes introduced by current events and continuing to resolve state changes introduced by previous events — thus producing a cumulative value.

State of the entire system can be then evaluated at every time step producing time-series data of an evolving dynamic network.

5.16 Domain Representation in NetInference: Friendship and Advice Networks

A classic example of two semantically significant networks is friendship and advice network[Krackhardt, 1990][Krackhardt, 1999]. The distinguishing feature of the friendship-advice dataset is the fact that, while the basic concepts of friendship and advice-giving are immediately understandable by a human observer, a meaningful SNA analysis of the combination of the two networks cannot be done using traditional methods.

The difficulty in combining friendship and advice networks is the fact that, semantically, edges in each of these networks represent fundamentally different concepts. Friendship is a reciprocal relationship between two humans that implies some shared beliefs, a degree of cooperation, and a degree of selflessness or willingness to sacrifice some self-interest in order to help a friend. Friendship is an intensely personal, emotional concept and in professional life can be a boost or a detriment to productivity in the workplace, depending on the circumstances.

On the other hand, advice networks are essentially information flows[Cross, Borgatti, and Parker, 2001] and can form in a professional, completely impersonal context as well as a context of personal affinity. Advice relationships are directed and may not be reciprocal. In this case, an advice relationship may also imply a degree of superiority. Furthermore, advice relationships may emerge from a friendship and friendships may evolve from a long-lasting advice relationships.

Cross, et. al. [Cross, Borgatti, and Parker, 2001], further decompose edges of advice networks into a number of subtypes. Specifically, people tended to provide: (1) solutions; (2) meta-knowledge; (3) problem reformulation; (4) validation and (5) legitimation. Such decomposition does not provide a useful insight into how advice networks are related to friendship or other personal networks – but it does point out the fact that semantics of advice edges is also is not uniform.

Thus, while collection of friendship and advice data on social networks is a standard practice and each of the networks can be analyzed and interpreted independently, the fundamental differences in semantics of friendship and advice edges prevent researchers from recognizing interdependence of the two networks and analyzing them as co-occurring complex phenomena.

In this section, I would like to propose an approach to treating such networks as semantically weighted constructs - which may open the door to

creation of combinatorial metrics spanning multiple networks.[1]

5.16.1 Construction of Friendship and Advice Semantics

Let us then construct a first-approach semantics for analyzing friendship and advice networks. Beginning with basic entities, let us assume that the only entities in this network are people and information. Let us also define two basic edge entities - personal affinity and a flow (see listing 5.6).

Let us then make a following set of conjectures to decompose the edges of an advice network:

1. If *Alice* gives unsolicited advice to *Bob*, the action can be decomposed as (a) an information transfer ($infoTransfer$) from *Alice* to *Bob* and (b) an assertion of authority ($asserts - authority$) edge: [2]

$$\begin{aligned} Alice &\rightarrow_{infoTransfer}\rightarrow Bob \\ Alice &\rightarrow_{asserts-authority}\rightarrow Bob \end{aligned}$$

2. If *Bob* solicits advice from *Alice*, the implication is:

$$\begin{aligned} Bob &\rightarrow_{respects}\rightarrow Alice \\ Alice &\rightarrow_{infoTransfer}\rightarrow Bob \end{aligned}$$

3. If *Alice* and *Bob* routinely exchange advice, we can assert $mutual - respect$ - by combining results of a previous rule applied in both directions:

$$\begin{aligned} Alice &\leftarrow_{infoTransfer}\rightarrow Bob \\ &\Downarrow \\ Bob &\leftarrow_{respects}\rightarrow Alice \end{aligned}$$

4. Existence of $asserts - authority$ edge between two actors results in a greater probability of an information transfer in the direction of the edge.

$$\begin{aligned} Alice &\rightarrow_{asserts-authority}\rightarrow Bob \\ &\Downarrow \\ Alice &\rightarrow_{infoTransfer}\rightarrow Bob \end{aligned}$$

[1] Please note that, at the risk of producing a work of pop-psychology, this example is grossly over-simplified. Yet, the chief goal of this section is demonstrating expressive capabilities of NetInference and not providing an exhaustive ontology describing friendship, advice and their implications

[2] Based on type of other relationships established in the group (e.g. formal vs. informal networks), assertion of authority may be accepted or contested. However, this level of detail is beyond the scope of this example.

Listing 5.6: "Basic Assertions on Advice Networks"

```
(defnode "person")
(defnode "information")

(defedge "likes"
 #:as #$emotion
 #:from #$person
 #:to #$person)

 (defedge "respects"
 #:as #$emotion
 #:from #$person
 #:to #$person)

 (defedge "asserts-authority"
 #:as #$emotion
 #:from #$person
 #:to #$person)

;;This edge is equivalent to saying "Person A told Person B
   about Fact C"
(defedge "infoTransfer"
 #:from #$person
 #:to #$person
 #:with-attr '(("subject" #$information))
 #:with-rules '(
   (defrule "validity"
     (if (not (existsEdge? #:from this.from #:to this.subject))
         (error)))
   (defrule "transfer"
     (defedge #:from this.to #:to this.subject))
))
```

5. Existence of *respect* edge between two actors results in a greater probability of an information transfer *against* the direction of the edge:

$$\begin{array}{ccc} Bob & \rightarrow_{respects} \rightarrow & Alice \\ & \Downarrow & \\ Alice & \rightarrow_{infoTransfer} \rightarrow & Bob \end{array}$$

All of these rules can be encoded as NetInference rules operating on concepts defined in listing 5.6.

Friendship is a much more complex concept, involving both a degree of cooperation, a degree of respect and a degree of emotional affinity. Fortunately, as basic units of reasoning have been already defined, we can express this interdependence:

1. If both *Alice* and *Bob* report a $friendship$ relationship, we can infer existence of edges representing mutual respect and emotional affinity (*like*). We can also infer the lack of $asserts - authority$ edge:

$$\begin{array}{ccc} Alice & \leftarrow_{isFriendOf} \rightarrow & Bob \\ & \Downarrow & \\ Alice & \leftarrow_{respects} \rightarrow & Bob \\ Alice & \leftarrow_{likes} \rightarrow & Bob \\ Alice & \not\leftarrow_{asserts-authority} \not\rightarrow & Bob \end{array}$$

   Using rules enumerated for advice networks, we can also infer that some information transfers will also occur between the actors.

5.16.2 Inferences on Friendship and Advice Networks

By using a set of rules described above and running the rule resolution algorithm until convergence, one can decompose the data collected as friendship and advice networks into more basic units - a directed networks of respect and authority, an undirected network of emotional affinity, and a set of information flows.

Thus, we can approach the problem of defining centrality[Freeman, 1979], on the friendship-advice diplex network by decomposing it into separate components as well:

- Out-degree of nodes on the authority network signifies actual position of the actor on the authority ladder (if it coincides with the formal network) or locates actors who are trying to assert authority beyond their means (i.e., "social climbers" or "problem children").

- Information flow centrality identifies experts and can predict diffusion of information through the organization. In this case, betweenness is probably a more important metric then degree centrality - as actors with high information flow betweenness are "information hubs" - collectors and disseminators of knowledge.

- Respect centrality is an in-degree of the respect network and signifies the position an actor holds in terms of respect.

- Degree of nodes in the affinity network signifies a social position of an actor - which may or may not coincide with his worthiness as a source of information, authority or respect.

While we can further define a "Superman Centrality" by combining the notions specified above, it would be a lossy preposition as these concepts hold different semantics and signify concepts that may not be combinable in a single actor (e.g. a person high in authority centrality may be respected but not a good source for information - and probably not well-liked as well).

5.17 Graph Query Language

As I have mentioned before, social network datasets have recently undergone significant change in their nature. Most classic SNA datasets have been small, self-contained, manually collected, and consisted of simple entities (e.g. binary graphs). Current state of the art of SNA employs large agglomerations of rich, machine-collected data encompassing open-ended data on large networks with often ill-defined boundaries.

In short, SNA has graduated from the ability to create reductionist theories from limited empirical data to the necessity of mining large, amorphous data spaces for confirmation of existing conclusions and derivation of new ones. A NetInference data space can be characterized as such an amorphous structure, and thus is a candidate for development and testing of data mining techniques that would allow efficient navigation and machine derivation of answers about actors and their correspondent social structures.

Navigating large, semantically loaded graph structures is a difficult problem. If a graph structure is treated as a collection of database tuples[Abiteboul, Quass, McHugh, Widom, and Wiener, 1997], it becomes easy to locate vertices and edges of the graph. However, in graphs, the structural attributes of an entity (i.e., pattern of connections related to a particular object) are frequently as important as a determinant of a role of the entity - and thus just as important for the purpose of data mining[McKay, Finin, and O'Hare, 1990].

Blau, Immerman and Jensen[Blau, Immerman, and D.Jensen, 2002] designed a system called QGraph that enables mining of graphs based on structural attributes of their nodes, edges and subgraphs - as well as properties of individual entities. A QGraph query is a labelled graph in which the vertices correspond to objects and edges to links. The query may place boolean conditions on the attribute values of objects and links, as well as global constraints relating one object or link to another. The query specifies the desired structure of vertices and edges, including types of links, existential and cardinal quantifier constraints that can be placed on extracted subgraphs. A QGraph query outputs a set of subgraphs - in NetInference terms, an ROI (sec. 5.14.2).

Of further interest is the QGraph's specification of what constitutes an entity of a graph. Without imposing a rigid taxonomy upon nodes and edges, QGraph uses semantically loaded concepts such as (in the film example outline in the paper) $film$, $peron$, $actorIn$ and $directorOf$. Specification of QGraph does not define any structure behind these concepts or their derivation. Thus, while providing significant power to the queries, the concepts of $nodeType$ and $edgeType$ are still essentially labels - which makes QGraph queries incapable of generalization.

For example, QGraph defines a concept of an $Oscar$ - a prestigious film award. However, there is a multitude of possible Oscar awards - Oscar for Best Picture, Best Actor, etc. As source data is coded, the designer of QGraph object taxonomy must make a trade-off: either every Oscar award is treated as a node to type $Oscar$, thus losing specificity — or a new node type has to be defined for every possible Oscar award, thus losing the ability to generalize.

NetInference addresses this problem by using the object inheritance mechanism to define node and edge type taxonomies. Thus, a query can specify properties as general or as specific as required by the user.

5.17.1 Design considerations

NetInference query system is built to satisfy a number of requirements:

- Queries must be able to efficiently find and return any object or collection of objects based on (a) an arbitrary boolean constraint on object's attributes, or (b) result of execution of an arbitrary expression on the object

- Queries must be able to find objects based on absolute, existential and cardinal quantifiers upon object's structural properties:

- Queries may return subgraphs and ROIs; the selection constraint may or may not match the return constraint, e.g. "return all subgraphs of

actors that are related to each other, and have been in films together"

5.17.2 Basic Queries

The NetInference query functions are designed to evoke similarity to SQL semantics, which would provide a familiar point of grounding to the user.

The *select* statement is defined as follows:

```
(select
    (from <source graph>)
    [(return <constraint expression>)]
    (where <constraint expression>)
)
```

where <source> graph is any object containing a graph - $object - store$, a subgraph or an ROI.

Constraint expressions are arbitrary Scheme expressions that return *true* or $false$. The object that the query is currently examining is referred to as "*this*". The *return* constraint is optional; if it is missing, *select* will return only nodes or patterns that have matched the *where* constraint.

For example, a simple query to select all actors in the actor-film graph would be written as:

```
(select
  (from object-store)
  (where (isParent? this #$Actor))
)
```

A more complex query to return an ROI consisting of actor, film pairs can be written as:

```
(select
  (from object-store)
  (return '((getField this "from") (getField this "to") this))
  (where (isParent? this #$actedIn))
)
```

The return constraint is a Scheme list of objects derived from the selected object (*this*); *select* will place them in a new subgraph structure and add the subgraph to the ROI.

Note that the query is written not to select pairs of $actor,film$), but rather to select the type of edge that commonly connects actors and films - $actedIn$. This simplifies the query by reducing the amount of indirection.

5.17.3 Quantifiers

NetInference can query graph entities based on a number of logical quantifiers described below:

Absolute quantifier: return a node $\iff$ it exhibits an exact set of edges as specified in the query.

> **Example:** "return a subgraph of all actors that acted in Casablanca"

```
(select (from object-store)
  (where (and (isParent? this #$actedIn)
              (equal? this.to.name "Casablanca")
              (isParent? this.from #$Actor))))
```

Existential quantifier: return a node if there exists at least one instance where the query pattern is satisfied.

> **Example:** "return all films that have had at least one remake"

```
(select (from object-store)
  (where (and (isParent? this #$film)
              (exists (from this.edges)
                      (where (isParent? this #$remakeOf))
              ))))
```

Cardinal quantifier: return a node if the query pattern is satisfied a given number of times.

> **Example:** "return all actors who have been in 5 or more films"

```
(select (from object-store)
 (where (and (isParent? this #$Actor)
             (< 5 (cardinality (from this.edges)
                               (isParent this.edges.to #$Film))))))
```

5.17.4 Summary

The semantics-based graph querying capabilities allow the user of NetInference to find and extract portions of a large graph structure based on a simple yet powerful query language that takes into account both attributes of individual objects and structural properties of subgraphs that the objects are embedded in.

Further, the query mechanism allows for building of generalized queries that use the object-oriented taxonomy of social network concepts to match a wide variety of related entities. The ability to create generalizable queries

based on the semantics of the data sets NetInference apart from labelled graph-based query systems such as Laurel and QGraph.

However, extraction of exact subgraphs is computationally expensive, and the main limitation of NetInference's query mechanism is its lack of facilities for inexact matching of subgraph structures. A logical first step in allowing inexact graph matching in NetInference is to allow fuzzy matches within the conditional statements inside *select*. It is also possible to define in NetInference the notion of fuzzy sets which would allow the search space of the query system to be pruned through pre-computation of potential matches.

Creating such extensions will not put a significant strain on the system as NetInference is based upon a complete Scheme interpreter. Thus, extensions can be implemented in a high-level language and dynamically loaded as needed.

5.18 Implementing Standard SNA Metrics in NetInference

As I have discussed earlier, the intent of NetInference is not to supplant the findings, algorithms and metrics of Social Network Analysis, but to build upon their structure and to add a layer of meaning to the mathematical constructs of SNA.

In this section, I demonstrate capabilities of NetInference as a tool for social network analysis through providing definitions and algorithms for computing standard SNA metrics within the NetInference data-space.

An interesting special feature of the NetInference object structure is the *locality* of definitions. In general, a node or an edge is defined as an object with attributes, and a set of attached rules or methods. The upshot of this property is the fact that methods of an object are unlikely to have direct access to (or even know of existence of) other objects - unless they are *adjacent* to them in graph terms - i.e. if the two objects are connected with an edge. Thus, global information, even as simple as number of nodes in the graph, is not available to rules embedded inside a node or edge object. Furthermore, not all objects in the data-space are necessarily graph objects — the object system does not restrict mixing an arbitrary number of object types within the object-store.

Global-level routines that operate on the entire object-store and compute global metrics can certainly be built. However, doing so violates the design philosophy of the object system, and specifically the properties of locality and late binding — and thus should be considered harmful[Dijkstra, 1968]. Keeping the properties of the object system in mind, I shall design a mechanism for implementation of social network metrics through purely object-oriented means.

5.18.1 Robust Graph Handling

In order to instantiate this design, let us return to the basic formalism of a graph, and redefine it through purely object-oriented means. In section 5.14, I defined a graph in NetInference as a set of nodes or vertices V connected by a set of edges E — or objects $Node$ and $Edge$. Nodes and edges of each graph represented in NetInference are defined as child objects of these abstract objects.

A $Node$ object is defined as:

$$Node = \begin{cases} nodeID : uniqueID \\ edges : setOf(E \Rightarrow Edge) \\ properties,\ methods\ and\ rules : inherited\ from\ Object \end{cases}$$

An $Edge$ is defined as:

$$Edge = \begin{cases} edgeID : uniqueID \\ from : N1 \Rightarrow Node \\ to : N2 \Rightarrow Node \\ properties,\ methods\ and\ rules : inherited\ from\ Object \end{cases}$$

Moreover, I defined a subgraph as A subgraph is defined as

$$G_{sub} = v_{sub} \in V, e_{sub} \in E$$

where v_{sub} and e_{sub} are subsets of the vertex and edge sets, respectively.

However, at that point, all elements were in place, but graphs only existed in an amorphous object-store collection and were recreated through edge traversals.

The need for graph-level computations dictates the fact that graph handling must be changed at least partially to support the notion of global calculation. However, assignment of a node to a particular graph need not be permanent, thus keeping with the spirit of amorphous object storage and late binding.

A NetInference graph class thus extends the notion of a subgraph as

$$Graph = \begin{cases} graphID : uniqueID \\ nodes : setOf(n \Rightarrow Node) \\ edges : setOf(e \Rightarrow Edge) \\ methods : setOf\ graph - level\ calculations \end{cases}$$

and thus is easily implemented as a NetInference abstract object:

```
(defobject "graph"
 #:with-attr '(("graphID" ""))
 #:with-nodes '((<list-of-nodes))
 #:with-edges '((<list-of-edges))
 #:with-methods '((<list-of-methods))
)
(setField #$graph "abstract" "true")
```

Note that both nodes and edges are represented as references (i.e. pointers), thus a particular node or edge can be a part of an arbitrary number of graphs. This design decision essentially turns *Graph* into a short-lived construct that can be created for the purpose of running a set of measures (e.g. through use of the query mechanism (see sec. 5.17)) and destroyed with no remorse moments later.

This temporary nature of the graph is a radical shift from the standard technique, and is a way to preserve properties of late binding and locality while implementing graph-level computations.

Furthermore, any object that is declared to inherit from *graph*, will immediately gain access to all of the functionality specified for graphs — thus any dataset returned by a query function is immediately analyzable.

5.18.2 Centrality Metrics

As I show in this section, it is possible to calculate virtually any graph-based centrality metric using NetInference graph representation. However, when nodes and edges are semantically loaded, any action that collapses them into a single-mode graph needs to be justified in terms of the semantics, lest further calculations on the resulting graph produce meaningless results.

For example, if the original data-space contains nodes of type *person* and *knowledge*, the data must be filtered to either single-mode $person \rightarrow person$ or $knowledge \rightarrow knowledge$ graph, or a bi-modal $person \rightarrow knowledge$ graph.

Using the query operator (*select*), a user can extract a subgraph that satisfies this condition, and then use any standard graph-theoretic measure to analyze it. In the remainder of this section, I discuss a number of analysis routines to calculate values of degree and closeness centrality, and a MetaMatrix measure of cognitive demand.

Degree

The degree of a point is viewed as important as an index of its potential communication activity Freeman[1979] defines degree centrality as

$$C_D(P_k) = \frac{\sum_{i=1}^{n} a(p_i, p_k)}{n-1}$$

where P_i, P_k are nodes in the graph, n is the number of nodes in the graph and $a(p_i, p_k) = \begin{cases} 1 \; if \; an \; edge \; p_i \rightarrow p_k \; exists \\ 0 \; otherwise \end{cases}$. A given point, p_k can at most be adjacent to $n-1$ other points in a graph. The range of $C_D(P_k)$, therefore, is $\frac{0}{n-1} = 0 \leq C_D \leq \frac{n-1}{n-1} = 1$. Furthermore, this figure is normalized with respect to the number of nodes in the graph, so comparison of graphs of different sized is possible.

In terms of NetInference, the easiest way to implement degree centrality is by use of a **cardinal** quantifier defined in the graph query subsystem (sec. 5.17:

$$C_D(P_k) = \frac{cardinality(edges(P_k))}{cardinality(nodes) - 1}$$

```
(defobject "graph"
...
#:with-methods '(
 (defmethod "degree"
   (foreach (let node (this.nodes))
     (setField node.freeman_degree
       (\ (cardinality (node.edges))
          (- (cardinality (this.nodes)) 1))))
 )
))
```

This algorithm calculates Freeman degree centrality on nodes of a graph and assigns the values as an attribute of each node of the graph. While it calculates correct values of centrality, the assignment of the value as a node attribute violates the principles of late binding — e.g. values of centrality will persist with the nodes long after the graph object has been destroyed.

Thus, a slightly different storage mechanism is needed, which will associate values of metrics computed on nodes of a graph with these nodes, but only within the scope of this single graph. A simple solution is to use an associative table, such as:

	Degree	Betweenness	Closeness
1	0.75	0.1	0.4
2	0.23	0.2	0.314

The table associates values of two keys: $nodeID$ and $measureID$ with the value — and thus can store results of every per-node calculation done on the graph inside the graph structure. Degree centrality code modified to use the associative table looks like this:

```
(defobject "graph"
#:with-attr '(("measures" (make-assoc-table)))
...
#:with-methods '(
```

```
  (defmethod "degree"
    (foreach (let node (this.nodes))
      (set-table-cell this.measures (node.nodeID) "degree"
        (\ (cardinality (node.edges))
           (- (cardinality (this.nodes)) 1))))
  )
))
```

This code preserves the locality and late-binding properties of the graph structure. Thus, if a particular node is a member of more then one graph, it can have a distinct centrality value associated with each of the graphs.

While the formula for computing Freeman degree centrality is simple, this example demonstrates the way a graph-based measure can be calculated, stored and referred to within NetInference. The next examples will use the associative-table storage mechanism by default.

Closeness

Closeness centrality was defined by Sabidussi in 1966 [Sabidussi, 1966]. He proposed that the centrality of a point be measured by summing the geodesic distances from that point to all other points in the graph. Actually, this is a measure of inverse centrality since it grows as points are far apart, and centrality in this context means closeness. Mathematically, this can be defined as following:

If we let $d(p_i, p_k)$ = the number of edges in the geodesic linking p_i and p_k, then Sabidussis measure of the "farness" of a point p_k is

$$C_c^{-1}(p_k) = \sum_{i=1}^{n} d(p_i, p_k)$$

$C_c^{-1}(p_k)$ grows with increasing distance between p_k and other points; it is an inverse of centrality for point p_k.

The algorithm for calculating closeness centrality for point p_k is essentially a breadth-first search of the graph. A full BFS of the graph will compute the shortest distances from p_k to every other node. However, one iteration of this process is $O(n^2)$ in computational complexity, so computation for the entire graph is $O(n^3)$ - which is a very computationally expensive process.

However, a shortcut can be made. Let $D_{i=1..n}^{j=1..n}$ be a matrix of real numbers, each cell i, j of the matrix containing the shortest distance between nodes p_i and p_j. Then, a full iteration of the BFS will only need to be run once. On any subsequent iteration, only cells that remained blank on the previous runs (i.e. corresponding to the edges that did not lie on a shortest path during a traversal from another starting point) will have to be filled in.

In NetInference, the distance table can be implemented as an associative table as well, so a more efficient algorithm can be implemented (see listing 5.7.

Listing 5.7: "Computing closeness centrality in NetInference

```
(defnode "graph"
.....
 #:with-attr '(("distance-table" (make-assoc-table)))
 #:with-method '(
    ;;BFS procedure takes a
   (define (bfs current-node, start-node)
     ;;iterate through every node that this node is connected
        to
       (foreach (let edge current-node.edges)
       ;;If this edge has not been traversed yet, travese it
          and recurse
       (if (eq? (get-table-cell distance-table start-node.
          nodeID edge.to.nodeID) 0)
         (begin
           ;;Set the link as traversed; distance=1
           (set-table-cell distance-table current-node.nodeID
              edge.to.nodeID 1)
           ;;Distance from start-node to the next node is d(
              start,current)+1
           (set-table-cell distance-table start-node.nodeID
              edge.to.nodeID
             (+ (get-table-cell distance-table start-node.
                nodeID current-node.nodeID) 1))
           (bfs edge.to, start-node)
 ))))

 ("closeness-centrality"
   (foreach (let node this.nodes)
     (bfs node node) ;;run the traversal with result caching
     (foreach (let node2 this.nodes)
       ;;compute the value of closeness centrality and insert
          it
       ;;into the attribute table
       (set-table-cell this.measures node "closeness-centrality
          "
         (+ (get-table-cell this.measures node.nodeID "
            closeness-centrality")
            (get-table-cell distance-table node.nodeID node2.
               nodeID)))
     )
   )
 )
)
)
```

5.18.3 MetaMatrix Measures: Cognitive Demand

Cognitive demand, described by Carley [Carley and Ren, 2001], measures the amount of effort each person expends in performing actual tasks using the knowledge, resource, task and communication networks of the MetaMatrix.

Cognitive demand is a notion similar to the task load measure developed at NASA [Hart and Staveland, 1988]. It measures the extent to which the person has to engage in mental activity to do the assigned tasks, defined as:

1. number of people person i interacts with / total number of people in the group
2. number of tasks person i is assigned to / total number of tasks
3. amount of information person i possesses / total amount of knowledge

Algorithmically, this is very similar to the calculation of degree centrality, and uses the **cardinality** operator:

```
(defmethod "cognitive_demand"
  (foreach (let node (this.nodes))
    (set-table-cell this.measures (node.nodeID) "cognitive-
       demand"
      (+ (\ (cardinality (node.edges) (where (isParent? node.
         edges.to #$person)))
            (- (cardinality (this.nodes) where (isParent? node
               .edges.to #$person))) 1)
         (\ (cardinality (node.edges) (where (isParent? node.
            edges.to #$task)))
            (cardinality (this.nodes) (where (isParent? node.
               edges.to #$task))))
         (\ (cardinality (node.edges) (where (isParent? node.
            edges.to #$knowledge)))
            (cardinality (this.nodes) (where (isParent? node.
               edges.to #$knowledge))))
      )
    )
  )
)
```

5.19 Reasoning about Terrorist Networks

The following example illustrates use of DyNetML for representing simple social network datasets. The data in this example is derived from the indictments against perpetrators of a terrorist bombing of the U.S. Embassy in Tanzania.

I will describe the example — which is a small but semantically dense dataset — in sections by introducing more complex concepts as they derive

from simpler concepts. At each step, I will demonstrate how rule resolution and ontological inference enable NetInference to answer questions that are not addressable with standard SNA techniques. At the end of this section, I will list complete source code for the ontology related to the Tanzania Bombing dataset.

5.19.1 Specifying Task Structures

Let us start by defining the notion of a task. In this ontology, a task is a node of a precedence graph; precedence relations specify what subtasks need to be accomplished before a given task can be started. A task occurs at a point of time and at a location. The notions of logical location and transitive closure of logical locations are defined in section 5.15.2.

```
(defnode "task"
 #:with-attr '(("time" 0) ("location" #$location))
 #:with-edges '(
   ("precededBy" #$precededBy)
 )
)

;;This edge defines precedence between tasks
(defedge "precededBy" #:from #$task #:to #$task
 #:with-rules '(
   ;;This rule forces tasks that precede each other in
       relations
   ;;to also precede each other in time.
   (defrule "time-precedence" (if (> this.from.time this.to.
       time) (error)))
 )
)
```

The following example defines a number of distinct tasks from the Embassy Bombing dataset and specifies their precedence relationship.

```
(defnode "surveillance" #:as #$task)
(defnode "weapon_training" #:as #$task)
(defnode "driving_training" #:as #$task)
(defnode "bomb_preparation" #:as #$task)
(defnode "bombing" #:as #$task)

(defedge (genid) #:as #$precededBy #:from #$bomb_preparation #:
   to #$weapon_training)
(defedge (genid) #:as #$precededBy #:from #$bombing #:to #
   $driving_training)
(defedge (genid) #:as #$precededBy #:from #$bombing #:to #
   $bomb_preparation)
(defedge (genid) #:as #$precededBy #:from #$bombing #:to #
   $surveillance)
```

A task is also defined to require knowledge and resources, and accomplished by a person:

```
;;These node definitions will be elaborated on later.
 (defnode "knowledge")
 (defnode "resource")
 (defnode "person")

;;Redefine task with information and resource requirements
 (defnode "task"
  #:with-attr '(("time" 0) ("location" #$location))
  #:with-edges '(
    ("precededBy" #$precededBy)
    ("requiresResource"  (defedge "requiresResource" #:from #
       $task #:to #$resource))
    ("requiresKnowledge" (defedge "requiresKnowledge" #:from #
       $task #:to #$knowledge))
    ("accomplishedBy" (defedge "accomplishedBy" #:from #$task #:
       to #:person)
  )
 )
```

A person is a more complex concept to define. Besides demographic information, a person possesses information and resources and can be involved in tasks:

```
 (defnode #$person
  #:with-attr '(...demographic attributes...)
  #:with-edges '(
    ("connectedTo" (defedege "connectedTo" #:from #$person #:
       to #$person))
    ("hasKnowledge" (defedge "hasKnowledge" #:from #$person #:
       to #$knowledge))
    ("hasResource" (defedge "hasResource" #:from #$person #:to #
       $resource))
    ("involvedIn" (defedge "assignedTo" #:from #$person #:to #
       $task))
  )
 )
```

5.19.2 Edge Derivation

Some of the tasks defined above are described as "training" - e.g. *driver_training*, *weapons_training*. We can define *training* as a task where some knowledge is transferred to a person undertaking the training; another person teaches at this training session:

```
 (defnode training as #$task
  #:with-edges '(
```

```
    ("knowledgeGiven" (defedge "knowledgeGiven" #:from #
        $training #:to #$knowledge))
    ("taughtBy" (defedge "taughtBy" #:from #$training #:to #
        $person))
    ("taughtTo" (defedge "taughtTo" #:from #$training #:to #
        $person))
  )
  #:with_rules '(
    ;;After taking the training, the student will know the
        subject taught
    (defrule "knowledgeTransfer"
      (defedge (getid) #:from this.taughtTo.to #:as hasKnowledge
          #:to this.knowledgeGiven.to))

    ;;After taking training, the student will know the teacher
        and respect him
    (defrule "respect"
      (defedge (getid) #:from this.taughtTo.to #:as #$respects
          #:to this.taughtBy.to))

    ;;And the teacher will assert some authority in regards to
        student
    (defrule "authority"
      (defedge (getid) #:from this.taughtBy.to #:as #
          $assertsAuthority  #:to this.taughtTo.to))
  )
)
```

5.19.3 Question Answering

The ontology defined here is an incomplete, yet functional specification of an organization engaged in some set of interdependent tasks, complete with knowledge and resource requirements.

One example of questions that can be answered using the reasoner mechanism is a process referred to by military analysts as *Capability Assessment*. It can be thought of as a determination whether a particular task is feasible and can be completed by the organization at hand.

In terms of entities defined so far, a task can be deemed $feasible$ if and only if: (a) *person* assigned to the task possesses *knowledge* and *resources* required for the *task*, and (b) all subtasks of this task are also feasible or already completed. This can be decomposed into:

$$isFeasible(Task) \iff$$
$$\forall task_i \in subtasks(Task) : isFeasible(task_i) \wedge$$
$$\forall resource_i \in requiredResources(Task) : obtainable(resource_i)$$

$$obtainable(resource_i) \iff$$
$$\exists person_i \in accomplishedBy(Task) : person_i hasResource resource_i$$

```
(defnode "task"
 ....
 (defrule "isFeasible?"
  (and
    ;;Apply feasibility rules to all of the subtasks
    (foreach this.precededBy (isFeasible? this.precededBy.to))

    ;;Apply resource congruence
    (foreach
      (let res this.requiresResources)
        (exists
          (from (let actor this.accomplishedBy)
            (where (exists (from (let res2 actor.hasResource))
              (equal? res2 res)))))
    )
    ;;...repeat for knowledge requirements...
  )
 )
```

At this point, the feasibility analysis does not take into account social networks between people. Let us use definition of friendship networks in section 5.16 and conjecture the following rule:

$$can - obtain(person_i, knowledge_i) \iff$$
$$\exists person_j \in infoTransfer(person_i) : person_j hasKnowledge knowledge_i$$

or, a *person* can obtain some $knowledge_i$ if he is in a relationship that permits information transfer with another person that has the required knowledge. To reiterate, the edge type $infoTransfer$ is an whose existence can be derived from collected friendship and advice networks through use of the reasoner rules, so collection of simple social network data is actually sufficient to fill the requirements of the above rules.

5.20 Example: Inference of Edges in a Terrorist Network

This section illustrates use of NetInference in a number of examples based on the dataset described in the previous section, and provides a set of instruction for usage of the tool to reason about MetaMatrix-based domains. The example also contains a number of examples of rule derivation.

The dataset for Tanzania Embassy Bombing is listed in section 5.21). The original dataset contains *agents*, *resources* and *tasks*, as well as edges:

- linking agents to each other (*social network*),
- agents to resources (*resource network*),
- agents to tasks (*task assignment*),
- tasks to resources (*requirement network*), and
- tasks to tasks (*precedence network*).

In the intelligence community, human or signal intelligence allows "Blue Team" to collect data on the social network (who knows whom) and the resource network (who has what resources)[Ronfeldt and Arquilla, 2001]. The precedence network and resource requirement network are generally well-known, as this information can be approximated from analyzing *post-factum* information on past terrorist events as well as from available terrorist training manuals.

The goal of an intelligence analyst would then be to determine if the terrorist cell or network has current capabilities for execution of attacks, and if not, how difficult would it be for the cell to obtain these capabilities. Then, if nodes capable of planning and executing an attack are found, it is very important to find the individuals behind the potential attack and target them for arrest or isolation.

The following scenario illustrates use of NetInference in this intelligence analysis scenario. For the purpose of demonstrating inference and query capabilities, I have deleted all edges linking agents to tasks. Thus, the goal of the system is to re-infer this information and determine if the terrorist cell in question is capable of executing a truck bomb attack.

The rules are set up as follows:

- A task can be decomposed into subtasks using **precededBy** edges.
- Any task or subtask has resource requirements, given by **requiredForTask** edges
- Agents have resources; given by **hasResource** edges.

- An agent **canObtain** a resource if he knows somebody that **has** this resource
- An agent **isCapable** of executing a task if and only if the agent **has** or **canObtain** all resources needed for the task AND **isCapable** of executing all sub-tasks of a task.

After all task assignment edges have been removed from the dataset, the rules were invoked with a set of queries as specified below. In this dataset, NetInference is capable of recovering 16 out of 23 task-assignment edges — re-inferring 88% of task assignments that could be accomplished by one person, and 69% of all task assignments.

A further enhancement consists of a rule that specifies that an agent **canDelegate** a task assignment to another agent if said agent **isCapable** of accomplishing the task. With this additional rule, NetInference uncovers 3 out of 5 edges that require group cooperation - i.e. 3 out of 5 people that participate in planning of a large-scale terrorist attack.

```
unix> ./netInference
Starting Guile...Done
Loading NetInference...Done
```

Guile Interpreter and NetInference have been loaded

```
(defnode "agent")
(defnode "task")
(defnode "resource")

(defedge "knows" #:from #$agent #:to #$agent)
(defedge "hasResource" #:from #$agent #:to #$resource)
(defedge "requiredForTask" #:from #$resource #:to #$task)
(defedge "precededBy" #:from #$task #:to #$task)
```

Give basic definitions of nodes and edges (can be also loaded from file as shown below)

```
(defedge "canObtainResource" #:from #$agent #:to #$resource)
(defedge "capableOf" #:from #$agent #:to #$task)
```

These edges are not present in the data file but will be inferred in the course of the demonstration

```
guile> (load "metamatrix.scm")
guile> (load "data/embassy.scm")
```

Load the MetaMatrix definitions and the data file (see next appendix for full listing)

```
;;;;Do a hierarchical decomposition of a task by finding
;;;;all its precedence relations and recursing through them
;;;;until all dependencies are found.
(define (checkPrecedents obj)
  (if (not (isParent? obj #$task))
      (error "argument is not a task object")
      )

  (newline)
  (display "Hierarchical decomposition of")
  (display obj)
  (newline)

  (let ((precedence-edges
   (select (getField obj "edges")
     (where '(isParent? this #$precededBy)))))
    (begin
    (map-fields  precedence-edges
     (lambda (obj2)
       (begin
   (display obj2)
   (if (not (equal? (getField obj2 "to")  obj))
       (checkPrecedents (getField obj2 "to"))
       #f)))

))))

guile> (checkPrecedents #$bombing)

Hierarchical decomposition of#<object  <bombing edges=""edges"" parent=""task"" />>
#<object  <246 from=""bombing"" parent=""precededBy"" to=""driving_training"" />>

Hierarchical decomposition of#<object  <driving_training edges=""edges"" parent=""task"" />>
#<object  <246 from=""bombing"" parent=""precededBy"" to=""driving_training"" />>
#f

#<object  <249 from=""bombing"" parent=""precededBy"" to=""bomb_prep"" />>
Hierarchical decomposition of#<object  <bomb_prep edges=""edges"" parent=""task"" />>
#<object  <243 from=""bomb_prep"" parent=""precededBy"" to=""weapon_training"" />>

Hierarchical decomposition of#<object  <weapon_training edges=""edges"" parent=""task"" />>
#f
```

Do a hierarchical decomposition of a task by finding out its precedents and recursing through them. This function also checks for illegal cycles in the precedence structure.

```
(define (hasResource? agent res)
    (if (not (isParent? agent #$agent))
      (error "argument is not an agent")
      )
    (if (not (isParent? res #$resource))
      (error "argument is not a resource")
      )

    (if (exists (edges agent) (lambda (this) (equal? (getField this "to") res)))
  #t)
)
```

Find out if an agent has direct access to a resource

```
(define (canObtainResource? agent res)
    (if (not (isParent? agent #$agent))
      (error "argument is not an agent")
      )
    (if (not (isParent? res #$resource))
      (error "argument is not a resource")
      )

    (if (hasResource? agent res) #t)

    (let ((friend-edges (select (edges agent) (where '(isParent? this #$knows)))))
      (if (exists friend-edges
      (lambda (this) (hasResource? (getField this "to") res)))
    #t #f)
      )
)
 guile>
```

An agent can obtain a resource if he already has it - or it he knows somebody that does.

```
(set-resolve-func #$agent
      (lambda (this)
        ;;For each resource, find out if the agent can get it
        (map-fields (select object-store (where '(isParent? this #$resource)))
        (lambda (res)
          (begin
            (display res)
            (if (canObtainResource? this res)
          (begin
            (display "Creating edge")
            (defedge (edgeid) #:as #$canObtainResource #:from this #:to res)))))
        )
        )
)
```

Now lets us add a rule to the "agent" class to support inference of "canObtainResource" edges. Essentially, the rule says "for every resource that can be obtained by this agent, add the appropriate edge to the system

```
guile>(resolveObject #$al-Fawwaz)
al-Fawwaz
#<object  <bomb_material edges=""edges"" parent=""resource"" />>
Creating edge

#<object  <building_for_bombmaking edges=""edges"" parent=""resource"" />>
Creating edge

#<object  <media_consultant edges=""edges"" parent=""resource"" />>
Creating edge

#<object  <money edges=""edges"" parent=""resource"" />>
Creating edge

#<object  <religious_extremism edges=""edges"" parent=""resource"" />>
Creating edge

guile> (select object-store (where '(isParent? this #$canObtainResource)))
#<object  <result523 edge492=""edge492"" edge496=""edge496"" edge500=""edge500"" edge504=""edge504"
    " edge508=""edge508"" edge512=""edge512"" edge516=""edge516"" edge520=""edge520"" />>
```

Run resolution on one object, then run a query to see which new edges have been created; These edges can be accessed by their IDs from the object-store or by saving query results to a variable

```
guile>(resolve-all)
```

Now we can run resolution on all nodes and create all missing edges of this type

```
(define (isCapableOf? agent task)

  ;;
  (display "Evaluating␣Capability␣of␣")
  (display agent)
  (display "to␣perform␣")
  (display task)
  (newline)
```

This function will evaluate an agent and a task and find out through hierarchical decomposition whether the agent in question can accomplish the task. The agent is deemed capable of performing the task if and only if all required resources can be located, and the agent can also accomplish all subtasks of the task.

```
  ;;check resource congruency
  (and
   (begin
     (display "Checking␣Resource␣Congruency...")
     (let ((task-resources
      (select (edges task) (where '(isParent? this #$requires)))))
       (alltrue (map-fields task-resources
          (lambda (requirement)
            (begin
       (display (getField requirement "to"))
       (newline)
       (canObtainResource? agent (getField requirement "to"))
       ))
            )
    ))
     )
```

Check all resource requirements of the task and determine if the task is feasible. If the resources are not available and cannot be obtained from friends, return failure.

```
    (begin
      (display "Checking␣Precedences")
      (let ((precedence-edges (select (edges task) (where '(isParent? this #$precededBy)))))
        (alltrue
     (map-fields precedence-edges
          (lambda (obj2)
            (begin
       (display obj2)
       ;;check for illegal cycles
       (if (not (equal? (getField obj2 "to") task))
           (isCapableOf? agent (getField obj2 "to"))
           #f))))))
      )

    )
)
```

Check all subtasks of a task and attempt to accomplish them as well.

```
guile>(select object-store (where '(and (isParent? this #$agent) (isCapableOf? this #$bombing))))
```

Query: which agents are capable of executing a terrorist attack This query will return a list of all agents that can muster resources and capabilities to execute a bombing. The query returns "false" - no agents are capable of executing a truck bomb attack on their own. Given this scenario, a group of 3 agents must come together before an attack is possible.

```
guile> (map-fields (select object-store (where '(isParent? this #$task)))
 (lambda (task) (select object-store (where '(and (isParent? this #$agent) (isCapableOf? this
     task)))))
```

This query will iterate through all tasks and find who is capable of executing it. The query returns a list of agent-task pairs.

5.21 Tanzania Embassy Bombing Dataset

```
(defnode "Mohammed_Rashed_Daoud_al-Owhali" #:as #$agent)
(defnode "Khalfan_Khamis_Mohamed" #:as #$agent)
(defnode "Mohammed_Sadiq_Odeh" #:as #$agent)
(defnode "Ahmed_the_German" #:as #$agent)
(defnode "Fazul_Abdullah_Mohammed" #:as #$agent)
(defnode "Wadih_al_Hage" #:as #$agent)
(defnode "Usama_Bin_Ladin" #:as #$agent)
(defnode "Ali_Mohammed" #:as #$agent)
(defnode "Ahmed_Khalfan_Ghailani" #:as #$agent)
(defnode "Mohammed_Salim" #:as #$agent)
(defnode "al-Fadl" #:as #$agent)
(defnode "al-Fawwaz" #:as #$agent)
(defnode "Jihad_Mohammed_Ali_Azzam" #:as #$agent)
(defnode "abouhalima" #:as #$agent)
(defnode "Abdullah_Ahmed_Abdullah_Saleh" #:as #$agent)
(defnode "Abdal_Rahman" #:as #$agent)
```

People comprising the organization

```
(defnode "surveillance" #:as #$task)
(defnode "weapon_training" #:as #$task)
(defnode "driving_training" #:as #$task)
(defnode "bomb_prep" #:as #$task)
(defnode "bombing" #:as #$task)
```

Tasks and subtasks that the organization attempts to accomplish

```
(defnode "religious_extremism"#:as #$resource)
(defnode "weapons_expertise"#:as #$resource)
(defnode "surveillance_expertise"#:as #$resource)
(defnode "media_consultant"#:as #$resource)
(defnode "building_for_bombmaking"#:as #$resource)
(defnode "money"#:as #$resource)
(defnode "bomb_material"#:as #$resource)
(defnode "truck"#:as #$resource)
```

Types of material and information resources

```
(defedge (genid) #:as #$knows #:from #$Mohammed_Rashed_Daoud_al-Owhali #:to #$Usama_Bin_Ladin)
(defedge (genid) #:as #$knows #:from #$Mohammed_Rashed_Daoud_al-Owhali #:to #$Jihad_Mohammed_Ali_Azzam)
(defedge (genid) #:as #$knows #:from #$Mohammed_Rashed_Daoud_al-Owhali #:to #
    $Abdullah_Ahmed_Abdullah_Saleh)
(defedge (genid) #:as #$knows #:from #$Mohammed_Rashed_Daoud_al-Owhali #:to $Abdal_Rahman)
(defedge (genid) #:as #$knows #:from #$Mohammed_Sadiq_Odeh #:to #$Wadih_al_Hage)
(defedge (genid) #:as #$knows #:from #$Ahmed_the_German #:to #$Abdullah_Ahmed_Abdullah_Saleh)
(defedge (genid) #:as #$knows #:from #$Fazul_Abdullah_Mohammed #:to #$Wadih_al_Hage)
(defedge (genid) #:as #$knows #:from #$Fazul_Abdullah_Mohammed #:to #$Usama_Bin_Ladin)
(defedge (genid) #:as #$knows #:from #$Wadih_al_Hage #:to #$Mohammed_Sadiq_Odeh)
(defedge (genid) #:as #$knows #:from #$Wadih_al_Hage #:to #$Fazul_Abdullah_Mohammed)
(defedge (genid) #:as #$knows #:from #$Wadih_al_Hage #:to #$Usama_Bin_Ladin)
(defedge (genid) #:as #$knows #:from #$Wadih_al_Hage #:to #$Ali_Mohammed)
(defedge (genid) #:as #$knows #:from #$Wadih_al_Hage #:to #$al-Fawwaz)
(defedge (genid) #:as #$knows #:from #$Wadih_al_Hage #:to #$abouhalima)
(defedge (genid) #:as #$knows #:from #$Usama_Bin_Ladin #:to #$Mohammed_Rashed_Daoud_al-Owhali)
(defedge (genid) #:as #$knows #:from #$Usama_Bin_Ladin #:to #$Fazul_Abdullah_Mohammed)
(defedge (genid) #:as #$knows #:from #$Usama_Bin_Ladin #:to #$Wadih_al_Hage)
(defedge (genid) #:as #$knows #:from #$Usama_Bin_Ladin #:to #$Ali_Mohammed)
(defedge (genid) #:as #$knows #:from #$Usama_Bin_Ladin #:to #$al-Fawwaz)
(defedge (genid) #:as #$knows #:from #$Ali_Mohammed #:to #$Wadih_al_Hage)
(defedge (genid) #:as #$knows #:from #$Ali_Mohammed #:to #$Usama_Bin_Ladin)
(defedge (genid) #:as #$knows #:from #$al-Fawwaz #:to #$Wadih_al_Hage)
(defedge (genid) #:as #$knows #:from #$al-Fawwaz #:to #$Usama_Bin_Ladin)
(defedge (genid) #:as #$knows #:from #$Jihad_Mohammed_Ali_Azzam #:to #$Mohammed_Rashed_Daoud_al-Owhali)
(defedge (genid) #:as #$knows #:from #$abouhalima #:to #$Wadih_al_Hage)
(defedge (genid) #:as #$knows #:from #$Abdullah_Ahmed_Abdullah_Saleh #:to #$Mohammed_Rashed_Daoud_al-
    Owhali)
(defedge (genid) #:as #$knows #:from #$Abdullah_Ahmed_Abdullah_Saleh #:to #$Mohammed_Sadiq_Odeh)
(defedge (genid) #:as #$knows #:from #$Abdullah_Ahmed_Abdullah_Saleh #:to #$Abdal_Rahman)
(defedge (genid) #:as #$knows #:from #$Abdal_Rahman #:to #$Mohammed_Rashed_Daoud_al-Owhali)
(defedge (genid) #:as #$knows #:from #$Abdal_Rahman #:to #$Abdullah_Ahmed_Abdullah_Saleh)
```

```
(defedge (genid) #:as #$hasResource #:from #$Mohammed_Rashed_Daoud_al-Owhali #:to #$religious_extremism)
(defedge (genid) #:as #$hasResource #:from #$Mohammed_Rashed_Daoud_al-Owhali #:to #$weapons_expertise)
(defedge (genid) #:as #$hasResource #:from #$Mohammed_Rashed_Daoud_al-Owhali #:to #
    $surveillance_expertise)
(defedge (genid) #:as #$hasResource #:from #$Khalfan_Khamis_Mohamed #:to #$weapons_expertise)
(defedge (genid) #:as #$hasResource #:from #$Khalfan_Khamis_Mohamed #:to #$surveillance_expertise)
(defedge (genid) #:as #$hasResource #:from #$Khalfan_Khamis_Mohamed #:to #$building_for_bombmaking)
(defedge (genid) #:as #$hasResource #:from #$Mohammed_Sadiq_Odeh #:to #$religious_extremism)
(defedge (genid) #:as #$hasResource #:from #$Mohammed_Sadiq_Odeh #:to #$weapons_expertise)
(defedge (genid) #:as #$hasResource #:from #$Mohammed_Sadiq_Odeh #:to #$surveillance_expertise)
(defedge (genid) #:as #$hasResource #:from #$Mohammed_Sadiq_Odeh #:to #$building_for_bombmaking)
(defedge (genid) #:as #$hasResource #:from #$Fazul_Abdullah_Mohammed #:to #$religious_extremism)
(defedge (genid) #:as #$hasResource #:from #$Fazul_Abdullah_Mohammed #:to #$weapons_expertise)
(defedge (genid) #:as #$hasResource #:from #$Fazul_Abdullah_Mohammed #:to #$surveillance_expertise)
(defedge (genid) #:as #$hasResource #:from #$Fazul_Abdullah_Mohammed #:to #$building_for_bombmaking)
(defedge (genid) #:as #$hasResource #:from #$Fazul_Abdullah_Mohammed #:to #$bomb_material)
(defedge (genid) #:as #$hasResource #:from #$Wadih_al_Hage #:to #$religious_extremism)
(defedge (genid) #:as $hasResource #:from #$Wadih_al_Hage #:to #$weapons_expertise)
(defedge (genid) #:as #$hasResource #:from #$Wadih_al_Hage #:to #$bomb_material)
(defedge (genid) #:as #$hasResource #:from #$Usama_Bin_Ladin #:to #$media_consultant)
(defedge (genid) #:as #$hasResource #:from #$Usama_Bin_Ladin #:to #$money)
(defedge (genid) #:as #$hasResource #:from #$Ali_Mohammed #:to #$religious_extremism)
(defedge (genid) #:as #$hasResource #:from #$Ali_Mohammed #:to #$surveillance_expertise)
(defedge (genid) #:as #$hasResource #:from #$Ahmed_Khalfan_Ghailani #:to #$building_for_bombmaking)
(defedge (genid) #:as #$hasResource #:from #$Ahmed_Khalfan_Ghailani #:to #$truck)
(defedge (genid) #:as #$hasResource #:from #$Mohammed_Salim #:to #$religious_extremism)
(defedge (genid) #:as #$hasResource #:from #$Mohammed_Salim #:to #$weapons_expertise)
(defedge (genid) #:as #$hasResource #:from #$Mohammed_Salim #:to #$money)
(defedge (genid) #:as #$hasResource #:from #$al-Fadl #:to #$religious_extremism)
(defedge (genid) #:as #$hasResource #:from #$al-Fadl #:to #$weapons_expertise)
(defedge (genid) #:as #$hasResource #:from #$al-Fawwaz #:to #$media_consultant)
(defedge (genid) #:as #$hasResource #:from #$Jihad_Mohammed_Ali_Azzam #:to #$religious_extremism)
(defedge (genid) #:as #$hasResource #:from #$Jihad_Mohammed_Ali_Azzam #:to #$weapons_expertise)
(defedge (genid) #:as #$hasResource #:from #$Abdullah_Ahmed_Abdullah_Saleh #:to #$surveillance_expertise)
(defedge (genid) #:as #$hasResource #:from #$Abdullah_Ahmed_Abdullah_Saleh #:to #$building_for_bombmaking
    )
(defedge (genid) #:as #$hasResource #:from #$Abdullah_Ahmed_Abdullah_Saleh #:to #$bomb_material)
(defedge (genid) #:as #$hasResource #:from #$Abdullah_Ahmed_Abdullah_Saleh #:to #$truck)
(defedge (genid) #:as #$hasResource #:from #$Abdal_Rahman #:to #$bomb_material)
```

Who has access to materials and information?

```
(defedge (genid) #:as #$requires #:from #$weapon_training #:to #$weapons_expertise)
(defedge (genid) #:as #$requires #:from #$driving_training #:to #$weapons_expertise)
(defedge (genid) #:as #$requires #:from #$bomb_prep #:to #$weapons_expertise)
(defedge (genid) #:as #$requires #:from #$bomb_prep #:to #$bomb_material)
(defedge (genid) #:as #$requires #:from #$bombing #:to #$religious_extremism)
(defedge (genid) #:as #$requires #:from #$bombing #:to #$weapons_expertise)
```

Material and information requirements for subtasks

```
(defedge (genid) #:as #$precededBy #:from #$bomb_prep #:to #$weapon_training)
(defedge (genid) #:as #$precededBy #:from #$bombing #:to #$driving_training)
(defedge (genid) #:as #$precededBy #:from #$bombing #:to #$bomb_prep)
```

Precedence relations between the main task (bombing) and its subtasks

5.22 Getting Started with NetInference - Step by Step

At this stage of development, NetInference does not have a graphical interface; all interactions with the system are done via a command shell. While command a command shell is not as user friendly, it affords immediate access to all functionality of NetInference. Further, it diminishes system requirements for supporting NetInference. Currently, NetInference can be used on any system that supports the GNU compiler suite (i.e. GCC, G++, libtool, etc).

The command shell is started by typing in the terminal:

```
unix> ./netInference
Starting Guile...Done
Loading NetInference...Done
guile>
```

The `guile>` prompt is the main interface to the system; any NetInference or SCHEME commands can be given at this moment.

For the purpose of this example, let us walk through working with MetaMatrix data and enhancing the ontology and the dataset given in the previous set of examples. First, load the MetaMatrix ontology:

```
guile> (load "metamatrix.scm")
```

As the previous example stated, the ontology defines the following categories of nodes and edges:

```
(defnode "agent")
(defnode "task")
(defnode "resource")

(defedge "knows" #:from #$agent #:to #$agent)
(defedge "hasResource" #:from #$agent #:to #$resource)
(defedge "requiredForTask" #:from #$resource #:to #$task)
(defedge "precededBy" #:from #$task #:to #$task)
```

However, the dataset has a number of nodes that my be treated differently for purposes of inferencing:

```
(defnode "driver_expertise" #:as #$resource)
(defnode "weapons_expertise" #:as #$resource)
```

These nodes, while classified as **resource** actually describe knowledge or information resources. The difference between information resources and physical resources is the fact that information can be copied, or passed from person to person. In fact, the dataset also includes specification for tasks specialized in procurement of knowledge - or **training**:

```
(defnode "weapon_training" #:as #$task)
(defnode "driving_training" #:as #$task)
```

In this example, the ontology will be enhanced to allow inference on the notions of knowledge and training.

First, let us define knowledge as a special kind of resource, having knowledge as a special case of having a resource, and change the parent class of *driver_expertise* and *weapons_expertise*.

```
guile> (defnode "knowledge" #:as #$resource)
guile> (defedge "hasKnowledge" #:as #$hasResource)
guile> (setField #$driver_expertise "parent" #$knowledge)
guile> (setField #$weapons_expertise "parent" #$knowledge)
```

Then, let us define the notion of training: Training is a task that adds knowledge to its participants. To do this, several other notions have to be defined as well:

```
guile> (defnode "training" #:as #$task)
guile> (defnode "weapon_training" #:as #$training)
guile> (defnode "driving_training" #:as #$training)

guile> (defedge "participatedIn" #:from #$person #:to #$task)
guile> (defedge "subjectTaught" #:from #$training #:to #
    $knowledge)
```

Then, the following rule can be defined: Every student that participates in a training session learns the subject taught:

```
guile> (addRule #$training
 (lambda (training)
   ;;For every agent linked to a training event
   (mapFields (select (edges training) (where '(isParent? this.
       to #$agent)))
     ;;;
     (lambda (student)
       ;;add an edge from the student to the knowledge being
           taught
       (defedge #:as #$hasKnowledge #:from student
         #:to (select (edges training) (where '(isParent?
             this #$subjectTaught)))))))))
```

The most mysterious to an outside user concept in above code is the notion of *lambda* functions. Essentially, a *lambda* is a temporary function that behaves like any SCHEME function but does not have a name. Lambda-functions can be passed to other functions as arguments and stored inside data structures, as well as inside NetInference objects. The first use of a *lambda-function* in this code is to frame the rule as an executable object and pass it to to the rule resolution mechanism of the object.

The second use is in the *mapFields* function iterates over results of a query, and applies a *lambda* function to them, in this case creating edges for every iteration.

This rule is now a part of definition of **training** class, and will be applied to all people involved in a training session at the next iteration of the rule resolution algorithm. However, one more step of preparation is required before this enhancement is fully functional.

The original MetaMatrix ontology did not formally link training events with knowledge concepts defined in the dataset; these are merely represented as textual features of the node labels. The formal links need to be defined by creating several edges:

```
guile> (defedge #:as #$subjectTaught #:from #$weapon_training
    #:to #$weapons_expertise)
guile> (defedge #:as #$subjectTaught #:from #$driving_training
    #:to #$driving_expertise)
```

Now the ontology enhancement is ready to be applied. Let us observe its application on the following nodes and edges:

```
(defnode "Khalfan_Khamis_Mohamed" #:as #$agent)
(defnode "Mohammed_Sadiq_Odeh" #:as #$agent)

(defedge #:as #$participatedIn #:from #$Khalfan_Khamis_Mohamed
    #:to #$weapon_training)
(defedge #:as #$participatedIn #:from #$Mohammed_Sadiq_Odeh #:
   to #$weapon_training)
```

Then, run one iteration of rule resolution by typing:

```
guile> (resolve-all)
```

The following edges will be created for two agents described above:

```
(defedge #:as hasKnowledge #:from #$Khalfan_Khamis_Mohamed #:
   to #$weapons_expertise)
(defedge #:as hasKnowledge #:from #$Mohammed_Sadiq_Odeh #:to #
   $weapons_expertise)
```

This exercise provides a basic walk-through for adding a semantic concept to an ontology and using the resolver to apply this concept to an existing dataset with minimal manual intervention. It makes use of many capabilities of NetInference, including object-oriented architecture, query system and rule resolution algorithm.

5.23 Inferencing Capabilities for the MetaMatrix

Sections 5.19, 5.20 and 5.22 detail portions of a larger ontology that builds a qualitative machine understanding of MetaMatrix structures and datasets. This ontology includes rules for inferring:

- Social network ties achieved through completion of tasks: for example, if two people attend the same training, one can infer that they know each other;
- Knowledge ties: knowledge can be communicated by acquaintances or through training;
- Resource ties: resources can be obtained through acquaintances and social ties;
- Task completion: ability to achieve an objective given a current state of the network.

The ontology, as it stands now, depends heavily on nodes and edges that comprise the tactical model of the task: the task precedence network, and knowledge and resource requirements for every task. This is due to the fact that inference rules for other networks depend on the tactical model data to stay constant throughout the reasoning process.

To test inferential performance of the MetaMatrix ontology, I have removed sets of edges from the Embassy Bombing dataset and allowed NetInference to re-infer them based on the rules of the ontology. The measurement was conducted as follows: after removing of a set of edges, one round of rule resolution was run, and number of re-inferred edges counted, as well as number of edges that were inferred by the ontology but were not a part of the original dataset. During the tests, only edges of one type were removed, other edge types were kept as constant.

The edge inferences for resource network include edges inferred as **canObtainResource** and knowledge network includes edges inferred during participation in training.

The ontology performed as follows:

The capability of NetInference to re-infer removed edges degrades with removal of social network edges and resources edges, as these edges are used as basic building blocks for reasoning about tasks and capabilities. In the dataset described in section 5.21, knowledge and resource information was confounded into a single set of edges. However, using participation in training as a guideline, NetInference was able to uncover 6 edges related to knowledge acquisition. Meanwhile, if all data on task capability was removed, the

	Original	Dropped	Re-Inferred	New Inferences
Social Network	29	5 10 15	3 6 7	8 3 2
Resource Network	36	5 10 15	2 2 0	10 6 4
Knowledge Network	0	0	0	6
Task Capablity (single agent)	23	23	16	2
Task Capability (group)	5	5	3	1

ontology would enable recovery of 88% of tasks that could be accomplished by a single agent.

5.24 Scalability and Computational Complexity

NetInference is designed to be scalable, even though it frequently operates on complex domain data.

Object-based inference (e.g. inferring transitive closure and similar concepts) has a computational complexity proportional to the depth of the object hierarchy, or close to $< O(log(n))$ where n is the number of objects in the object-store. Through load-testing, inference in an artificially created hierarchy 100,000 levels deep has taken ≈ 40 seconds.

The rule resolution algorithm operates by activating rules attached to a node, and then following edges of the node to its neighbors. Thus, computational complexity of the rule resolution algorithm is on the order of number of nodes plus number edges of the graph, or $O(n + n^2)$ where n is number of nodes. However, the real complexity of rule resolution depends on the semantic interdependence of nodes and concepts. If the interdependence factor is small, the propagation of change through the network will stop at the immediate neighbors of the starting node; if the concepts are highly interdependent, there is a possibility of oscillation which would cause the rule resolution algorithm to not converge. The case of non-convergence is handled by breaking up rule resolution into discrete iterations.

The query mechanism operates by matching a declarative model to the nodes and edges in the object-store. The worst-case computational complexity of queries is $O(n * k)$ where n is the number of objects in the object-store and k is the size of the model. However, if the declarative models specify not

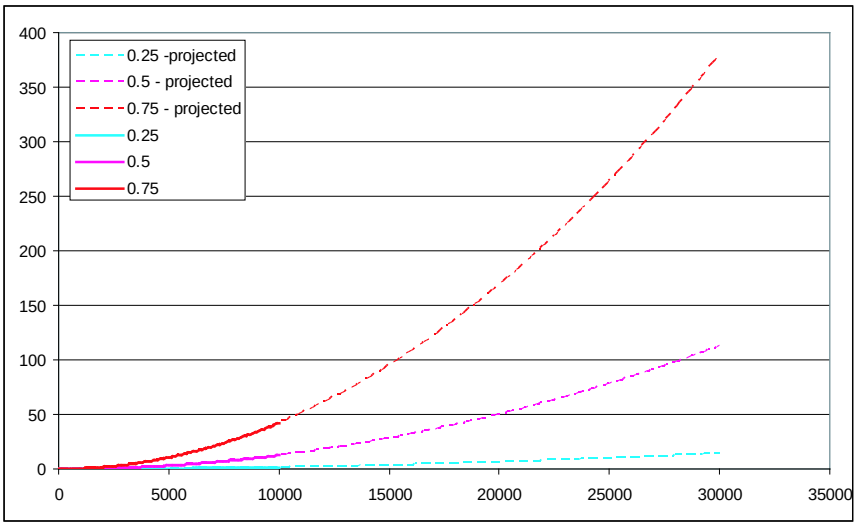


Figure 5.7: Timing of Rule Resolution for node interdependence levels of 0.25, 0.5 and 0.75

only the shape of sought subgraph, but also its semantics, the query mechanism prunes the dataset very quickly. As the model gets more semantically specific, the average-case complexity of queries will approach $O(n)$.

Measured and projected timings on the inference engine can be found in figure 5.7. The timings were obtained by running the inference engine on randomly generated structures with different interdependence levels. The interdependence level is generated as density of isA edges, and is an Erdös random graph structure.

It is difficult to conduct full scalability tests on model matching due to the fact that generation of realistic declarative models for load-testing is a non-trivial problem. Most likely, such models will have to be generated using evolutionary programming techniques, and thus constitute a separate research project beyond the scope of this dissertation.

5.25 Linking NetInference and NetWatch through DyNetML

NetInference allows the user to infer existence of nodes and edges not present in the original data but implied by the domain model and rule-set. Similarly, NetWatch can be viewed as a means of inferring evolution of an organizational network, through the means of agent-based simulation. Given the complimentary nature of the two tools, it is useful to provide a means for their interoperation.

The two tools can coexist and exchange data by means of DyNetML,

which I have designed and implemented specifically to facilitate data interchange between various social network analysis and simulation tools. DyNetML is described in detail in chapter 8. In this section, I discuss the means of conversion of DyNetML files (e.g. ones produced by NetWatch) for use with NetInference, and conversion of NetInference data for use with NetWatch.

The main difficulty in round-trip conversion is the fact that NetWatch (and DyNetML) rely on a flat type system, and NetInference is object-oriented and thus implements a hierarchical type derivation system. The typesystem employed by NetInference provides a considerably richer means of description of nodes and edges in complex organizational systems, but it should also interoperate with simpler representations.

Fortunately, the NetInference query mechanism provides a simple means for "flattening" of the object hierarchy. Thus, information can be output in a DyNetML format through a simple mechanism:

```
 (define people (select (from object-store) (where (isParent?
    this "person"))))
 (define knowledge (select (from object-store) (where (isParent
    ? this
"knowledge"))))
 (define tasks (select (from object-store) (where (isParent?
    this
"task"))))
 (define resources (select (from object-store) (where (isParent
    ? this
"resource"))))
 (define organizations (select (from object-store) (where (
    isParent?
this "organization"))))

 (write-dynetml people knowledge resources tasks organizations)
```

In the resultant DyNetML files, the class hierarchy will be preserved through the *Properties* mechanism (described in detail in section 8.7). The name of the parent object is output in DyNetML as a property of the node named "parent". The parent object is similarly output as DyNetML, continuing up the hierarchy until encountering a basic parent object (e.g. "person"). In this example, we have output a node of type $media-consultant$ which is derived from *employee*, and in turn, from *person*:

```
<node id="media-consultant">
 <properties>
  <property name="parent" type="string" value="employee"/>
 </properties>
</node>

<node id="employee">
 <properties>
```

```
  <property name="parent" type="string" value="person"/>
 </properties>
</node>
```

DyNetML input operates on similar principle. If the DyNetML node contains a "parent" property, the imported node will be interpreted as derived from the stated parent-node:

```
<node id="media-consultant">
 <properties>
  <property name="parent" type="string" value="employee"/>
 </properties>
</node>

is interpreted as:

 (defnode "media-consultant" #:as #$employee
  #:with-attr ...the rest of node properties...
 )

 ;if "employee" has not been defined yet, a placeholder node
    will be created
 (defnode "employee" #:as #$person)
```

Using the DyNetML conversion, NetInference can generate input files for NetWatch, or read in and interpret NetWatch output files.

5.26 Conclusions and Future Work

The goals of NetInference are two-fold: provide a consistent means for specifying semantics of nodes and edges in a multi-plex, multi-mode network, and provide a means for inference on entities in the semantic structure.

NetInference accomplished these goals by defining a consistent language where semantics of nodes and edges can be specified through object-oriented mechanisms of encapsulation and inheritance. A number of provisions for specifying inference rules, such as a existential, universal and cardinal quantifiers, allow for quick implementation of complex relationships between people, organizations or other entities. A graph query system makes use of the quantifiers and rule resolution to quickly query graphs and extract subgraphs and regions of interest that can be used for further exploration of the network data space.

NetInference does not provide ready-made ontologies for reasoning about any type of data that could be encountered. Creation of such comprehensive ontologies is a time- and resource-consuming proposition. One of the chief limitations of NetInference is that any dataset encoded in it would require development of its corresponding ontology. However, this also means that the

ontologies do not have to be comprehensive and a minimal set of rules and assertions may be sufficient for datasets that do not exhibit high ontological variability.

Future work on NetInference will be conducted in a several directions:

Fuzzy subgraphs and probabilistic edges At this point, NetInference facilities do not expressly forbid but also do not facilitate implementation of probabilistic edges. While edge objects can be assigned a probability value, the rule resolution and graph query systems do not take probabilities into account.

After changes to the rule resolution system are made, the existence of probabilistic edges will allow Bayesian inferences on social network graphs.

With addition of probabilistic rules to the graph query system, it becomes possible to specify fuzzy rules for matching subgraphs - which is more powerful and significantly more efficient way to query graph data.

Network model fitting The goal of this extension of NetInference is, given a network and set of graph models (defined as sets of objects and rules), to determine which of the given models best fits the network in question. Drawing from work on computer vision and object recognition, I will develop distance metrics that measure similarity between areas in the subject network and models. The purpose of model fitting is, to given a number of theoretical assumptions, determine which fit the empirical data and which are more applicable as a description of the empirical network.

Evolution of network models Treating all a set of rules as a search space, my goal is to create an evolutionary-programming solution that will automatically develop descriptive models of empirical data. These rule-sets can be then used to find other networks that exhibit similar properties - or be used as an input to a natural language generation system that will create verbal descriptions of these networks based on the inferred rules.

Part III

Generation, Warehousing, Manipulation and Interchange of Large-Scale Social Structure Datasets

Chapter 6

Generation of Network Topologies for Simulation Experiments

Testing large-scale dynamic network simulation packages such as NetWatch requires a large quantity of test data to be available for each of the experiments. The test data includes initial topologies of agents' social networks and specification of knowledge networks for each of the agents to fit an empirically derived distribution of knowledge. Another task is creation of realistic task structures that could be used to simulate performance of complex interdependent projects by groups of agents.

The main concern in generation of artificial data is its realism. Based on open-source empirical data (such as described in sec. 2.3), the artificial datasets need to approximate certain qualities or parameters found in the empirical data. However, it is unclear at the outset what parameters need to be emulated to achieve highest fidelity simulation.

Frequently, theories of network topologies in a particular setting are proposed. For example, large amount of social network research relies on assumptions made by Erdös [Erdös and Rényi, 1960] regarding topology and distances in random graphs. However, it is now clear that purely random graphs are not a good approximation of topology of social networks. Other proposed topologies include scale-free networks[Barabasi, 2002] and small world networks[Watts and Strogatz, 1998].

While none of these theories has emerged as a clear winner and new ideas of network topologies in large-scale social networks are frequently published, it is important to make simulation tools independent of the models and theories of initial network topology. Furthermore, a simulation tool that is proven and validated through docking and comparison with empirical results can be used as a means to test validity of multiple theories of network topology - or test its own assumptions against all possible networks.

Testing the software on machine-generated data, as opposed to empirical data only, allows the user to conduct repeatable tests that stress certain aspects of the software and help in debugging and optimization of software performance.

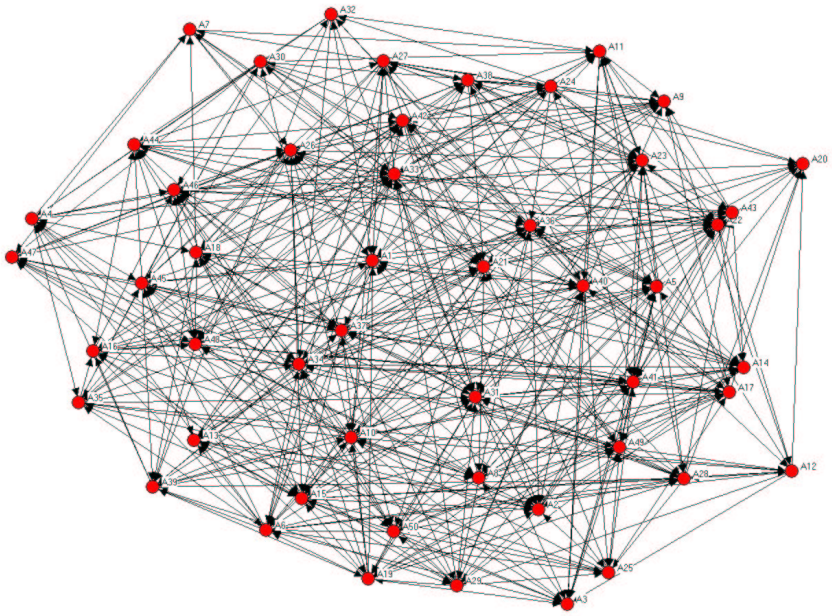


Figure 6.1: A Uniform Random Network

6.0.1 Generating Person-to-Person Networks

Uniform Random Graphs

The study of random graphs dates back to the work of Erdös and Rényi whose seminal papers[Erdös and Rényi, 1960][Erdös and Rényi, 1961] laid the foundation for the theory of random graphs.

There are three standard models for uniform random graphs[Alon and Spencer, 1992]. Each has two parameters. One parameter controls the number of nodes in the graph and one controls the density, or number of edges.

For example, the random graph model $G(n, e)$ assigns uniform probability to all graphs with n nodes and e edges while in the random graph model $G(n, p)$ each edge is chosen with probability p.

Scale-Free Networks

One of the most interesting features of a large class of the complex networks under study now is their scale-free behavior: each node of the network is connected to some other k nodes. The number of connections obeys a power-law distribution, i.e. $P(k) \sim k^{\gamma}, 2 \leq \gamma \leq 3$ for most networks considered.

Such networks are dubbed "scale-free" because the fluctuations of the distribution around the average value k are infinite (they do not possess any particular scale). The difference between a scale-free network and a random network (where every link between different nodes is present with a

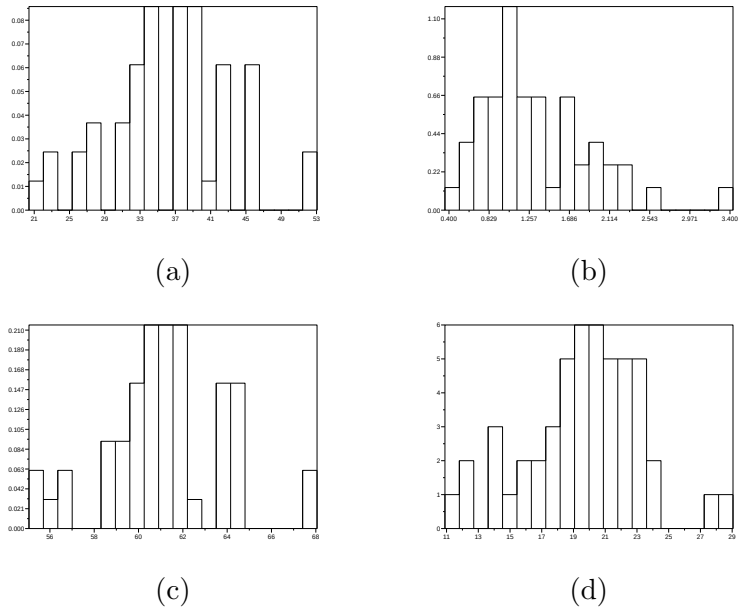


Figure 6.2: Distribution of centralities in a uniform random network: (a)Degree, (b)Closeness, (c)Betweenness, and (d)Eigenvector

probability p, resulting in a Poisson degree distribution) hints towards some mechanisms that generated the observed network features. One of the most celebrated models that explains the emergence of scale-free networks is the Barabasi- Albert (BA) model[Barabási and Albert, 1999].

According to the BA model, the two essential ingredients for the formation of scale-free networks are growth and preferential attachment. Growth implies that new nodes are added to the network over time at a more or less constant rate. Preferential attachment means that a newly added node connects preferentially to nodes that already have a high degree: a new node tries to attach to authoritative nodes and the degree of a node is an effective representation of its authoritativeness. It has been shown that, if the probability to connect to a site is linearly proportional to its degree, then growth and preferential attachment indeed generate scale-free networks[Krapivsky, Redner, and Leyvraz, 2000].

Cellular Networks

The above-mentioned algorithms for generating simulated organizational data can be summarized as creating an approximation of real social phenomena (i.e., organizational structure) by means of an analytically solvable function or a statistical mechanism.

Below we present an alternative approach, which relies on the observa-

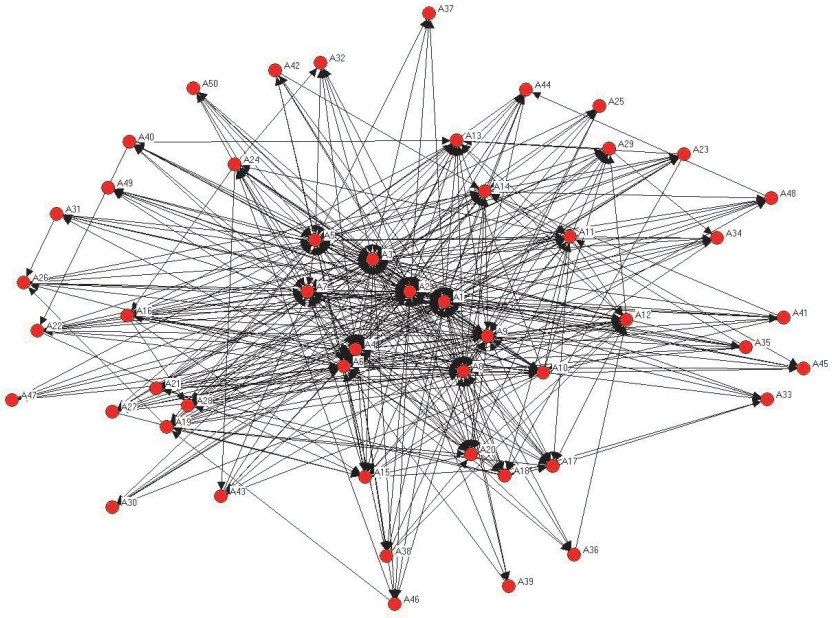
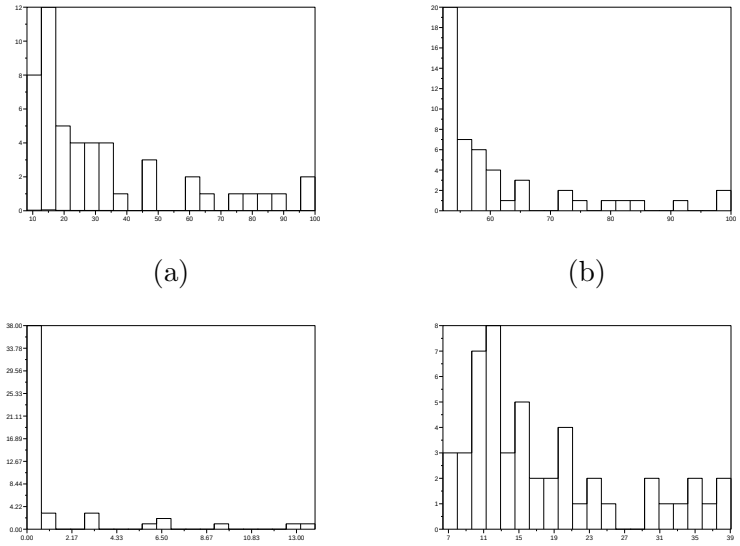


Figure 6.3: A Scale-Free Network generated by preferential attachment



(a) (b)

(c) (d)

Figure 6.4: Distribution of centralities in a scale-free network: (a)Degree, (b)Closeness, (c)Betweenness, and (d)Eigenvector

tions of organizational structure of extant covert networks via creation of a network profile.

We define a generative network profile as a collection of observations and measurements that, when taken together, can be used as a generative function for creating networks similar to ones observed in the real world.

The method of generating simulated organizational structures from profiles should be generalizable to many different types of organizations. However, for every type of organization the components of a generative profile would be different.

In this section we present a generative profile of a cellular covert network based on the publicly available dataset on September 11th hijackers[Krebs, 2001].

Based on publicly available data collected by Krebs[Krebs, 2001], the following profile of the structure of covert networks has been derived [Carley, Lee, and Krackhardt, 2002]:

- The network consists of small cells (mean cell size of 6 members) with very low interconnection between cells.
- Internally, the cells exhibit dense communication patterns.
- There is a very low probability of two individuals communicating by chance (0.007).
- The probability of triad closure (link from x to y being more likely if both x and y are linked to third party z) is 0.181.
- Senior members of each of the cells are often also parts of other cells and interact with other senior members on the network.
- Cell leaders are more knowledgeable than other members.
- Cell members share an ideological doctrine but also specialized knowledge (i.e. bombmakers, drivers, operatives).
- Cells use information technologies and electronic communication.

The aforementioned parameters form a statistical profile from which we can generate simulated organizational networks. The plot on figure 6.5 shows a covert network generated using parameters specified above.

The algorithm for generating a network based on the above profile is represented in listing 6.1

Listing 6.1: "Generating Cellular Networks"

```
//Generate Cells
CREATE cells with
  cell_size()=normally distributed random variable
     (mean=average cell size, std.dev = 0.17*mean);

//Assign agents to cells
FOR all agents DO
  current_cell=random cell
  IF current_cell is not full THEN
    assign an agent to current_cell
  ELSE pick a new cell; repeat this operation.
  END IF
END FOR

//Fill in connections inside cells
FOR all cells DO
  PICK a random agent inside the cell to serve as a leader

  //Internally, generate a uniform network
  FOR all agents inside the cell DO
    generate links within cell with the given density
  END FOR

  //Bring the probability of triad closure in line with the
     measurements
  IF probability of triad closure significantly less then
     measured value
    Add a small random number of edges; repeat the measurements
  ELSE
    Drop a small random number of edges; repeat the measurements
  END IF
END FOR

FOR all cell leaders picked in previous step
  Generate links among cell leaders to produce required inter-
     cell density
END FOR
```

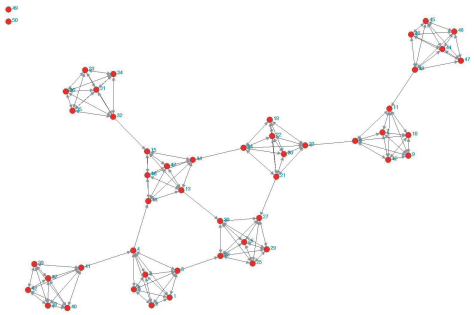


Figure 6.5: Red Team: A Cellular Covert Network

6.0.2 Generalization and Optimization of Network Profiles

At this point, the choice of profile components lies in the hands of the researcher and creation of a profile is a manual task. However, creation of such profiles can be represented as an optimization problem.

Ideally, this profile should form a generative model with the following properties[Chakrabarti, Zhan, and Faloutsos, 2004]:

- *Parsimony:* It would have a small number of parameters;
- *Realism:* It would generate graphs that match the properties of real graphs (degree exponents, diameters, etc.) with the appropriate valued of its parameters;
- *Generation Speed:* It would generate graphs in linear time on number of nodes and edges.

Creation of general-purpose generative profiles can be done with using the following assumptions:

- Let the network consist of a finite number of layered groupings. For example, a corporate network may be viewed as a collection of (a)people, (b) workgroups, (c)departments, (d)divisions, and (e)an entire corporation - resulting in a 5 levels of groupings.
- Assume that groupings at each of the levels (e.g. departments) connect to each other with a network structure that can be expressed with a generative function (unform, scale-free, etc).

A generalized algorithm for generation of complex organization network can be described as a traversal of the hierarchy of layered groupings from most specific to most general while applying a generative function for each of the layers to generate edges at the given layer.

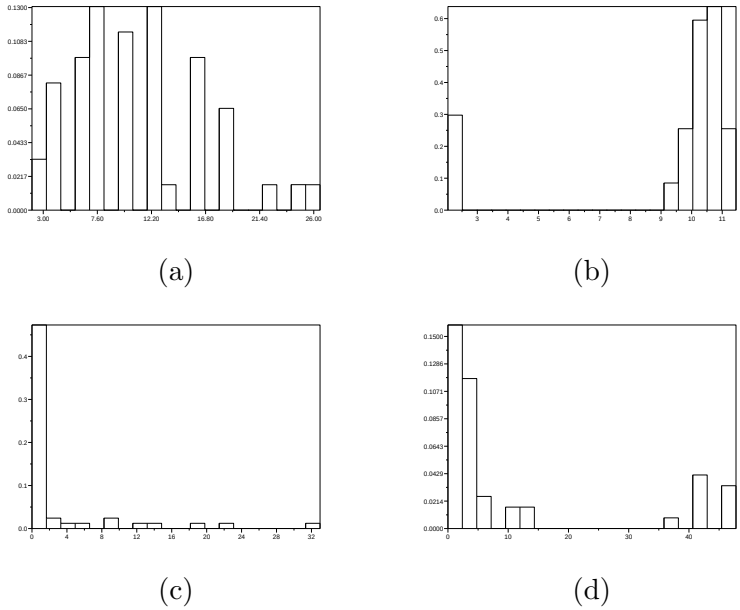


Figure 6.6: Distribution of centralities in a cellular network: (a)Degree, (b)Closeness, (c)Betweenness, and (d)Eigenvector

Thus, generation of a complex network can be parameterized with a profile consisting of (a)number of layers , (b)size of groupings at each layer, and (c)a simple generative function for each layer.

Given that number of simple generative functions is finite such parametrization can be then viewed as an optimization problem, defined as traversal of a state-space of generative profiles and evaluating the fit of each generative profile to a population of known networks.

6.0.3 Generating Knowledge Networks

Knowledge is represented in the MetaMatrix as a set of nodes, with each representing facts or groups of facts. Knowledge that an agent possesses is referred to as an edge between an *Agent* node and a *Knowledge*; knowledge that is required to accomplish an primitive task is represented as an edge between *Task* and *Knowledge* nodes; etc.

Based on data available on structure of terrorist training[Carley, Lee, and Krackhardt, 2002], NetWatch generates agent-knowledge networks using a profile of the knowledge network of a cellular organization.

The knowledge that the agents possess is divided into a three main categories. These categories encompass (a) general doctrine and ideology of the organization, (b) shared training and skills in MO of the organization (e.g. communication procedures, clandestine operations), (c) specialist task-

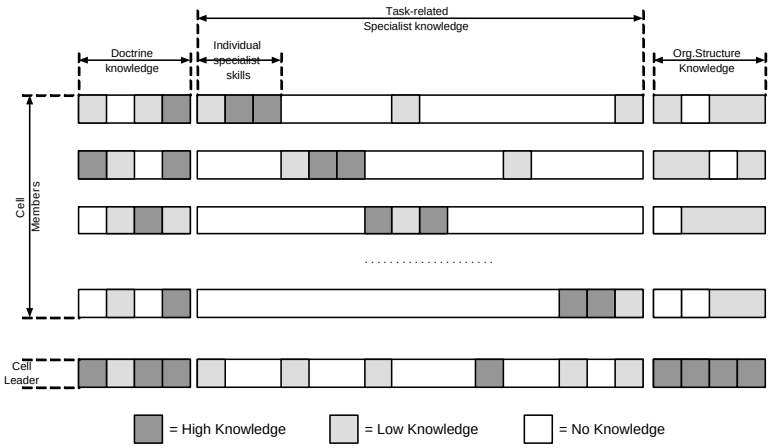


Figure 6.7: Knowledge Distribution of NetWatch Agents

related skills (e.g. bomb-making, sniper skills, getaway car driving), and (c) knowledge of overall organizational structure.

The algorithm for generating the knowledge network presumes the existence of well-formed cells, as generated by the algorithm in section 6.0.1. The following principles are followed:

- Cell leaders are more knowledgeable than other members. As cell leaders are recruited from the ranks of experienced operatives, their doctrinal knowledge is high and they possess many of the shared skills of the other agents. They also possess a small amount of knowledge in each of the specialist areas. This knowledge is not sufficient to replace specialist agents but is sufficient to proficiently delegate subtasks during execution of a complex operation.

- Cell members share an ideological doctrine and a *modus operandi*, further referred to as *"shared knowledge"*. Adherence to a militant ideology is a driving factor in recruiting of operatives in terrorist organizations and is further amplified during training of studies in an a militant religious academy.

  Shared M.O. skills are derived from shared training camp experiences that terrorist organization recruits undergo. Shared skills include communication procedures, clandestine operation skills, preservation of secrecy during planning and preparation of operations.

- Cell members possess specialized knowledge that outlines their specific function within a cell; these facts are further referred to as *"specialist knowledge"*.

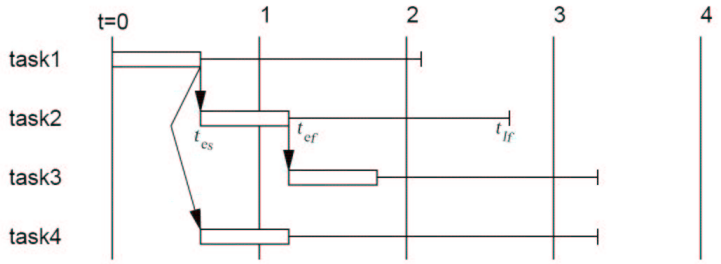


Figure 6.8: Construction of a Task Network as a Precedence Graph

- A specialized portion of the knowledge network deals with overall knowledge of the organizational structure and policies. This knowledge is privileged information distributed only to cell leaders and is further referred to as *privileged knowledge*. However, rank-and-file cell members may obtain small amounts of the privileged information through interaction with other agents outside the primary cell.

The algorithm that generates knowledge networks as outlined above is fairly simple. The knowledge network is divided into portions based on purpose of each fact(e.g. shared knowledge, specialist knowledge, privileged knowledge)(see figure 6.7).

Then, for each agent a_i and fact f_k the algorithm generates a probability $P_{i,k}$ of existence of a an edge $a_i - f_k$ based on the group that the agent belongs to (i.e. cell leader vs. rank-and-file) and what group the fact belongs to (i.e. shared, specialist or privileged).

The edges are then instantiated with a roll of the dice.

Algorithm Parameters

The knowledge network generator depends on the following parameters:

- Proportion of shared knowledge
- Proportion of specialist knowledge
- Proportion of privileged knowledge

6.0.4 Generating Task Structures

The goal of task network generator is to issue complex task specifications such that would engage the planning and task delegation capabilities of agents as described in section 3.9.

The task network consists of a set of *primitive* and *compound tasks* with their precedence relations expressed as $Task-Task$ edges in the MetaMatrix.

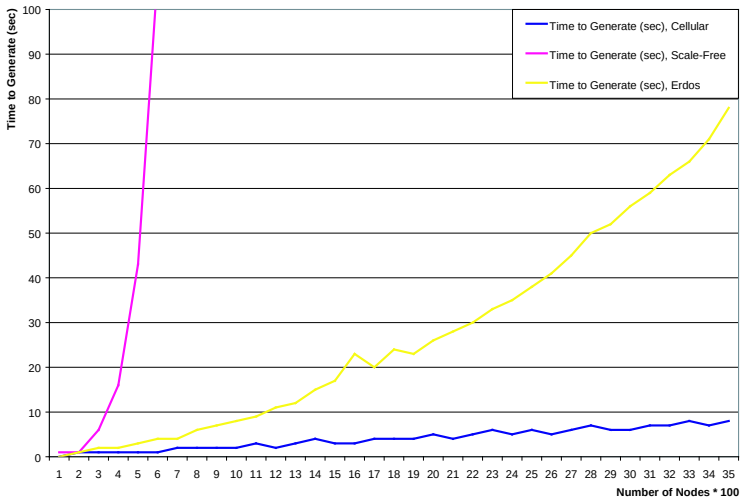


Figure 6.9: Time requirements to generate networks

The complexity of the task network in terms of feasibility of execution can be controlled by varying the *average connectivity* (sum of predecessors and successors) of a task[Collins, Tsvetovat, Mobasher, and Gini, 1998]. This parameter can be essentially thought as controlling the parallelism within the task network.

If the people-to-people network was generated as a cellular network, assignments of people to subtasks ($Person - Task$ edges) are uniformly distributed within each cell. This results in various degrees of subtask difficulty (amount of resource seeking and delegation required to accomplish the task). When people-to-people networks are created as random or scale-free graphs, the task assignments are distributed uniformly throughout the entire network which results in some tasks being not feasible.

6.1 Scalability

To estimate efficiency of the network generation algorithms, I have conducted timing runs of each of the algorithms for generation of people-to-people networks: Erdös random graphs, scale-free networks with preferential attachment, and cellular networks. We varied the size of the network to be generated from 100 to 3500 nodes.

Figure 6.9 shows the time in seconds to generate a network of a given size with each of the algorithms. The least efficient of the algorithms is the preferential attachment algorithm, which grows exponentially. The high computational complexity of the simple preferential attachment algorithm (i.e. not enhanced with heuristics and not parallelized) has been shown by

[Krapivsky, Redner, and Leyvraz, 2000]. Due to the design of the algorithm, it can be broken down into independent local processes, which can be executed by a parallel machine in near linear time [Machta and Machta, 2005]. Use of this algorithm becomes impractical for networks over 2000 agents, where generation of the graph took approximately 10000 seconds, or a little under 3 hours.

Erdös random graphs have been shown[Erdös and Rényi, 1960] to have a quadratic complexity ($\Theta(n^2)$). However, one iteration of edge generation is a very fast operation, so the algorithm remains practical in generating networks of up to 20000 nodes (generation time is 120 seconds).

The cellular network generation algorithm performs in near-linear time due to the fact that cells are small and self-contained. The computational complexity of the cellular network generator is $\Theta(\sigma_{cell}\frac{n}{k}k^2 + \sigma_{intercell}n) = \Theta(\sigma_{cell}nk + \sigma_{intercell}n)$ where n is the number of nodes, k is the mean size of a cell, and σ_{cell} and $\sigma_{intercell}$ are, respectively, densities inside the cell and between cells. Thus, when k is much smaller then n, the complexity of the cellular network generator is close to $\Theta(n)$. In practical terms, this means that even very large networks can be generated in relatively short times, with a 20,000 node network taking less then 20 seconds to generate.

6.2 Conclusion

All of the network generation algorithms described above are used as a testbed for NetWatch, a large-scale multi-agent simulation of covert networks.

While realism of data generated by any of these algorithms can be disputed and nothing is more realistic then empirical data, the use of diverse techniques for generating initial data allows the simulation researcher to test the multi-agent system on networks of widely varying sizes and topologies. Due to small quantities of available empirical data, this is currently not possible to do without resorting to artificially generated data.

As a software engineering tool, the network generation package provides a consistent interface to all of its generation functions - therefore enabling the user (e.g. NetWatch) to test performance of the simulation tools on a wide variety of source networks. This also forces the simulation to remain independent of the initial network topology and thus allow for multi-theory testing of simulation tools.

Chapter 7

Towards an Integrated Analysis Toolchain: Enabling Technologies for Rich Social Network Data

In the past, social network (and more complex social structure) data has been treated as distinct "datasets". Each of these datasets represented a largely self-contained conceptual chunk: a snapshot of a group or organization, frozen in time. As one proceeded with analysis, new aspects of the dataset were born, again largely self-contained but, in a researcher's mind, still connected with the original data.

However, as quality - level of detail and granularity - of social structure data increased (largely facilitated by automated text analysis tools such as AutoMap[Diesner, E.T.Lewis, and K.M.Carley, 2001]), its quantity rapidly increased as well. Furthermore, existing ways of data representation did not fare well with multi-mode matrices or with rich data (where every node and edge carried multiple attributes). Furthermore, as research groups in the field joined in large-scale projects, there arose a need for a well-defined data interchange format.

Thus, the obsolescence of ad-hoc approaches to storage and manipulation of such data has become inevitable. It also became apparent that new ways to query, manipulate and extract subsets of the data were required.

One possible solution to the problem of managing large and heterogenous datasets is to use the notion of a toolchain, as opposed to building large self-contained tools.

The concept of an analysis toolchain is derived from the software engineering concept of development toolchains[Ritter, 1989]. A software development toolchain consists of a number of small self-contained tools such as editors, project management tools, compilers, debuggers and analysis tools such as profilers. While each of these tools is a separate product developed by a different group of people, an implicit agreement on data interchange formats allows users to mix and match tools to create a development environment uniquely suitable to the project at hand. It is also important to note that tools within a toolchain may vary in complexity, size, and features.

In a similar manner, a dynamic network analysis toolchain needs to consist of a number of self-contained tools for gathering, storage, manipulation,

analysis and simulation of dynamic social networks.

The development of the integrated analysis toolset was driven by both a practical and a theoretical goal. From a practical standpoint we wanted to enable the more effective and efficient collection and analysis of network data across space and time to enhance the knowledge about complex social groups as well as to provide support for decision making and action planning regarding such systems.

From a theoretical standpoint, the data and its analysis could be deployed to develop theories or validate hypotheses about the evolution of groups and the co-evolution of types of networks such as social and knowledge networks.

7.1 Requirements for an Interoperable Network Analysis Tool Chain

While the research community has developed a number of very powerful tools for gathering, analyzing and visualizing relational or network data, the tools are rarely interoperable: file import/export options make it possible to use multiple analysis tools within a single project, but data format conversion can be a work and time intensive process. Furthermore, a lack of automation and scripting features does not allow the batch-processing of data and report generation, thus vastly increasing labor and time requirements for the analysis of complex data.

To enhance decision making processes that involve analysis of dynamic and rich network data, one must develop solutions to these drawbacks.

Summarizing the requirements for an interoperable dynamic network analysis toolchain we can identify the following criteria:

1. Network data collected with various techniques, by various people, groups, or agencies and stored or maintained at different sites needs to be represented and stored in a common format to ensure the consistent and compatible representation of various networks or identical networks in various states. Such a common format facilitates data sharing and fusion. Further, having a common format, even if the data is not shared, enables different groups to run the same tools and share the results of the analysis in a meaningful way even when data itself cannot be shared.

2. All tools embedded in a tool chain need to be capable of using (reading/ writing) the same set of data. This means that the output from one tool can be used as input to other tools. This does not mean that each tool needs to use all the data. Each tool can operate on the relevant subset but without altering the basic data format. While not a requirement for a toolchain, it is beneficial from a web-service approach if all interchange among tools is in XML.

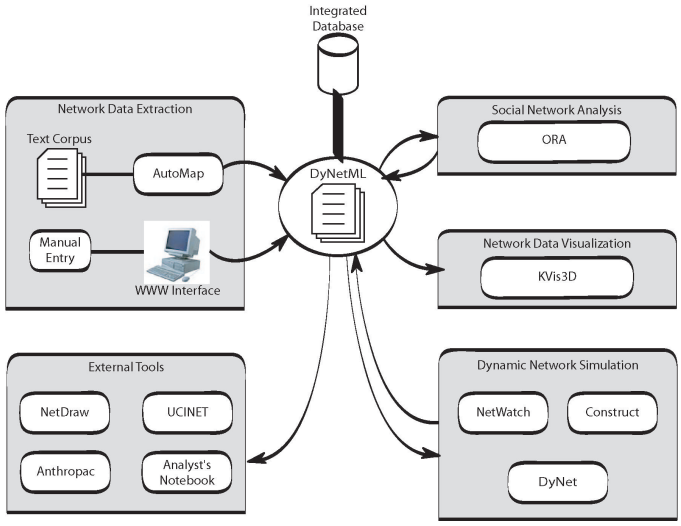


Figure 7.1: Dynamic Networks in DyNetML

3. Inputs to and outputs from one tool chain need to be compatible with external tools or tool chains that provide complementary analytic procedures. This is needed in order to make use of a wider spectrum of analytic power as provided by a variety of social network analysis software, standard statistical tool kits and multi-agent simulation tools.

4. The tools need to scale to large data sets in order to adequately analyze complex systems. Large here means thousands to hundreds of thousands of nodes.

5. The tools need to be robust in the face of missing data. Robustness here means that measures react with little sensitivity to slight modifications of the data.

6. Social and organizational systems need to be represented by a model that captures the entities that such systems are typically composed of, relations between these entities and attributes of the specific entities and relations. Such a model and its implementation need to be expandable as new entity types and relations need to be considered.

7. In addition to the analysis of the relations between and among the entities in the network, the attributes of nodes and relations need to be captured and used to understand the relational data.

The CASOS laboratory has proposed the creation of a seamless analysis toolchain, allowing researchers to mix and match data gathering, analysis and visualization tools interactively and/or to create analysis scripts of batch-mode processing of large datasets or repetition of the same analysis on

different datasets. All tools in such a suite need to be united with a common data interchange format to enable interoperability.

Figure 7.1 shows the relationship of tools developed within and outside the CASOS lab and their integration in a common toolchain.

As part of development of the CASOS toolchain, I have created a number of enabling technologies that contribute towards a creation of an integrated data acquisition, analysis and simulation toolchain. These tools include:

1. DyNetML: a rigorously defined data interchange language that will tie together a heterogenous set of storage, manipulation and analysis tools

2. NetIntel: a database-backed system for easy storage of large quantities of social structure data;

3. a set of data manipulation tools that enable powerful query capabilities

The following chapters discuss the rationale and details of design and implementation of DyNetML and NetIntel, linking the development of these technologies to the creation of a common toolchain.

Chapter 8

DyNetML: A Robust Interchange Language for Rich Social Network Data

Current state of the art in social network data representation presents a fairly bleak picture. Each of analysis and simulation packages uses its own, proprietary and incompatible data format. Some of the file formats do not even have a specification document, making the files unreadable without the software that produced them.

Data formats that were designed for interoperability (such as DL) are rarely expressive enough to fully represent the datasets.

As a result, most researchers are forced to deal with data interchange in a makeshift fashion, at best increasing the workload and at worst resulting in loss of data integrity.

8.1 Requirements for Social Network Data Interchange

1. The data interchange format shall be contained in human-readable text files that are at the same time easily parsable by computers.
2. The data interchange format shall allow an entire dataset, complete with all computed measurements, to be stored in one file.
3. The data interchange format shall provide maximum expressive power to its users, allowing:
    - Typed nodes (types may include "person", "resource", "organization", "knowledge", etc)
    - Multiple sets of nodes of the same type (to express multiple units within the company, etc)
    - Multiple typed attributes per node
    - Typed edges
    - Multiple typed attributes per edge

- Multiple graphs (sets of edges) expressed within the same file
- Dynamic network data expressed in a single file

4. The data interchange format shall allow developers to extend it in a fashion that will not break existing software.

5. The data interchange format shall be flexible enough to be used as both input and output of analysis tools.

Current social network data formats have a number of deficiencies:

Binary files are very difficult to read if exact specification of the file format is not provided. Significant extra efforts are required to keep compatibility with other tools or between versions of the same tool.

Multiple files used for specification of rich data or saving analysis output present a number of problems. First of all, there is a significant potential for data loss due to misplaced or corrupted files (for example, while sent through email). Secondly, a consistent naming scheme for all files and a file catalogue are required to prevent data loss - which requires a certain amount of discipline on the part of the researcher (as these features are not included in the analysis software)

Raw Data file such as binary matrices or edge lists lack the expressiveness required to represent multiple relations between nodes or evolution of social networks over time.

Human-Readable Data in text files or spreadsheets solves the expressivity problem but requires extensive post-processing by hand or with post-processing scripts. However, these programs often represent the weakest link in the software chain (due to hasty design and dependance on outside tools such as Perl or Awk).

General-purpose Graph XML formats such as GraphXML or GXL provide a good approximation of fulfilling the requirements for a data interchange format. They are both human- and machine-readable, and offer extensible capabilities to allow some amount of customization. However, each of the general-purpose formats in existence has a number of individual drawbacks that make it difficult to adapt it for use with social network data.

Table 8.1 shows a comparison of a number of existing formats, both plain-text and XML, in terms of the features required for an expressive interchange format for social network data.

8.1.1 Existing Data formats

The *DL* format supported by UCINET[S.P. Borgatti and Freeman, 2002] is a flexible and human readable data format, and it can contain multiple matrices in a single file. However, the matrices in a DL file must be of a single type.

Format	Parseability	Multi-Mode	Dynamic Graph	Node Attributes	Edge Attributes	Graph-Level Attributes	Attribute Typing
UCINET Native	Proprietary, poorly documented	No	No	In separate files	No	No	Numbers
UCINET DL	Ad-Hoc	No	No	No	Single value	No	Numbers
UCINET VNA	Ad-Hoc	No	No	Extensible	Single value	No	Numbers
PAJEK NET	Rigorous Grammar	No	No	Extensible	Extensible	No	Weak
GraphML	XML	No	Delta	Fixed Set, Graph Drawing	Fixed Set, Graph Drawing	No	Fixed
GraphXML	XML	No	Delta	Extensible	Extensible	No	Weak
GXL	XML	Fixed Node and Edge Types	No	Extensible	Fixed Set	No	String only
ODL	XML	Fixed Node Types, Untyped Edges	No	Extensible	Extensible	No	Strong
DyNetML	XML	Extensible Node and Edge Types	Delta or Full Graph	Extensible	Extensible	Extensible	Strong

Table 8.1: Comparison of Social Network Data Formats

For example, a single DL file cannot contain one matrix containing Agent by Agent data, and another with Agent by Knowledge data. Thus to represent an entire Meta-Matrix, multiple DL files must be used, which increases the likelihood of data inconsistencies.

However, one of the largest drawbacks of DL format is the fact that it is defined in an ad-hoc fashion and lacks a stable grammar. This results in subtle and difficult to detect incompatibilities between tools that use DL as an interchange format.

Moreover, DL files make it very difficult to communicate rich social network data as DL matrices do not support attributes embedded within the data files. While it is possible to communicate dynamic social network data in DL, its handling is somewhat arcane and unstable.

PAJEK[Batagelj and Mrvar, 2003] comes closest to defining a universal interchange format for social network data. PAJEK *.net* format is defined using a rigorous and stable grammar and allows for arbitrary rich data in both nodes and edges.

However, PAJEK files do not have a ready facility for expressing dynamic network data, or for communicating multi-mode and multi-plex networks.

Thus, both of the dominant data formats for social network software have a number of obvious drawbacks that make them unsuitable for communicating large amounts of rich data.

8.1.2 XML-Based Graph Data Formats

The first issue that I address in assessing alternative data formats is that of a rigorous yet expressive grammar. A further requirement is that the new format must be easily implementable for support in various software products.

The XML[XML.org, a] language provides powerful facilities for building expressive and stable data formats. Through use of DTDs (Document Type Definitions) and XML Schemata, it is possible to create a data format that would be both easy to implement and use and provide ample expressive power.

The nature of rich social network data is that it combines traditional attribute based datasets with relational, or graph-based data.

A number of data formats for managing graph-based data, including GraphXML[Herman and Marshall, 2000], GraphML[Brandes, Eiglsperger, Herman, Himsolt, and Marshall, 2002] and GXL[Holt, Winter, and Schürr, 2000], propose XML-based languages for treating graph and relational data.

GraphXML[Herman and Marshall, 2000] is an early XML-based graph language designed to fill the need of graph interchange languages in the graph drawing community. It is essentially a single-purpose language that communicates the graph structure and semi-fixed sets of attributes (such as color, shape and location of nodes). GraphXML is the least flexible of the currently available languages and has been largely superseded by GraphML.

GraphML[Brandes, Eiglsperger, Herman, Himsolt, and Marshall, 2002] is an XML format for generalized graph structures. Its characteristic feature is ability to add modules that implement specific extensions or additional data, such as information related to graph drawing. These modules can be combined or stripped without altering the underlying graph structure, which affords GraphML a large degree of flexibility. GraphML is also the only published format that supports manipulation of dynamic graph data.

The main disadvantage of GraphML is that it requires significant modifications in order to support multi-mode and multi-plex data. Due to its orientation towards the graph drawing community, most of the extensions written to the day are related to graphical representation of graphs and do not contribute to GraphML's ability to express social network data.

Additional advantage of GraphML is that it is supported in large number of graph analysis and drawing software, including yFiles products[yWorks], Ashwood libraries[ObjectStyle, 2005] and in future version of PAJEK[Batagelj and Mrvar, 2003]

GXL[Holt, Winter, and Schürr, 2000][Taentzer, 2001][Winter, 2001] was developed to enable interoperability between software reengineering tools and components. GXL facilitates process of combining single-purpose tools for parsing, source code extraction, architecture recovery, data flow analysis,

pointer analysis, program slicing, query techniques, source code visualization, object recovery, restructuring, refactoring into a single reengineering workbench.

The conceptual data model of GXL is a typed, attributed, directed graph. It can also be used to represent instance data as well as schemas for describing the structure of the data. Moreover, the schema can be explicitly stated along with instance data.

GXL comes very close to fulfilling requirements for social network data interchange format. However, the advantage of enforcing strong typing and directionality of the graph edges comes with a drawback of forcing a particular model of social structure into the data - as many social network measures are not defined on directed graphs.

8.1.3 XML Data Formats for Social Network Data

ODL[Stacy, 2004] (Organizational Design Language) is an XML-derivative language for representing organizational structures developed by Aptima. ODL was designed approximately at the same time as DyNetML, and answers to a very similar set of requirements. ODL represents rich social network data as a collection of typed nodesets, and a set of graphs connecting these nodes with edges.

A characteristic feature of ODL is its use of bindings to represent unknown or probabilistic edges. A binding element acts as a placeholder for zero or more concrete elements. For example, if the sender of a message is known, but the recipient is not, the recipient of the message can be represented by an empty binding element. As a list of probable recipients emerges, they can be added to the binding element complete with their probability of being to the communication.

While ODL is designed for a purpose that is very similar to that of DyNetML, it possesses a number of shortcomings that limit its use for analysis of rich dynamic networks. First of all, ODL does not have a ready representation of dynamic data. Dynamic representations can be achieved by including multiple graphs, each representing a time period, but that technique does not provide a rigorous encapsulation of a multi-mode network changing over time.

A further shortcoming is due to the fact that ODL operates with a fixed set of concept types (Agent, Knowledge, Resource, Task, Event, Communication, Location and Organization). These concept types are not readily extensible, and the methods for handling them are not consistent between types. This places an undue burden on the developer and maintainer of the tools and may cause inconsistencies in data interpretation if ODL was used for data interchange between various software tools.

While use of ODL as an interchange standard may be difficult (both due

to its design and due to the fact that the language is proprietary), translation between ODL and DyNetML can be easily accomplished using XSLT transform mechanism. Such translator was designed by Aptima and is used to bridge their simulation tools to ORA[Carley and Reminga, 2004] and other tools developed at the CASOS lab.

8.1.4 Advantages and Disadvantages of XML-based Data Interchange

The main advantage of expressing complex data as XML is the ability of XML to import hierarchical and object-oriented structures into a human-readable text file. This solves a major representational challenge of other text-based data formats, which are more suitable for representation of flat attribute tables or matrix-based structures.

The representation of complex data structures enabled by XML would be a much lesser advantage if XML did not provide facilities for specifying the schemata for data contained in XML files. However, both DTD and XML Schema facilitate creation of strong grammars that dictate file contents.

The combination of a rigorous markup grammar and data schema facilities reduces the error rate in creation and parsing of XML documents, and provides significant improvement in structure and implementation of third-party code for use with the data format.

Another major advantage of all XML formats is that the rigorous definition of XML grammars allows for easy transformation between various graph formats. Brandes and Lerner[Brandes, Lerner, and Pich, 2004] demonstrate this by applying an XSLT[Wadler, 1999] transformation to convert between GXL and GraphML formats described above.

An XSLT conversion also exists between the DyNetML format described below and the ODL format developed at Aptima. **TODO: MORE HERE**

The major drawback of XML-derived data formats stems from the size and complexity of XML files. The growth in size is dictated by needs for rigorous markup, as every data element requires a number of delimiter tags to describe its function to the parser. While XML files get quite large, they compress very well with any of the currently available algorithms, resulting in average of 90% compression rates.

Furthermore, the memory requirements of XML (and especially DOM[XML.org, b]) parsers are significantly higher then these for parsing simple text-based formats. Use of SAX[XML.org, c] parsers to process large XML files provides significant memory and processing time savings, but adds burden on the software developer as SAX parsers are significantly more difficult to use.

8.2 Extensibility and Modularity of XML Data Formats

A major portion of discussion regarding the future of XML-based data formats for social network data is centered around extensibility of the format for use in and inter-operation of a variety of applications.

A data format specification designed as a common standard, pursues a number of conflicting goals[Stacy, 2004]. The first goal is providing a common language for all tools to be integrated. This goal argues for a least common denominator approach, that is, for a subset of network organizational representations shared by all current and future integrated applications.

The second goal is to provide a comprehensive means for each tool to represent all the data it requires, whether or not any other tool can use that data. This goal argues for a least common multiple approach, that is, for a large set of representations that cover the needs of all integrated applications.

The approach DyNetML takes is a hybrid of the two approaches outlined above. First, DyNetML provides a mechanism for specifying an arbitrary number of typed, named properties at node, edge, node-type, graph, and dataset level. This mechanism is generic and does not violate the overall graph-oriented structure of the data, and thus can be handled by existing tools in a backwards-compatible fashion. This extensibility mechanism serves well for adding custom data points within the graph (e.g. specifying the probability density function of an edge; specifying demographic information on a node).

If the custom data does not fit with the existing graph model (e.g. Anthropac questionnaire data), DyNetML allows for modular extension using auxiliary schemata. This modularity specification follows the W3C XML Schema recommendations[Consorcium, 2004]. A developer extending DyNetML with modular data must include the specification of an extension module with the XSD schema of DyNetML and specify its role in the DyNetML structure via XSD entities.

8.3 DyNetML: A Data Interchange For Rich Social Network Data

To enable interchange and transmittal of rich social structure data, I have designed DyNetML, an XML-based language that fits the requirements outlined above. While DyNetML is still a work in progress, it is slowly gaining industry acceptance as a data interchange format. It is supported by all tools developed at the CASOS laboratory at Carnegie Mellon University, and is a part of ORA, a MetaMatrix-based organizational network analysis tool.

```
<node id="Mohammed_Rashed_Daoud_al-Owhali">
  <properties>
    <property name="knowledgeAccumulated" type="double" value="
        0.7500" />
    <property name="placeOfBirth" type="string" value="Kabul,␣
        Afganistan" />
  </properties>
</node>

<edge source="Mohammed_Rashed_Daoud_al-Owhali" target="
    Usama_Bin_Ladin"
type="double" value="1.000000" />
  <properties>
    <property name="observedOn" type="date" value="01/01/2000" /
        >
    <property name="edgeProbability" type="double" value="0.12"
         />
    <property name="edgeStrength" type="double" value="0.12" />
  </properties>
```

Figure 8.1: Examples of custom properties for a node and an edge

As of August 2004, DyNetML will be natively supported by UCINET, the premier software package for social network analysis. Native support is also under development for a number of software tools at the Department of Defence. Through the use of translation tools, DyNetML is also used by Aptima, University of Connecticut and a number of other companies and institutions.

8.4 DyNetML: an XML-Derived Social Network Language

To address the needs of data interchange and requirements outlined in section 8.1, we have designed DyNetML: an XML-derived language for expressing rich social network data.

The following example illustrates use of DyNetML for representing simple social network datasets (also illustrated on figure 8.2). This and further examples are derived from the Tanzania Embassy Bombing dataset.

8.4.1 DyNetML Format Overview

DyNetML represents dynamic network data as sets of time-slices. Each of the time-slices is a descriptive snapshot of the organization at a given time.

Listing 8.1: A simple network in DyNetML

```
<?xml version = "1.0" encoding = "UTF-8"?>

<DynamicNetwork
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation = "DyNetML.xsd">
<MetaMatrix>
<nodes>
  <nodeset id="agent" type="agent">
    <node id="Mohammed_Rashed_Daoud_al-Owhali"/>
    <node id="Mohammed_Sadiq_Odeh"/>
    <node id="Fazul_Abdullah_Mohammed"/>
    <node id="Wadih_al_Hage"/>
    <node id="Usama_Bin_Ladin"/>
    <node id="Ali_Mohammed"/>
  </nodeset>
</nodes>

<networks>
  <graph id="social_network" sourceType="agent" targetType="
      agent" isDirected="true">
    <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="
        Usama_Bin_Ladin" type="double" value="1"/>
    <edge source="Mohammed_Sadiq_Odeh" target="Wadih_al_Hage"
        type="double" value="1"/>
    <edge source="Fazul_Abdullah_Mohammed" target="Wadih_al_Hage
        " type="double" value="1"/>
    <edge source="Fazul_Abdullah_Mohammed" target="
        Usama_Bin_Ladin" type="double" value="1"/>
    <edge source="Wadih_al_Hage" target="Mohammed_Sadiq_Odeh"
        type="double" value="1"/>
    <edge source="Wadih_al_Hage" target="Fazul_Abdullah_Mohammed
        " type="double" value="1"/>
    <edge source="Usama_Bin_Ladin" target="
        Mohammed_Rashed_Daoud_al-Owhali" type="double" value="1"/
        >
    <edge source="Usama_Bin_Ladin" target="
        Fazul_Abdullah_Mohammed" type="double" value="1"/>
    <edge source="Usama_Bin_Ladin" target="Wadih_al_Hage" type="
        double" value="1"/>
    <edge source="Ali_Mohammed" target="Wadih_al_Hage" type="
        double" value="1"/>
    <edge source="Ali_Mohammed" target="Usama_Bin_Ladin" type="
        double" value="1"/>
  </graph>
</networks>

</MetaMatrix>
</DynamicNetwork>
```

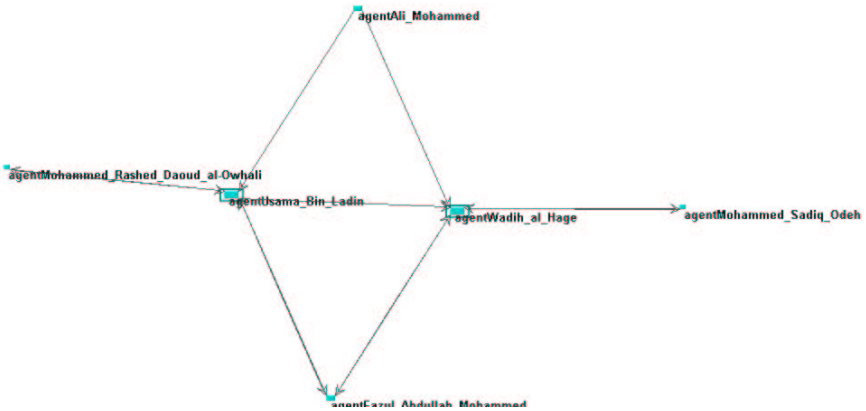


Figure 8.2: A simple network in DyNetML

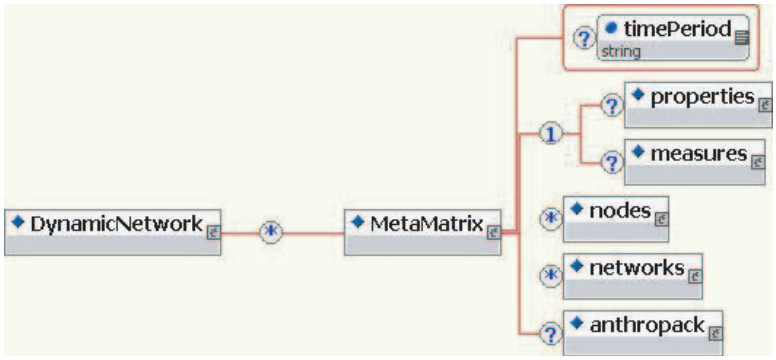


Figure 8.3: Dynamic Networks in DyNetML

Figure 8.3 shows the top-level hierarchy of DyNetML files. A **DynamicNetwork** element is defined as a sequence of **MetaMatrix** elements, each representing a snapshot of the organization for one time period.

Each of the **MetaMatrix** elements consists of:

- an optional **TimePeriod** attribute that allows clear identification of each timeslice.
- a set of **properties** and **measures**, representing data about the whole of the timeslice (see section 8.7 for a complete definition).
- a **nodes** element, containing one or more nodesets (section 8.5.1).
- a **networks** element, containing all networks in this timeslice (section 8.6).
- an **anthropac** element that facilitates linking of network data to anthropological data.

8.5 Representing Multiple Node and Relation Types

While designed predominantly for dealing with social network data, DyNetML format is shaped as a generalized graph data interchange framework.

DyNetML represents graphs as sets of nodes nodes (vertices) and relationships (edges) between them. The node specification allows for detailed specification of each vertex, as well as addition of rich data related to it.

8.5.1 Specifying Individuals and Nodes

Nodes are organized into **nodesets**, which should be thought about as logical groupings of nodes (by type, by affiliation, etc).

Each nodeset has to be identified with a unique **id** attribute (see figure 8.4), and a **type** attribute. For more detailed description of node types, see section 8.5.1.

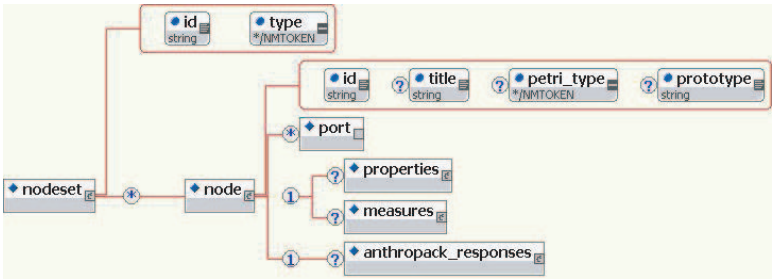
DyNetML allows for an arbitrary number of nodesets (and arbitrary number of nodesets of each type) and an arbitrary number of nodes in each nodeset.

A node specification consists of the following (see figure 8.4):

- **id**: a unique ID *note: it is advisable for ease of searching to use node IDs that do not contain spaces or special characters and are limited in length to 32 characters.*
- **title**: a human-readable title of the vertex (free of restrictions posed on node ID field).
- **prototype**: an optional attribute specifying a parent class of a node. Node prototype can be used to specify additional details about the node.
- Element **port** allows the user to specify inflows and outflows of each node by allowing multiple connection points within each node.
- **Properties** and **Measures** elements allow specification of arbitrary rich data for each node. They are described in more detail in section 8.7.
- **Anthropac** element provides a vehicle for connecting anthropological data with social network data.

Ports and Multiple Connection Points

In order to implement multiple types of connections within the same graph, and to enable use of graphs as nodes of other graphs, we have implemented a system of ports.




```
<node id="Mohammed_Rashed_Daoud_al-Owhali">
  <port name="inbox" port_type="input"/>
  <port name="outbox" port_type="output"/>
  <properties>
    <property name="knowledgeAccumulated" type="double" value="
        0.7500" />
  </properties>
  <measures>
    <measure name="taskExclusivity" type="double" value="0.5017"
        >
      <input id="social_network" />
    </measure>
    <measure name="cognitiveLoad" type="double" value="0.1354">
      <input id="social_network" />
    </measure>
  </measures>
</node>
```

Figure 8.4: Specification of Vertices in DyNetML

A port can be viewed as a point where an edge attaches to a vertex. Thus, a directed edge connecting a port specified as input to another node's port specified as output represents a resource or information flow across the edge.

Since one can specify multiple input and output ports for every node, it is possible to represent a number of distinct flows along every edge while maintaining clear separation between different types of links.

A port is defined as follows (see figure 8.4):

- attribute **name** specifies a port ID that is unique for this node
- attribute **port_type** is a multiple choice, with possible values "input", "output" and "general"

Node Types in DyNetML

DyNetML has been designed to assist the flow of information between software tools by not only enforcing a consistent structured format upon the

data, but also by specifying a constant vocabulary. Since the language has been designed in service of the Social Network Analysis community, we specify a set of standard node types that could be used to express a majority of rich social network data.

The standard node types are: **agent, organization, knowledge, resource, task, location**, and **graph**.

While the architecture of DyNetML allows developers to easily add node types, to ensure inter-operability of tools using DyNetML it is advisable to refrain from expanding the vocabulary unless absolutely required. To provide a more fine-grained node type mechanism, it is best to use the **prototype** attribute of nodes to specify arbitrary subtypes.

8.6 Representing Relations in DyNetML

DyNetML format allows the user to specify multiple graphs within a single framework, including graphs that share vertices with other graphs.

An example for use of such system is the case where a number of individuals are engaged in multiple relationship types - such as the formal network, informal advice network, or familial ties network.

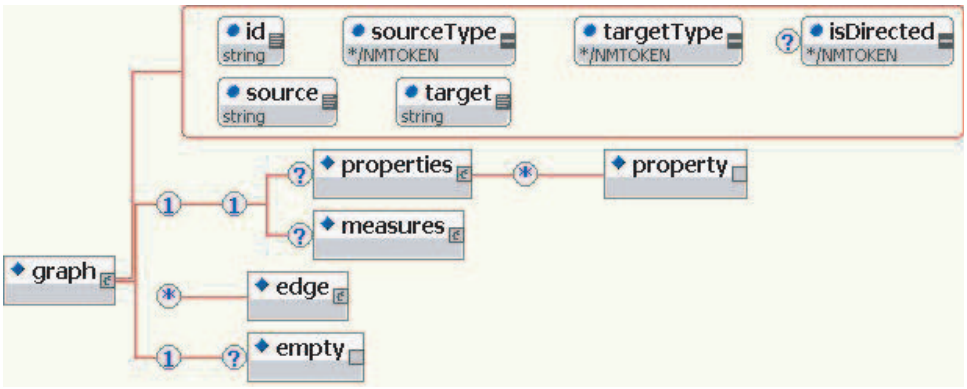
Each graph is specified as follows: (see figure 8.5)

- **id** attribute is the graph's unique ID.
- **source** attribute specifies the nodeset from which the source nodes are taken.
- **sourceType** attribute specifies the type of nodes contained in the source nodeset.
- **target** attribute specifies the nodeset from which the target nodes are taken.
- **targetType** attribute specifies the type of nodes contained in the target nodeset.
- **isDirected** attribute specifies whether the edges of this graph are directed; the attribute can only take values of "true" or "false".

The graph then includes **properties** and **edges** elements (see 8.7), followed by a set of **edge** elements that comprise the actual graph.

8.6.1 Edges

Edges (see figure 8.6) of the graph include the following attributes:

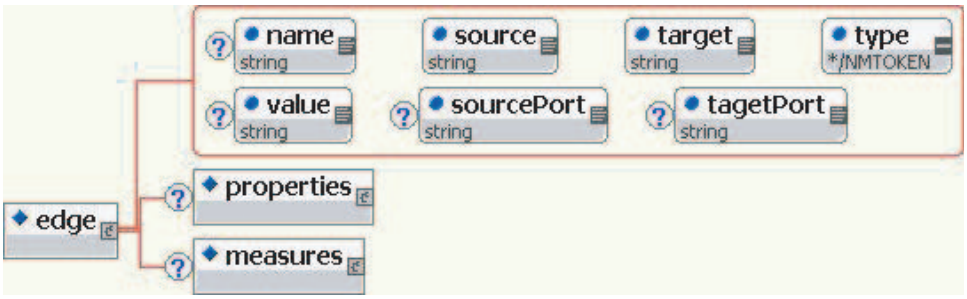



```
<networks>
  <graph sourceType="agent" targetType="agent" id="
      social_network">
    <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="
        Usama_Bin_Ladin" type="double" value="1.000000" />
      <properties>
        <property name="observedOn" type="string" value="
            01/01/2000" />
      </properties>
      <measures>
        <measure name="probability" type="double" value="0.10">
          <input id="social_network" />
        </measure>
    <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="
        Jihad_Mohammed_Ali_(Azzam)" type="double" value="6.000000
        " />
    ...more edges...
  </graph>
  ...more graphs...
</networks>
```

Figure 8.5: Specification of Graphs in DyNetML

- **source** and **sourcePort** attributes specify the source node and port that an edge originates from. The source node should be a part of the nodeset specified in the **source** attribute of the graph. **source** attribute is required; **sourcePort** is optional if no ports have been defined for the source node.

- **target** and **targetPort** attributes specify the source node and port that an edge connects to. The target node should be a part of the nodeset specified in the **target** attribute of the graph. **target** attribute is required; **targetPort** is optional if no ports have been defined for the target node.




```
<edge source="Mohammed_Rashed_Daoud_al-Owhali" sourcePort="
    outbox" target="Usama_Bin_Ladin" targetPort="inbox" type=
    "double" value="1.000000" />
  <properties>
    <property name="observedOn" type="string" value="
        01/01/2000" />
  </properties>
  <measures>
    <measure name="probability" type="double" value="0.10">
      <input id="social_network" />
    </measure>
```

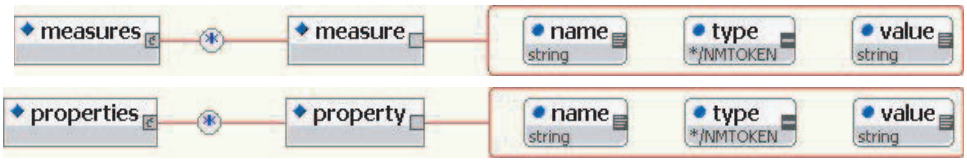
Figure 8.6: Specification of Edges in DyNetML

- **type** attribute is required for every edge. If the edge is unweighted, the type attribute should be set to "binary"; other acceptable edge value types are "double" and "string"
- **value** attribute specifies the edge weight or value; the type of the value should match the type specified in **type** attribute.
- **name** attribute is an optional string that allows the user to a add human-readable title to an edge.
- **properties** and **measures** elements are optional and allow addition of rich data to edge-level specification. A complete description of these elements can be found in section 8.7.

8.7 Representing Graph, Node and Edge Attributes

One of the important facilities of DyNetML is its ability to attach rich data or attributes to every element of the structure.

The rich data, specified as **properties** and **measures**, can be added to the **MetaMatrix**, **node**, **graph** and **edge** objects. The mechanisms for handling the rich data are identical for all objects.




```
<properties>
  <property name="nationality" type="string" value="Saudi" />
  <property name="religion" type="string" value="Muslim" />
  <property name="education" type="string" value="College" />
</properties>
<measures>
  <measure name="taskExclusivity" type="double" value="0.0000">
    <input id="social_network" />
  </measure>
  <measure name="knowledgeExclusivity" type="double" value="
      0.9793">
    <input id="social_network" />
  </measure>
  <measure name="cognitiveLoad" type="double" value="0.0208">
    <input id="social_network" />
  </measure>
</measures>
```

Figure 8.7: Specification of Properties and Measures

Properties and **Measures** objects are syntactically similar (see figure 8.7) and consist of a set of name-value pairs. The main distinction between them is that **Properties** should be thought of as attributes inherent to the subject, such as information obtained from a questionnaire or otherwise known about the subjects.

Measures, on the other hand, are computed by analysis tools and inserted into the dataset during processing.

The guidelines for naming properties and measures are following:

- Names should be descriptive of the nature of data contained within.
- Measure names should include the name of the tool that generated them.

For example, the measure of Freeman centrality computed by NetStat tool should look as:

```
<measure name="netstat_freeman_centrality" type="double"
value="3.14"/>
```

8.8 Complex Social Networks in DyNetML

The basic use of DyNetML is specification of rich social network data, including properties and measures attached to objects within the network. DyNetML also allows for an arbitrary number of network superimposed upon each other, and specification of network data over time.

The example below is a heavily commented small dataset containing two types of nodes (people and facts), and three networks (friendship, advice and knowledge). It is derived from a full MetaMatrix dataset on the terrorist bombing of the U.S. embassy in Tanzania.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MetaMatrix SYSTEM "DyNetML.dtd">



<MetaMatrix timePeriod="1997">




<nodes>
  <nodeset id="agent" type="agent">
    <node id="Mohammed_Rashed_Daoud_al-Owhali"/>
    <node id="Khalfan_Khamis_Mohamed"/>
    <node id="Mohammed_Sadiq_Odeh"/>
    <node id="Ahmed_the_German"/>
    <node id="Fazul_Abdullah_Mohammed"/>
    <node id="Wadih_al_Hage"/>
    <node id="Usama_Bin_Ladin"/>
    <node id="Ali_Mohammed"/>
    <node id="Ahmed_Khalfan_Ghailani"/>
    <node id="Mohammed_Salim"/>
    <node id="al-Fadl"/>
```

Header Information

Time periods can be labelled with arbitrary string labels

Nodes are broken up into nodesets by type (e.g. agent, knowledge, resource, task, etc)

```
  <node id="al-Fawwaz"/>
    <properties>
      <property name="knowledge" type="double" value="0.2500" />
    </properties>
    <measures>
      <measure name="taskExclusivity" type="double" value="0.0000">
        <input id="social_network" />
      </measure>
      <measure name="knowledgeExclusivity" type="double" value="0.9793">
        <input id="social_network" />
      </measure>
      <measure name="cognitiveLoad" type="double" value="0.0208">
        <input id="social_network" />
      </measure>
    </measures>
```

This is a more complex node with properties and attached measures

```
  <node id="Jihad_Mohammed_Ali_(Azzam)"/>
  <node id="abouhalima"/>
  <node id="Abdullah_Ahmed_Abdullah_(Saleh)"/>
  <node id="Abdal_Rahman"/>
</nodeset>
```

```
<nodeset id="type" type="task">
  <node id="surveillence"/>
  <node id="weapon_training"/>
  <node id="driving_training"/>
  <node id="bomb_prep"/>
  <node id="bombing"/>
</nodeset>
```

"TASK" nodeset outlines the types of tasks that the group performs

```
  <nodeset id="resource" type="resource">
    <node id="building_for_bombmaking"/>
    <node id="money"/>
    <node id="bomb_material"/>
    <node id="truck"/>
  </nodeset>
```

"RESOURCE" nodeset: tangible resources for functioning of organization

```
  <nodeset id="knowledge" type="knowledge">
    <node id="religious_extremism"/>
    <node id="weapons_expertise"/>
    <node id="surveillance_expertise"/>
    <node id="media_consultant"/>
  </nodeset>
```

"KNOWLEDGE": non-tangible or mental resources of the organization

```
</nodes>
```

Now we specify the graphs that comprise the metamatrix.

```
<networks>
  <graph id="social_network" sourceType="agent" targetType="agent" isDirected="true">
    <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="Usama_Bin_Ladin" type="
        double" value="1"/>
    <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="Jihad_Mohammed_Ali_(Azzam)
        " type="double" value="1"/>
    <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="Abdullah_Ahmed_Abdullah_(
        Saleh)" type="double" value="1"/>
    <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="Abdal_Rahman" type="double
        " value="1"/>
    <edge source="Mohammed_Sadiq_Odeh" target="Wadih_al_Hage" type="double" value="1"
        />
    <edge source="Ahmed_the_German" target="Abdullah_Ahmed_Abdullah_(Saleh)" type="
        double" value="1"/>
    <edge source="Fazul_Abdullah_Mohammed" target="Wadih_al_Hage" type="double" value
        ="1"/>
    <edge source="Fazul_Abdullah_Mohammed" target="Usama_Bin_Ladin" type="double"
        value="1"/>
    <edge source="Wadih_al_Hage" target="Mohammed_Sadiq_Odeh" type="double" value="1"
        />
    <edge source="Wadih_al_Hage" target="Fazul_Abdullah_Mohammed" type="double" value
        ="1"/>
    <edge source="Wadih_al_Hage" target="Usama_Bin_Ladin" type="double" value="1"/>
    <edge source="Usama_Bin_Ladin" target="Mohammed_Rashed_Daoud_al-Owhali" type="
        double" value="1"/>
    <edge source="Usama_Bin_Ladin" target="Fazul_Abdullah_Mohammed" type="double"
        value="1"/>
    <edge source="Usama_Bin_Ladin" target="Wadih_al_Hage" type="double" value="1"/>
    <edge source="Ali_Mohammed" target="Wadih_al_Hage" type="double" value="1"/>
    <edge source="Ali_Mohammed" target="Usama_Bin_Ladin" type="double" value="1"/>
    <edge source="al-Fawwaz" target="Wadih_al_Hage" type="double" value="1"/>
    <edge source="al-Fawwaz" target="Usama_Bin_Ladin" type="double" value="1"/>
    <edge source="Jihad_Mohammed_Ali_(Azzam)" target="Mohammed_Rashed_Daoud_al-Owhali
        " type="double" value="2"/>
    <edge source="abouhalima" target="Wadih_al_Hage" type="double" value="1"/>
    <edge source="Abdullah_Ahmed_Abdullah_(Saleh)" target="Mohammed_Rashed_Daoud_al-
        Owhali" type="double" value="1"/>
    <edge source="Abdullah_Ahmed_Abdullah_(Saleh)" target="Mohammed_Sadiq_Odeh" type=
        "double" value="1"/>
    <edge source="Abdullah_Ahmed_Abdullah_(Saleh)" target="Abdal_Rahman" type="double
        " value="1"/>
    <edge source="Abdal_Rahman" target="Mohammed_Rashed_Daoud_al-Owhali" type="double
        " value="1"/>
    <edge source="Abdal_Rahman" target="Abdullah_Ahmed_Abdullah_(Saleh)" type="double
        " value="1"/>
  </graph>
```

This graph specifies the social network of the organization, i.e. who knows or speaks to whom.
NOTE: source and target of each edge should be a valid node; however, it's up to the software developer to ensure that, or to check consistency in any code that imports this data

```
<graph id="knowledge_network" sourceType="agent" targetType="knowledge">
  <properties>
    <property name="title" type="string" value="this␣graph␣specifies␣who␣knows␣what
        ">
  <properites>
  <measures>
    <measure name="centralization" type="double" value="0.023">
    <measure name="density" type="double" value="0.45">
  <measures>
  <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="religious_extremism" type=
      "double" value="1"/>
  <edge source="Khalfan_Khamis_Mohamed" target="weapons_expertise" type="double"
      value="1"/>
  <edge source="Khalfan_Khamis_Mohamed" target="surveillance_expertise" type="
      double" value="1"/>
  <edge source="Mohammed_Sadiq_Odeh" target="religious_extremism" type="double"
      value="1"/>
  <edge source="Fazul_Abdullah_Mohammed" target="religious_extremism" type="double"
       value="1"/>
  <edge source="Wadih_al_Hage" target="religious_extremism" type="double" value="1"
      />
  <edge source="Ali_Mohammed" target="surveillance_expertise" type="double" value="
      1"/>
  <edge source="Mohammed_Salim" target="religious_extremism" type="double" value="1
      "
  <edge source="al-Fadl" target="religious_extremism" type="double" value="1"/>
  <edge source="Jihad_Mohammed_Ali_(Azzam)" target="religious_extremism" type="
      double" value="1"/>
  <edge source="Abdullah_Ahmed_Abdullah_(Saleh)" target="surveillance_expertise"
      type="double" value="1"/>
  <edge source="Usama_Bin_Ladin" target="media_consultant" type="double" value="1"/
      >
  <edge source="al-Fawwaz" target="media_consultant" type="double" value="1"/>
</graph>
```

Specifies who in the organization knows or has trained in which areas of knowledge This graph has a number of associated measures and a text property

```
<graph id="resource_network" sourceType="agent" targetType="resource" isDirected="
    true">
  <edge source="Khalfan_Khamis_Mohamed" target="building_for_bombmaking" type="
      double" value="1"/>
  <properties>
    <property name="edgeDescription" type="string" value="propertyOwnership">
  <properites>
  <measures>
    <measure name="probability" type="double" value="0.85">
  <measures>
  <edge source="Mohammed_Sadiq_Odeh" target="building_for_bombmaking" type="double"
       value="1"/>
  <edge source="Khalfan_Khamis_Mohamed" target="building_for_bombmaking" type="
      double" value="1"/>
  <properties>
    <property name="edgeDescription" type="string" value="propertyLease">
  <properites>
  <measures>
    <measure name="probability" type="double" value="0.4">
  <measures>
  <edge source="Fazul_Abdullah_Mohammed" target="building_for_bombmaking" type="
      double" value="1"/>
  <edge source="Wadih_al_Hage" target="bomb_material" type="double" value="1"/>
  <edge source="Usama_Bin_Ladin" target="money" type="double" value="1"/>
  <edge source="Ahmed_Khalfan_Ghailani" target="truck" type="double" value="1"/>
  <edge source="Mohammed_Salim" target="money" type="double" value="1"/>
  <edge source="Abdullah_Ahmed_Abdullah_(Saleh)" target="building_for_bombmaking"
      type="double" value="1"/>
  <edge source="Abdal_Rahman" target="bomb_material" type="double" value="1"/>
</graph>
```

Relationship between people and tangible resources of the organization
This graph illustrates use of edge properties to represent arbitrary rich data

Assignment of people to tasks

```
<graph id="people_task" sourceType="agent" targetType="task">
  <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="surveillence" type="double
      " value="1"/>
  <edge source="Khalfan_Khamis_Mohamed" target="weapon_training" type="double"
      value="1"/>
  <edge source="Mohammed_Sadiq_Odeh" target="weapon_training" type="double" value="
      1"/>
  <edge source="Ahmed_the_German" target="bombing" type="double" value="1"/>
  <edge source="Wadih_al_Hage" target="weapon_training" type="double" value="1"/>
  <edge source="Ali_Mohammed" target="surveillence" type="double" value="1"/>
  <edge source="Ahmed_Khalfan_Ghailani" target="bomb_prep" type="double" value="1"/
      >
  <edge source="Mohammed_Salim" target="weapon_training" type="double" value="1"/>
  <edge source="al-Fadl" target="weapon_training" type="double" value="1"/>
  <edge source="Jihad_Mohammed_Ali_(Azzam)" target="surveillence" type="double"
      value="1"/>
  <edge source="Abdullah_Ahmed_Abdullah_(Saleh)" target="bomb_prep" type="double"
      value="1"/>
  <edge source="Abdal_Rahman" target="weapon_training" type="double" value="1"/>
</graph>
```

Resource requirements for tasks

```
<graph id="resource_task" sourceType="resource" targetType="task">
  <edge target="weapon_training" source="weapons_expertise" type="double" value="1"
      />
  <edge target="driving_training" source="weapons_expertise" type="double" value="1
      "/>
  <edge target="bomb_prep" source="weapons_expertise" type="double" value="1"/>
  <edge target="bombing" source="religious_extremism" type="double" value="1"/>
  <edge target="bombing" source="weapons_expertise" type="double" value="1"/>
</graph>
```

```
  <graph id="knowledge_task" sourceType="knowledge" targetType="task">
    <edge source="religious_extremism" target="surveillence" type="double" value="1"/
        >
    <edge source="weapons_expertise" target="surveillence" type="double" value="1"/>
    <edge source="surveillance_expertise" target="surveillence" type="double" value="
        1"/>
    <edge source="media_consultant" target="surveillence" type="double" value="1"/>
    <edge source="weapons_expertise" target="weapon_training" type="double" value="1"
        />
    <edge source="surveillance_expertise" target="weapon_training" type="double"
        value="1"/>
    <edge source="weapons_expertise" target="driving_training" type="double" value="1
        "/>
    <edge source="weapons_expertise␣target="bomb_prep"␣type="double"␣value="1"/>
␣␣␣␣<edge␣source="surveillance_expertise"␣target="bombing"␣type="double"␣value="1"/>
␣␣␣␣<edge␣source="surveillance_expertise"␣target="bombing"␣type="double"␣value="1"/>
␣␣</graph>
```

Knowledge requirements for tasks

```
  <graph id="task_network" sourceType="task" targetType="task" isDirected="true">
    <edge source="bomb_prep" target="weapon_training" type="double" value="1"/>
    <edge source="bombing" target="driving_training" type="double" value="1"/>
    <edge source="bombing" target="bomb_prep" type="double" value="1"/>
    <edge source="bombing" target="surveillence" type="double" value="1"/>
  </graph>
```

Precedence relationship of tasks in the grand task of the organization (i.e. large-scale attack)

```
</networks>
</MetaMatrix>
```

End of example

Chapter 9

Storage and Manipulation of Social Structure Data

9.1 Rationale for Building a Social Network Database

Social network analysis focuses on the relations among and between entities in a social or organizational system. For most of its history, social network analysis has operated on a notion of a *dataset* - a clearly delimited and pruned set of observations that have been encoded and parameterized using a particular set of assumptions or policies. In such a dataset the entities are represented as nodes, and the relations between them as edges or links. The datasets were often painstakingly collected by hand over long periods of time and pruned to illustrate a phenomenon or hypothesis with the greatest possible clarity. Thus, traditional SNA datasets came to be viewed as self-contained units of data, potentially with some shared characteristics (such as data collection or encoding methods) but also with potential for vastly different assumptions at any stage of the process.

Each dataset, as defined in the SNA terms, represents a largely self-contained conceptual chunk: a snapshot of a social system at one point in time. As one proceeds with analysis, new measures computed on the dataset are introduced, again largely self-contained but, except in a researcher's mind, still unconnected with the original data in a except in assorted ad-hoc fashions. Over the past ten years there have been a growing number of studies in which the researchers made use of the social network and attributes or networks at two or more points in time.

Social network data can be multi-mode (various types of relationships such as friendship, kinship), multi-link (connections across various meta-matrix entities) and multi-time period. Furthermore, nodes and edges can have multiple attributes such as the formal position of an employee in a company or the types of relationships between employees (multi-mode). We refer to data that is multi-mode, multi-link and multi-time period in which both nodes and edges can have attributes that carry information on to how to interpret, evolve, and impact these nodes and edges as "rich" data.

With the advent of tools for automated gathering of relational data such

as AutoMap[Diesner and Carley, 2004][Diesner and Carley, 2005] and computational models including multi-agent social-network simulation tools such as Construct[Schreiber and Carley, 2004] and NetWatch[Tsvetovat and Carley, 2004], the sheer quantity of available data has increased exponentially. Moreover, most of the data gathered, processed, or simulated in this manner is rich network data of groups with hundreds or thousands of actors.

Ad-hoc approaches to storage and manipulation of such data significantly increase the labor burden upon the researcher, and it became apparent that new ways to query, manipulate and extract subsets of the data were required. Furthermore, as research groups in the field joined in large-scale projects, there arose a need for a well-defined data interchange format.

There is a pressing need to automatically collect data on social systems as rich network data, analyze such systems to find hidden relations and groups, prune the datasets to locate regions of interest, locate key actors, characterize the structure, locate points of vulnerability, compare and contrast alternative networks, visualize the structure of a system as a whole or in part and simulate change in a system as it evolves naturally or in response to strategic interventions over time or under certain impacts, including modification of data. To meet this challenge, we need to move beyond the traditional approach[Carley, 2002c].

The amount and quality of data collected by the automated data gathering tools and simulation tools suggest that a relational database (RDBMS) tool needs to be used to manage and query the data. However, dealing with large quantities of network data presents a number of unique challenges from the data warehousing point of view.

I discuss the design and construction of NetIntel[Tsvetovat, Diesner, and Carley, 2005] - an RDBMS-based system for warehousing, merging and manipulating large network datasets.

9.2 Databases and Gathering of Network Intelligence

In the aftermath of the September 11th attacks, it was noted that coherent information sources on terrorism and terrorist groups were not available to researchers[Gruenwald, McNutt, and Mercier, 2003]. Information was either available in fragmentary form, not allowing comparison studies across incidents, groups or tactics, or made available in written articles - which are not readily suitable for quantitative analysis of terrorist networks. Data collected by intelligence and law-enforcement agencies, while potentially better organized, is largely not available to the research community due to restrictions in distribution of sensitive information.

To counter the information scarcity, a number of institutions developed unified database services that collected and made available publicly accessible information on terrorist organizations. This information is largely collected from open source media, such as newspaper and magazine articles, and other mass media sources.

Such open-source databases include:

- RAND Terrorism Chronology Database[Corporation, 2003] - including international terror incidents between 1968 and 1997
- RAND-MIPT (Memorial Institute for Prevention of Terrorism) Terrorism Incident Database[Houghton, 2002], including domestic and international terrorist incidents from 1998 to the present
- MIPT Indictment Database[Smith and Damphousse, 2002] - Terrorist indictments in the United States since 1978.

  Both RAND and MIPT databases rely on publicly available information from reputable information sources, such as newspapers, radio and television.

- IntelCenter Database (ICD)[IntelCenter, 2005] includes information on terrorist incidents, groups and individuals collected from public sources, including not only traditional media outlets and public information (such as indictments), but also information learned from Middle East-based news wire services. Separately, IntelCenter also collects information from Arabic chat-rooms and Internet-based publications - although value of such data is questionable and data may be tainted by propaganda.

The focus of these databases is the agglomeration of publicly available data and dissemination of it to researchers, both in the public and private sectors. Little of the work in large public databases has been focused on enabling social network analysis or link analysis of covert and terrorist networks. The IntelCenter Corporation has released a dataset mapping relationships between members of Al-Qaida[IntelCenter.com, 2003]. However, that dataset was delivered as virtually a read-only diagram that did not facilitate quantitative analysis of the data.

Furthermore, the data in the above databases is frequently presented in a proprietary format, making it difficult to employ other software for analysis purposes.

On the commercial software frontier, I2 Corporation has been marketing Analyst's Notebook[Corporation, 2005], a software product for integrating and charting network-based intelligence on criminal and terrorist organizations. This software is in use in many governmental and law enforcement

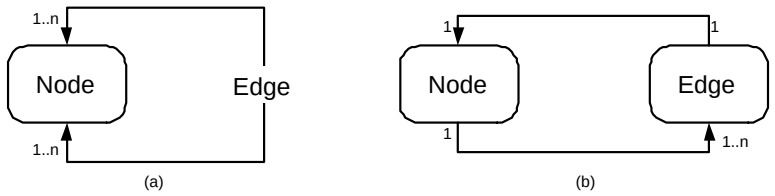


Figure 9.1: Schemata for Graph Data: (a)Simple relations, (b)Rich relational data

agencies, and allows significant integration with data collection, communication and other technologies. A separate product, *iBase*, provides shared storage and data integration facilities of the product. However, the product implements very few quantitative analysis tools and does not allow ready export of network data into analysis packages.

The goal of the NetIntel project, as described in this chapter, is to provide a means for ready collection and integration of network data, with an emphasis on making the data available for quantitative analysis with standard software tools, and making the database accessible on the basis of open standards for database connectivity.

9.3 Existing Work

Storing and manipulating massive persistent graph data is a non-trivial preposition. Despite the fact that majority of data captured by businesses and organizations is relational in nature and can be efficiently described in terms of graphs, much of database and data mining research in the past decade has concentrated on propositional data[Neville and Jensen", 2002].

In propositional data, instances and objects are assumed to be identical and independently distributed (*i.i.d.*). Relational data violates this assumption. Relationships among objects reflect dependence among instances, and the instances themselves are heterogenous. Rich social network data, such as information extracted via text analysis, further supports this fact by attaching semantics and attribute sets to both instances and the relations themselves.

Further, even the part of the data mining community that routinely deals with relational data has focused on learning patterns from the data and structure of relations. An important, but oft-overlooked aspect of storing relational data is that of data selection and transformation.

The basic schemata for storing graph data in a relational database (RDBMS) (figure 9.1(a)) is a many-to-many relation recursively defined on a table. Figure 9.1(b) illustrates a slightly more complicated scenario, turning edges from a simple relation to an object that can contain rich attribute sets.

To extract subsets of a large graph both of these schemata in a query language such as SQL, it is necessary to build a recursive **JOIN** operator. However, it should be noted that SQL[Melton, 1996] explicitly disallow recursive **JOIN**s, and thus lack facilities for ready implementation of graph data manipulation.

A number of research systems, such as Lorel[Abiteboul, Quass, McHugh, Widom, and Wiener, 1997] and QGraph[Blau, Immerman, and D.Jensen, 2002], solve this problem by building graph-based DBMS systems from scratch or building on top of object-oriented storage mechanisms, and adding a graph query language.

Lorel[Abiteboul, Quass, McHugh, Widom, and Wiener, 1997] language, while not specifically designed for manipulation of social network data, is built using a graph metaphor as the underlying data model and thus suitable for storage of relational data. It also introduces existential quantifiers that allows selection of graph subsets based on existence (as opposed to properties) of their relations. However, Lorel can be only implemented on top of an experimental *Lore*[Jennifer Widom, 2004] DBMS - which is no longer maintained and thus not suitable for use in a production system.

QGraph[Blau, Immerman, and D.Jensen, 2002] presents a visual language specifically designed for querying and updating graph databases. A key feature of QGraph is that the user can draw a query consisting of vertices and edges with specified relations between their attributes. The response will be the collection of all subgraphs that have the desired pattern. QGraph implements an advanced set of quantifiers that allows the user to specify not only existence of relations, but also the cardinality of edge subsets, and also place conditional operators on both nodes and edges. QGraph is well-suited for dealing with general graph data, and can be well adapted for storing and retrieving complex social network data.

However, the graphical nature of the language that is one of its selling features for a stand-alone DBMS, presents a significant disadvantage for integration of QGraph into an analysis toolchain such as one described in chapter 7. QGraph also shares Lorel's disadvantage of being built on top of an experimental storage engine.

9.4 Storage Requirements for Social Structure Data

Mindful of the limitations of both traditional RDBMS systems and experimental graph query languages, we suggest that a scalable solution to storage and manipulation of graph-based data is not via creation of custom database systems or query languages, but rather via extension of database tools found and widely used in the industry.

The structure of the database shall be defined in a way that preserves the character and integrity of the data (i.e. is aware of underlying graph properties of the data). The structure shall be designed in an extensible manner, allowing easy addition of new attributes, node and edge types.

The database system shall not only keep track of the units of social structure data (such as nodes and edges) but also sources of such, enabling the creation of large-scale multi-source datasets while preserving the original data sources.

The database shall have an easy-to-use web-based interface, allowing users to enter, search and edit data as well as access manipulation and query capabilities described below.

9.5 Requirements for Data Manipulation Tools

The data manipulation tools shall be closely coupled to the database system described above. The foremost requirement for the subsystem is the ability to extract subsets of the data based on:

- the source (or sources) of data (e.g. "Find all social structure data that came from New York Times" or "Find all data that came from New York Times article from 10/10/2003")
- attributes of nodes and edges (e.g. "What is the network of people who were born in Syria?")

The manipulation tools shall include both existential, universal and cardinality quantifiers for specifying structure of subgraphs.

The manipulation tools shall be able to extract subsets of the network based on graph-theoretic properties of the network such as graph distance (e.g. "Find all nodes at a graph distance of 2 or less from a given node") and graph density (e.g. "Find all nodes embedded in subgraphs with given density").

The manipulation tools shall allow easy completion of incomplete datasets (e.g. "Given a set of people, find all organizations and resources connected to them")

The query tools shall enable the creation of time-slices from the complete dataset of any subset thereof, if time-dependent data is present.

Finally, the query tools shall be easily combined into scripts, resulting in extremely powerful structure-aware data manipulation capability.

9.6 Database Design

A DBMS chosen for the underlying storage engine must feature advanced query capabilities (both SQL and procedural), as well as stored procedure and trigger capabilities. For the current implementation, we have chosen PostgreSQL[PostgreSQL, 2005] database. Of the databases available under the GPL license, it offers the most complete implementation of ANSI SQL, and an implementation of PL/SQL - a procedural language that is portable to other industry database systems such as Oracle.

The majority of the data manipulation tools are implemented as functional extension to SQL and thus available within standard SQL queries. Two end-user interfaces to the database (a command-line system and a web-based interface) are also awailable. These interface serve as a means to import raw data from data gathering tools such as AutoMap and export data into analysis tools such as ORA.

The Web-based interface allows easy navigation and editing of large bodies of data, as well as some access to data manipulation and query tools.

9.6.1 Database Schema

The database schema is designed to preserve flexibility inherent in the source data while enforcing some regularity upon the datasets.

2 tables, **Node** and **Edge**, compose the basic graph structure. Two separate tables contains a set of Node Types and Edge Types, thus making the graph structure in the database semantically extendable.

A **Document** table stores data sources that contribute to the creation of the database and links them through many-to-many relations to the graph entities (Nodes and Edges).

A set of tables store domain-dependent data on each of the node types. These tables are not static in the database schema, but rather created automatically at the same time as a new Node Type, thus ensuring both flexibility and referential integrity of data.

9.6.2 Thesaurus

Due to the fact that data for the database comes from many disparate sources and includes many foreign names, alternative spellings of such names are inevitable.

The database uses a separate **Thesaurus** table to store alternative spellings of names of entities. When an entity (Node or Edge) is inserted, queried or updated, a *Trigger Function* checks spelling of the entity's name or ID and makes sure that the ID is spelled in a canonical way within the dataset.

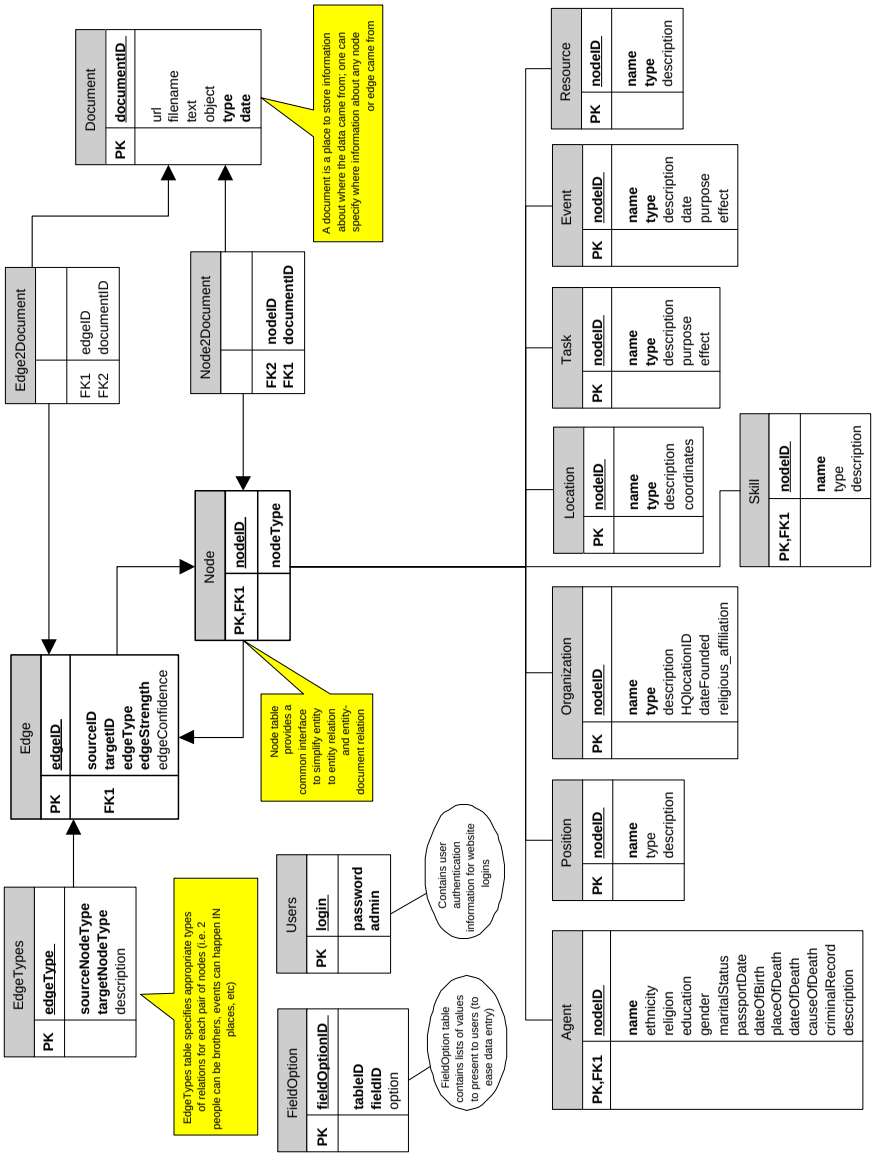


Figure 9.2: NetIntel Database Schema

Unfortunately, the data populating the Thesaurus table had to be compiled by hand. However, with a simple conversion tool, NetIntel can make use of thesauri written for use with AutoMap and can therefore capitalize on the manual work that was invested in their creation.

9.7 Data Manipulation

The data manipulation tools are closely coupled to the database system described above. The foremost requirement for the subsystem is the ability to extract subsets of the data based on:

- the source (or sources) of data (e.g. "Find all social structure data that came from New York Times" or "Find all data that came from New York Times article from 10/10/2003").
- attributes of nodes and edges (e.g. "What is the network of people who were born in Syria?").

The query tools enable the creation of time-slices from the complete dataset of any subset thereof if time-dependent data is present.

The above queries are easily accomplished using SQL and the Document tracking tables. This query has been implemented as a part of **db2dynetml** export program and is useable with a single command-line option.

9.7.1 Graphs in NetIntel Database

Let the graph be defined as

$$G = (V, E) : \begin{cases} V = \{v_i : [nodeid, nodeType, attributes]\} \\ E = \{e_i : [source, dest, edgeType, attributes]\} \end{cases} \quad (9.1)$$

where V is the set of graph vertices and E is the set of graph edges. For purpose of database storage, vertices and edges are stored in relational database tables.

9.7.2 Graph Subsets

The manipulation tools are designed to extract subsets of the network based on graph-theoretic properties of the network such as graph distance (e.g. "Find all nodes at a graph distance of 2 or less from a given node") and graph density (e.g. "Find all nodes embedded in subgraphs with given density").

Graph subsets are defined by a union of two conditional operators - the vertex condition and the edge condition:

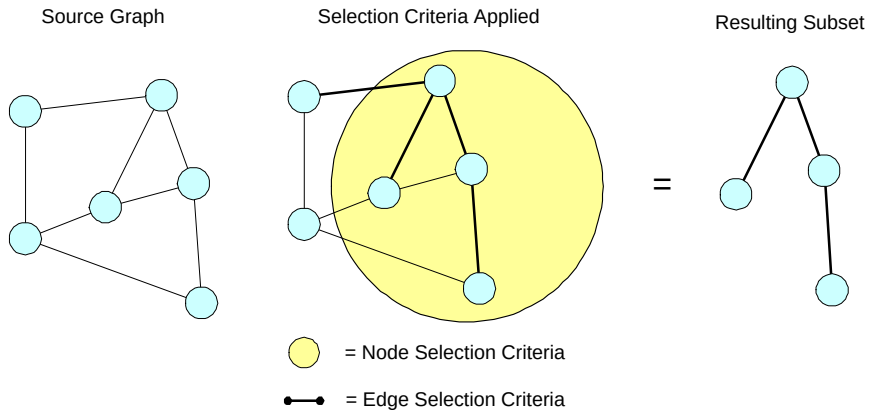


Figure 9.3: Graph Subset Creation

$$g_{CV,CE} = \begin{cases} v_{CV} & v_i \in V, CV(v_i) = true \\ e_{CE} & e_i \in E, CE(e_i) = true \end{cases} \tag{9.2}$$

where $g_{CV,CE}$ is the graph subset containing the results of the subset operation, CV and CE are logical operations (i.e. **WHERE** statements in SQL) that return true if the given node or edge, respectively, are to be included in the subset.

The CV and CE operations may place an arbitrary set of constraints on the selection of nodes and edges.

If $CV == null$ but $CE! = null$, a set of edges is extracted into the graph subset, and the vertex set consists of all vertices that are referred to as a source or destination of the selected edges.

If $CE == null$ but $CV! = null$, a set of vertices and all edges that connect them are extracted into the subset.

The graph subset as extracted by this operation is stored in a persistent database view and can be revisited or used as a basis for future pruning operations or queries.

9.7.3 EgoNet: Generating Ego Network subsets

The *EgoNet* operation returns a subset of the graph that consists of nodes that are located at or within a given graph distance. All edges that exist between these nodes are included in the subset, although it is possible to prune the subset further by applying an edge condition.

$$g_{egonet(v_c,d)} = \begin{cases} v_{egonet} & v_i \in V, distance(v_i, v_c) < d \\ e_{egonet} & e_i \in E, source(e_i) \in v_{egonet} \wedge dest(e_i) \in v_{egonet} \end{cases} \tag{9.3}$$

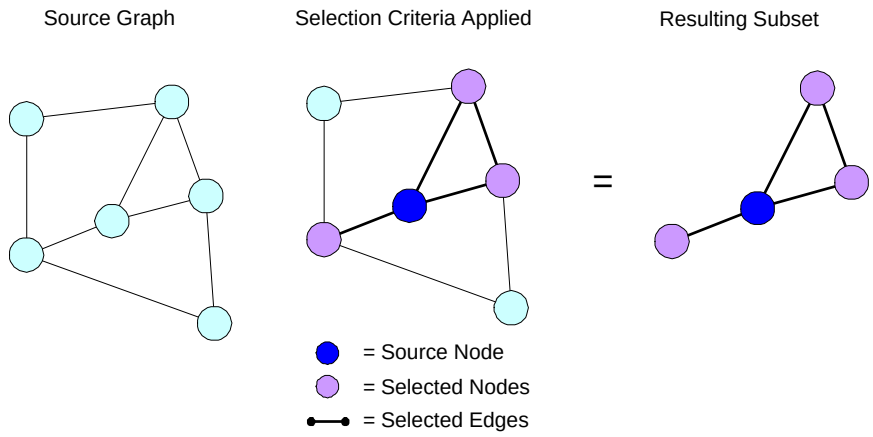


Figure 9.4: Extraction of EgoNet

where v_c is the center node specified by the user, and d is the maximum graph distance between the center node and any other node in the subset.

It is a special case of the subset operations, as it imposes a relational condition upon the selection of nodes and requires computation of graph distance as an intrinsic database operation.

The *EgoNet* function performs a breadth-first search of the graph, starting at the center and expanding its search radius until it is equal to the specified maximum radius d.

Pure SQL is not very suitable graph-theoretic computations as most of them require recursion, which is expressly forbidden in SQL semantics. To implement graph traversals within the database, the recursive component is written in PL-SQL, a procedural language shared between Oracle and a number of other database engines. The PL-SQL function then calls SQL *SubSet* operations and builds up recursive views of the database.

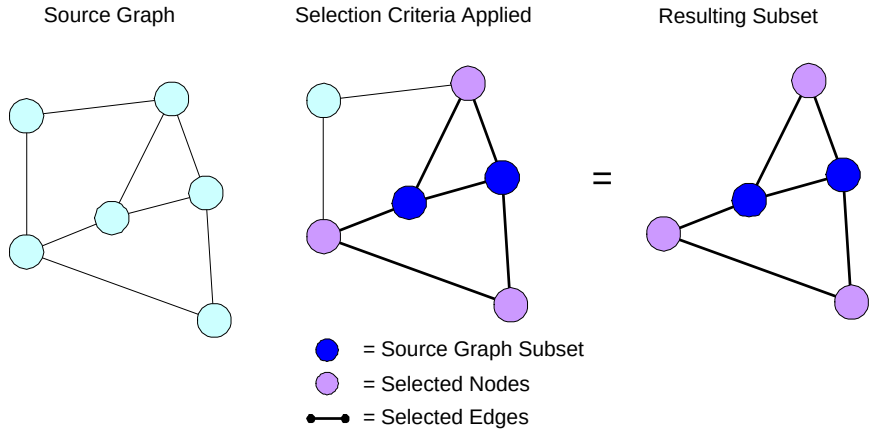


Figure 9.5: Network Expansion

Format:	`egonet(center-node-id,distance)`
Arguments:	
center-node-id ⇒ string	the ID of the center node of the ego network
distance ⇒ int	maximum graph distance between the center node and any other node in the subset
Returns:	A **SELECT** set of objects of type **Node**.
Example Usage	
SQL:	`SELECT * FROM egonet('bin_ladin',2);`
db2dynetml:	`db2dynetml <database> <output file> -ego bin_ladin -distance 2`
WWW:	In the node list, click on "Egonet" button next to the center node

9.7.4 Network Expansion

The purpose of *NetExpand* operation is, starting with an incompletely defined graph subset, find all surrounding network features. For example, *NetExpand* can be used to find all organizations, resources, locations and tasks connected to a given group of people.

The *NetExpand* function takes a graph subset S_source returns a subset of the graph S_result such that all nodes in S_result are at or within a given graph distance from one of the nodes in S_source. In a simpler fashion, it can be defined as a union of ego networks (*EgoNet* of all nodes in the S_source:

$$NetExpand(S_{source}, d) \quad = \quad \bigcup(EgoNet(v_i, d), v_i \quad \in \quad S_{source} \quad (9.4)$$

Format:	`fullnet(source-nodeset,distance)`
Arguments:	
center-node-set ⇒ name	the name of database view of type **Node** that contains the center nodeset. Database view can be created by running the following command: `CREATE VIEW <name> AS SELECT * FROM NODE WHERE ...node conditional...`
distance ⇒ int	maximum graph distance between the center nodes and any other node in the subset
Returns:	A **SELECT** set of objects of type **Node**.
Example Usage	
SQL:	`SELECT * FROM fullnet(source-nodeset,2);`
db2dynetml:	`db2dynetml <database> <output file> -net source.xml -distance 2`
WWW:	Load a database subset on screen by importing a DyNetML file, running queries or manual selection; then click on "Expand Network" in the toolbar

9.7.5 Nodeset Pruning

The *ExcludeCond* function removes nodes from a graph subset given a based on a selection criteria. The function is defined as:

$$g_{ExcludeCond}(source, CV) =$$
$$= \begin{cases} v_{exclude} & v_i \in source, CV(v_i) = false \\ e_{exclude} & e_i \in E, source(e_i) \in v_{exclude} \wedge dest(e_i) \in v_{exclude} \end{cases} \quad (9.5)$$

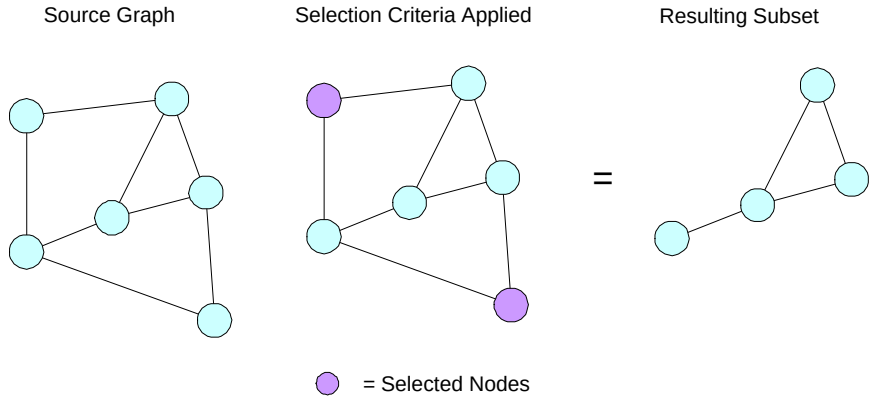


Figure 9.6: Pruning the network subsets

Format:	**exclude-cond(source-nodeset,exclude-conditional)**
Arguments:	
source-nodeset → name	the name of database view of type **Node** that contains the center nodeset.
exclude-conditional → sql	SQL statement containing the exclusion condition
Returns:	A **SELECT** set of objects of type **Node**.
Example Usage	
SQL:	**SELECT * FROM exclude-cond(source-nodeset,"WHERE nodeID!='bin_ladin'");**
db2dynetml:	**db2dynetml database-name output-file -net source.xml -exclude-cond "WHERE nodeID!='bin_ladin'"**
WWW:	Currently no Web interface implemented

The *ExcludeNodeset* function removes nodes from a graph subset given a based on membership in another set. The function is defined as:

$$g_{ExcludeNodeset}(source, exclude-list) = \\ = \begin{cases} v_{exclude} & v_i \in source, v_i \notin exclude-list \\ e_{exclude} & e_i \in E, source(e_i) \in v_{exclude} \wedge dest(e_i) \in v_{exclude} \end{cases} \quad (9.6)$$

Format:	**exclude-nodeset(source-nodeset,exclude-nodeset)**
Arguments:	
source-nodeset → name	the name of database view of type **Node** that contains the center nodeset.
exclude-nodeset → name	the name of a database view containing nodes to exclude from the given nodeset
Returns:	A **SELECT** set of objects of type **Node**.
Example Usage	
SQL:	**SELECT * FROM exclude-cond(source-nodeset,exclude-nodeset);**
db2dynetml:	**db2dynetml database-name output-file -net source.xml -exclude-list exclude.xml**
WWW:	Load the source nodeset from a DyNetML file, click on "Limit by" button in the toolbar and select a DyNetML file containing the exclusion list.

9.8 Auxiliary Tools

Current subsetting tools are written as stored procedures, and can be accessed through any programmatic interface to PosgreSQL, the Web interface or through a command-line programme **db2dynetml**.

The manipulation tools allow easy completion of incomplete datasets (e.g. "Given a set of people, find all organizations and resources connected to them"). To initiate dataset completion, the **db2dynetml** tool is launched with a DyNetML file containing the subject dataset. It then runs a set of graph-traversal expansions on the network and stores their results as a new network.

9.8.1 Exporting data from the database: db2dynetml

db2dynetml is a cross-platform command-line tool that exports information stored in a NetIntel database into a DyNetML file. Versions of the tool exist for Windows NT/2000/XP, Linux and OpenBSD; other operating systems that utilize a standard GNU compiler architecture can be also supported.

Command Usage

```
prompt> db2dynetml database-name output-file
 [-h<db host> ] [-u<db username> ] [-ego<egonet center> ]
 [-distance<egonet distance> ] [-doc<documentID> ]
 [- net<DyNetML file>] [-exclude<Exclusion List>]
```

The command-line parameters are explained below:

`database-name`	name of the database to connect to; **Required**
`-h db_host`	name or IP address of the database host (Optional; default value is the the name of the central CASOS application server)
`-u db_username`	database username (Optional; specify when not using a default host. The program will request a password interactively if the database requires password authentication
`output_file`	name of a DyNetML output file; **Required**
`-doc documentID`	Extract a dataset that is related to a particular source document
`-ego egonet_center`	Generate an ego network centered around a node (see section 9.7.3
`-distance egonet_distance`	Radius of the ego network; requires -ego or -net
`-net DyNetML_file`	expand a network from one specified in the file (see section 9.7.4
.	*Note: -ego and -net are mutually exclusive*
`-exclude Exclusion_List`	prune dataset (resulting from -net, -ego or -doc options) using exclusion list stored in a DyNetML file.

9.8.2 Importing Data into the database

dynetml2db is a tool for importing data stored in DyNetML files into the database. The database will not only store the data contained in the DyNetML file but also track the origin of data by creating an entry in the document

table. Thus, the exact copy of the imported document can be retrieved using the `-doc` switch of the `db2dynetml`.

Command Usage

```
prompt> dynetml2db <database name> <input file>  [-h<db host> ]
 [-u<db username> ] [-m<message> ]
```

The command-line parameters are explained below:

`database-name`	name of the database to connect to; **Required**
`-h db_host`	name or IP address of the database host (Optional; default value is the the name of the central CASOS application server)
`-u db_username`	database username (Optional; specify when not using a default host. The program will request a password interactively if the database requires password authentication
`input_file`	name of a DyNetML file to be imported; **Required**
`-m message`	tag the imported data with a message; messages can be viewed and searched via the WWW interface

9.9 WWW Interface to NetIntel dataset

Figure 9.7 shows the WWW interface for entry, editing and manipulation of data stored in the NetIntel database. The interface allows a user to enter new nodes and edges, search the database for occurrence of keywords and build subsets of data based on attribute values as well as graph-theoretic measures.

The WWW interface allows graphical controls for building complex queries against the database, as well as easy import and export of data.

The WWW interface is written as in PHP and runs on an Apache server.

9.9.1 Managing Graph Data

The main screen (Figure 9.8) of the NetIntel interface consists of a toolbar that contains a number of common actions, and a list of graph nodes or objects in the database.

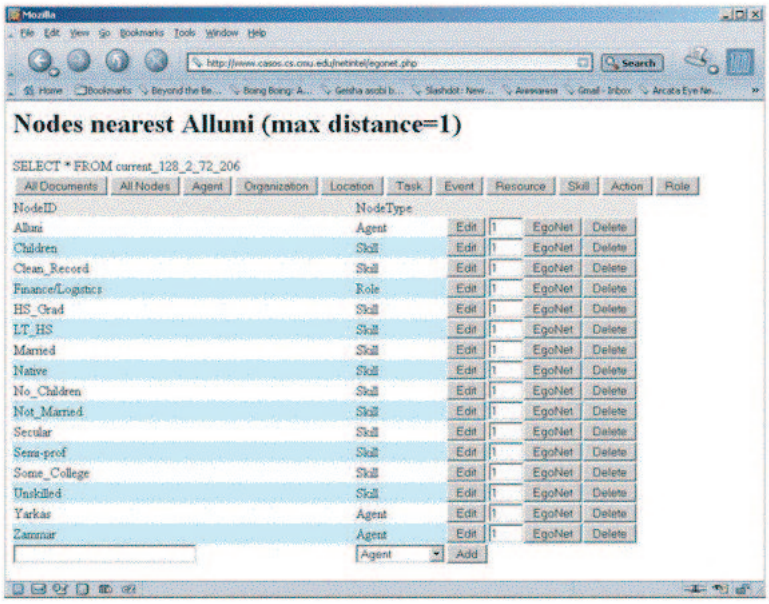


Figure 9.7: Screenshot of the WWW Interface to NetIntel Database

The toolbar of the NetIntel interface allows the user to import and export data, as well as run queries and run graph expansion:

`All Documents`	Displays the list of documents imported into the database (see section 9.9.2)
`All Nodes`	Reset the current subset and display all of the nodes in the database
`Export to DyNetML`	Construct a DyNetML file from the currently selected subset of the network
`Import DyNetML`	Import a DyNetML file and register it as a data source document
`Expand Net`	Run the NetExpand operation (see section 9.7.4)
`Agent`, `Organization`, `Location, etc`	Limit the nodes displayed by node type. A button will be created for every type of nodes present in the database

The node list contains nodes in the currently selected database subset. The current subset can be manipulated using any of the query, network expansion and sorting functions. The current subset can also be exported as a DyNetML file using the `Export to DyNetML` button on the toolbar

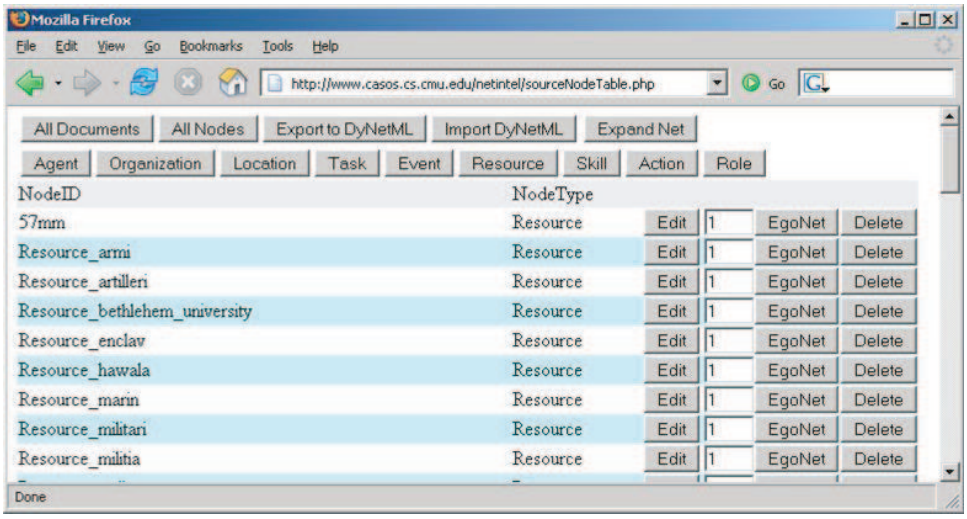


Figure 9.8: Screenshot: Main toolbar and Listing of Nodes

Each of the nodes in the list is displayed with its ID, type and 3 buttons that allow operations on individual nodes:

`Edit`	Brings up the editing screen (see below)
`EgoNet`	Runs the EgoNet function (section 9.7.3) on the selected node and makes EgoNet selection the current subset. The distance parameter of the EgoNet function can be entered to the left of the button
`Delete`	Deletes a node and all edges associated with it.

The node editing screen (figure 9.9) allows the user to edit the attributes of a node and view and edit edges associated with the node. Number and type of attributes are dictated by the type of the node.

The list of edges is displayed below the node attributes. The edges are sorted by the type of target nodes (e.g. links between an agent and organization will be shown in the Organization section). Each of the edges is listed with its label, as well as strength and confidence values. Pressing the `GO` button next to the edge target displays the target node in the editing screen.

This behavior of the interface allows for fast manual data entry from text sources. For example, to code a statement *"Hamas was headed by Yassin and Rantissi"*, the user should add a node of type **Organization** and name it *Hamas*. The new node will be automatically displayed in the editing screen. In the **Agent** category of the edge list, the user should type the name *Yassin* and add the edge. Then, *Rantissi* is added with the similar operation. Both nodes will be automatically added to the database and can be edited by pressing the `GO` button next to them.

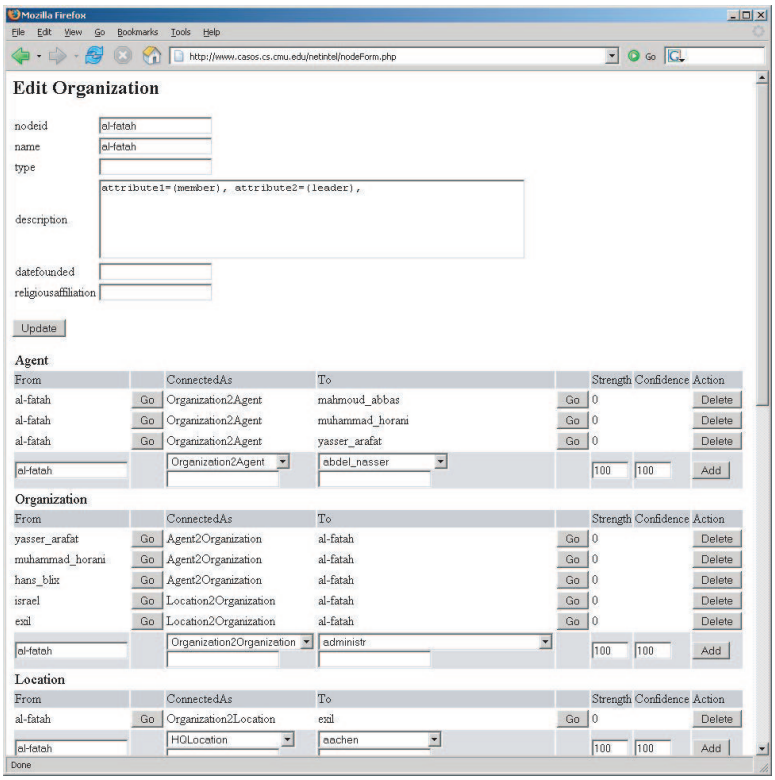


Figure 9.9: Screenshot: Viewing and editing information in an imported document

9.9.2 Managing Data Source Documents

NetIntel interface allows the user to view and manage any number of data source documents after they have been imported into the database. This makes it possible to integrate multiple data sources as well as retain access to each dataset individually. Figures 9.10 and 9.11 show the interface for managing collections of documents and associated network data.

9.10 Conclusion

NetIntel is a flexible database system designed for handling large volumes of graph-based and social network data. NetIntel is built as an extension to a standard SQL database, and thus does not require custom or experimental database storage engines, and can utilize the wealth of third-party interface software and APIs. While not as full-featured as QGraph or other systems designed specifically for graph manipulation, it provides a reliable means of storage and manipulation of graph-based data and easy-to-use facilities for export of data into analysis tools as well as online browsing and data entry.

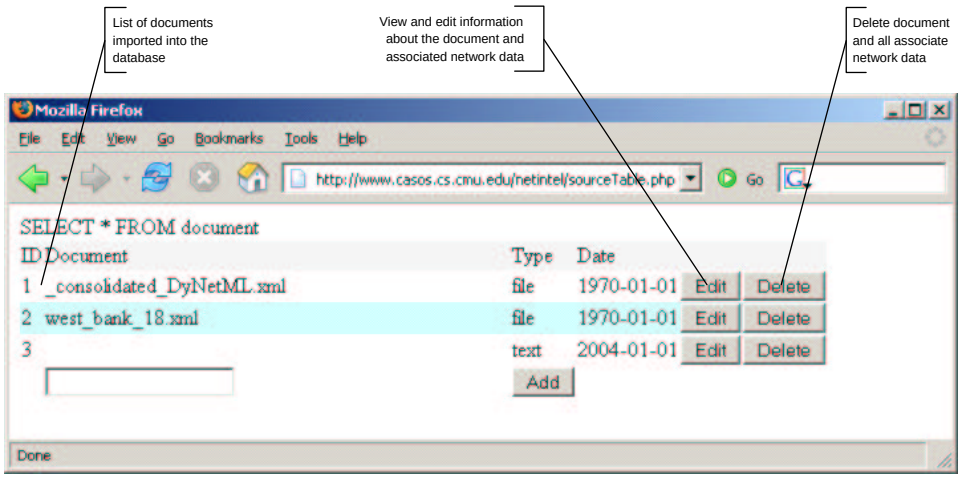


Figure 9.10: Screenshot: List of imported documents

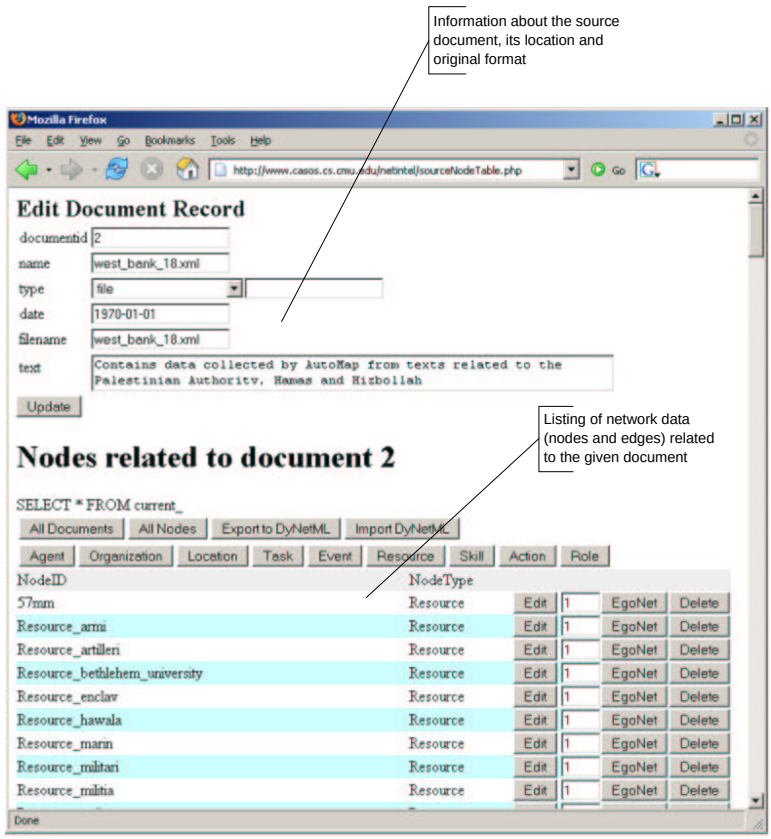


Figure 9.11: Screenshot: Viewing and editing information in an imported document

Bibliography

H. Abelson, G. J. Sussman, and J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press and McGraw-Hill, Cambridge, MA and New York, NY, 1985.

S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.

Al-Quaeda. Al quaeda training manual: Declaration of jihad against unholy tyrants. URL: `http://www.usdoj.gov/ag/trainingmanual.htm`, 2001.

D. Alberts, J. Garstka, and F. Stein. *Network Centric Warfare: Developing and Leveraging Information Superiority*. CCRP Publication Series, 1999.

H. Aldrich. *Organizations Evolving*. Sage Publications, London, 1999.

N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley and Sons, New York, 1992.

T. D. S. C. A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng. Daml-s: Semantic markup forweb services. In *Proccedings of Semantic Web WorkShop (SWWS 2001)*, 2001.

M. Ashworth. Identifying key contributors to performance in organizations: The case for knowledge-based measures. CASOS Working Paper, 2003.

M. Ashworth and K. M. Carley. Identifying critical human capital in organizations. CASOS Working Paper, 2002.

M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonik. The object-oriented database system manifesto. In *Proceedings of the First International Conference on Deductive and Object-Oriented Databases*, pages 223–240, Kyoto, Japan, 1989. URL `citeseer.ist.psu.edu/atkinson89objectoriented.html`.

W. A.Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):., 1970.

R. Axtell and J. Epstein. Agent based modeling: Understanding our creations. *The Bulletin of the Santa Fe Institute*, pages 28–32, Winter 1994.

H. Baligh, R. Burton, and B. Obel. *Devising expert systems in organization theory: The organizational consultant.*, volume Michael Masuch (Ed.) of *Organization, management, and expert systems*, pages 35–57. Walter De Gruyter, Berlin, Germany, 1990.

A. Barabasi. *Linked: The New Science of Networks*. Perseus Publishing, Cambridge, 2002.

A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, Oct 1999.

V. Batagelj and A. Mrvar. *Pajek - analysis and visualization of large networks*, pages 77–103. Springer, Berlin, 2003.

N. Berry. The international islamic terrorist network. *CDI Terrorism Project*, September 2001.

P. Biernacki and D. Waldorf. Snowball samplong. problems and techniques of chain referral sampling. *Sociological Methods Research*, 10(2):141–163, 1981.

H. Blau, N. Immerman, and D.Jensen. A visual language for querying and updating graphs. Technical Report TR 2002-037, University of Massachusetts Amherst, Computer Science, 2002. URL `citeseer.ist.psu.edu/686727.html`.

S. Borgatti, K. Carley, and D. Krackhardt. On the robustness of centrality measures under conditions of imperfect data. http://www.casos.cs.cmu.edu/publications/papers/CentralityRobustness5b.pdf, 2004.

U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. S. Marshall. Graphml progress report: Structural layer proposal. In *Proc. 9th Intl. Symp. Graph Drawing (GD '01) LNCS 2265*, pages 501–512. Springer-Verlag, 2002.

U. Brandes, J. Lerner, and C. Pich. Gxl to graphml and vice versa with xslt. In *International Workshop on Graph-Based Tools (GraBaTs0)*, 2004.

R. Breiger. *Handbook of Data Analysis (M. Hardy and A. Bryman, eds)*, chapter The Analysis of Social Networks, page 505526. Sage Publications, London, 2004.

R. L. Breiger and J. W. Mohr. Institutional logics from the aggregation of organizational networks: Operational procedures for the analysis of counted data. *Computational and Mathematical Organization Theory*, 10(1):17–43, May 2004.

R. Brooks. A layered intelligent control system for a mobile robot. In *Proceedings of the Third International Symposium of Robotics Research*, page 8. MIT Press, october 1985.

R. M. Burton and B. Obel. *Strategic Organizational Diagnosis and Design: Developing Theory for Application*. Dordrecht: Kluwer Academic Publishers, 1998.

L. Cardelli. A semantics of multiple inheritance. *Information and Computation*, 76:138–164, 1988.

K. Carley. Organizational learning and personnel turnover. *Organizational Science*, 3(1):2–46, 1992a.

K. Carley. Communication technologies and their effect on cultural homogeneity, consensus, and the diffusion of new ideas. *Sociological Perspectives*, 38(4):547–557, 1995.

K. Carley, J. Lee, and D. Krackhardt. Destabilizing networks. *Connections*, 24(3):79–92, 2002.

K. M. Carley. Organizational learning and personnel turnover. *Organization Science*, 3(1):20–46, 1992b.

K. M. Carley. Extracting culture through textual analysis. *Poetics*, 22:291, 1994.

K. M. Carley. On the evolution of social and organizational networks. *Research in the Sociology of Organizations*, 16(Special issue on Networks In and Around Organizations):3–30, 1999.

K. M. Carley. Inhibiting adaptation. *Proceedings of Command and Control Research and Technology Symposium*, 2002a.

K. M. Carley. Simulating society: The tension between transparency and veridicality. In *Proceedings of Agents 2002*, Chicago, IL, 2002b.

K. M. Carley. Smart agents and organizations of the future. In L. Lievrouw and S. Livingstone, editors, *The Handbook of New Media*, chapter 12, pages 206–220. Sage, Thousand Oaks, CA, 2002c.

K. M. Carley. *Dynamic Network Analysis*, pages 133–145. Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers. Comittee on Human Factors and National Research Council, ronald breiger and kathleen m. carley and philippa pattison, (eds.) edition, 2003a.

K. M. Carley. Dynamic network analysis. In K. C. R. Breiger and P. Pattison, editors, *Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers*, pages 361–370. Committee on Human Factors, National Research Council, 2003b.

K. M. Carley, M. Dombroski, M. Tsvetovat, J. Reminga, and N. Kamneva. Destabilizing dynamic covert networks. *Proceedings of the 8th International Command and Control Research and Technology Symposium*, 2003.

K. M. Carley and V. Hill. Structural change and learning within organizations. *In Dynamics of organizational societies : Models theories and methods. Edited by Alessandro Lomi*, forthcoming.

K. M. Carley and J. Reminga. Ora: Organization risk analyzer. Technical Report Technical Report CMU-ISRI-04-101, Carnegie Mellon University, School of Computer Science, Institute for Software Research International, 2004.

K. M. Carley and Y. Ren. Tradeoffs between performance and adaptability for c3i architectures. *Proceedings of the 2000 International Symposium on Command and Control Research and Technology*, 2001.

K. M. Carley and D. M. Svoboda. Modeling organizational adaptation as a simulated annealing process. *Sociological Methods and Research*, 25(1): 138–168, 1996.

J. F. Castro, J. Mylopoulos, F. M. R. Alencar, and G. A. C. Filho. Integrating organizational requirements and object oriented modeling. In *Proceedings of Fifth IEEE International Symposium on Requirements Engineering (RE '01)*, page 146, Toronto, Canada, August 2001.

D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-mat: A recursive model for graph mining. In *SIAM International Conference on Data Mining*, 2004.

J. Collins, M. Tsvetovat, B. Mobasher, and M. Gini. Magnet: A multi-agent contracting system for plan execution. In *Proceedings of SIGMAN 98*, 1998.

W. W. W. Consorcium. Xml schema specification. http://www.w3.org/XML/Schema, 2004.

N. S. Contractor and P. R. Monge. Using multi-theoretical multi-level (mtml) models to study adversarial networks. In R. Breiger and K. M. Carley, editors, *Summary of the NRC workshop on Social Network Modeling and Analysis*, National Research Council. forthcoming.

I. Corporation. Analyst's notebook: Powering your analysis, 2005. http://www.i2inc.com/Products/Analysts_Notebook/default.asp.

R. Corporation. Purpose and description of information found in the incident databases, 2003. http://www.tkb.org/RandSummary.jsp.

E. Costenbader and T. Valente. The stability of centrality measures when networks are sampled. *Social Networks*, 25:283–307, 2003.

K. M. C. Craig Schreiber. Key personnel: Identification and assessment of turnover risk. In *Proceedings of NAACSOS 2004*, Pittsburgh, PA, 2004.

R. Cross, S. P. Borgatti, and A. Parker. Beyond answers: dimensions of the advice network. *Social Networks*, 23:215235, 2001.

O.-J. Dahl and K. Nygaard. Simula — an algol-based simulation language. *Communications of the ACM, D.E. Knuth, ed.*, 9(9):671–678, September 1966.

N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *KDD'04*, Seattle, Washington, August 2004.

C. Deaton, B. Shepard, C. Klein, C. Mayans, B. Summers, A. Brusseau, M. Witbrock, and D. Lenat. The comprehensive terrorism knowledge base in cyc. In *Proceedings of 2005 MITRE Conference on Intelligence Analysis*, 2005.

J. Diesner and K. Carley. Automap1.2 - extract, analyze, represent, and compare mental models from texts. Technical Report CMU-ISRI-04-100, Carnegie Mellon University, School of Computer Science, Institute for Software Research International, 2004.

J. Diesner and K. Carley. *Revealing Social Structure from Texts: Meta-Matrix Text Analysis as a novel method for Network Text Analysis.*, chapter 4. Causal Mapping for Information Systems and Technology Research: Approaches, Advances, and Illustrations, V.K Naraynan, D.J. Armstrong (Eds.). Idea Group Publishing, Harrisburg, PA, 2005.

J. Diesner, E.T.Lewis, and K.M.Carley. Using automated text analysis to study self-presentation strategies. 2001.

E. W. Dijkstra. Letters to the editor: go to statement considered harmful. *Communications of the ACM archive*, 11(3):147–148, 1968.

P. DiMaggio. Culture and cognition. *Annual Review of Sociology*, 23:363, 1998.

E. Elliott and L. Kiel. A complex systems approach for developing public policy toward terrorism: an agent-based approach. In *Chaos, Solitons and Fractal, Proceedings of the October 2002 Denton Workshop*, 2002.

Erdös and Rényi. On the evolution of random graphs. *Publication of Mathematics Institute of Hungian Academy of Sciences*, 5:1761, 1960.

Erdös and Rényi. On the strength of connectedness of random graphs. *Acta Math. Acad. Sci. Hungar*, 12:261–267, 1961.

D. B. Eric Miller, Ralph Swick. Rdf: Resource description framework. http://www.w3.org/RDF/, 2004.

B. H. Erickson. Secret societies and social structure. *Social Forces*, 60(1): 188–210, 1981.

K. Erol, J. Hendler, and D. Nau. Semantics for hierarchical task network planning, 1994. URL `citeseer.ist.psu.edu/article/erol95semantics.html`.

T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In N. Adam, B. Bhargava, and Y. Yesha, editors, *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA, 1994. ACM Press. URL `citeseer.ist.psu.edu/finin94kqml.html`.

F.Lorrain and H. White. Structural equivalence of individuals in social networks. *Journal of Mathematical Sociology*, 1, 1971.

F. S. Foundation. Guile: Project gnu's extension language, 2005.

W. B. Frakes and P. B. Gandel. Representation methods for software reuse. In *Proceedings of the conference on Tri-Ada89*, 1989.

L. C. Freeman. Centrality in social networks: Conceptual clarication. *Social Networks*, 1:215–239, 1979.

N. Friedkin. *A Structural Theory of Social Influence*. Cambridge University Press, New York, 1998.

R. Goolsby. Combating terrorist networks: An evolutionary approach. In *Proceedings of the 8th International Command and Control Research and Technology Symposium*. Conference held at National Defence War College Washington DC, Evidence Based Research Vienna VA, 2003.

M. Granovetter. Network sampling: Some first steps. *American Journal of Sociology*, 81:1267–1303, 1976.

I. C. Group. Indonesia backgrounder: How the jemaah islamiyaa terrorist network operates. *Asia Paper*, (43), December 2002.

T. Gruber. Toward the deign of ontologies used for knowledge sharing. *Knowledge Acquisition*, 5(2):199–220, 1993.

L. Gruenwald, G. McNutt, and A. Mercier. Using an ontology to improve search in a terrorism database system. *Proceedings of the 14th International Workshop on Database and Expert System Applications (DEXA'03)*, 2003.

R. Gunaratna. *Inside Al Quaeda: Global Network of Terror*. New York University Press, New York, NY, 2002.

S. Hart and L. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Human mental workload*, pages 139–183. P.A. Hancock and N. Meshkati (Eds.), Amsterdam: Elsevier, 1988.

I. Herman and M. S. Marshall. Graphxml - an XML-based graph description format. In *Graph Drawing*, pages 52–62, 2000. URL `citeseer.ist.psu.edu/article/herman00graphxml.html`.

R. C. Holt, A. Winter, and A. Schürr. Gxl: Towards a standard exchange format. Technical Report 1–2000, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz, 2000. URL `citeseer.ist.psu.edu/holt00gxl.html`.

B. Houghton. Understanding the terrorism database. National Memorial Institute for Prevention of Terrorism Quarterly Bulletin, 2002.

E. V. J. Ian Davis. Relationship: A vocabulary for describing relationships between people, 2004. http://purl.org/vocab/relationship.

IntelCenter. Intelcenter database (icd), 2005. http://www.intelcenter.com/icd/index.html.

IntelCenter.com. Mapping al-qaeda v1.0. www.intelcenter.com, 2003.

S. A. Jennifer Widom. Lore: a database management system (dbms) for xml, 2004. http://www-db.stanford.edu/lore/.

D. Jensen, M. Rattigan, and H. Blau. Information awareness: a prospective techincal assessment. In *In proceedings of the Ninth ACM SIGKDD Internation Conference on Knowledge Discovery and Data Mining*, pages 378–387, Washington, DC, 2001. ACM Press.

D. Karnopp, D. Margolis, and R. Rosenberg. *System Dynamics: A Unied Approach*. John Wiley, 1990.

D. Krackhardt. Assessing the political landscape: Structure, cognition, and power in organizations. *Administrative Science Quarterly*, (35):342–369, 1990.

D. Krackhardt. The ties that torture: Simmelian tie analysis in organizations. *Research in the Sociology of Organizations*, 16:183–210, 1999.

D. Krackhardt. David krackhardt's sample network questionnaires, 2003. http://www.andrew.cmu.edu/user/krack/academic/questionnaires.html.

D. Krackhardt and K. M. Carley. A pcans model of structure in organizations. *Proceedings of the 1998 International Symposium on Command and Control Research and Technology*, pages 113–119, june 1998.

D. Krackhardt and J. Hanson. Informal networks: the company behind the chart. *Harvard Business Review*, 71(4):104–11, Jul-Aug 1993.

P. L. Krapivsky, S. Redner, and F. Leyvraz. Connectivity of growing random networks. *Physics Review Letters*, 85:4629–4632, 2000.

V. E. Krebs. Mapping networks of terrorist cells. *Connections*, 24(3):43–52, 2001.

J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

C. Langton. *Artificial Life*, pages 1–47. SFI Studies in the Sciences of COmplexity. Addison-Wesley, Redwood City, CA, 1989.

P. Lawrence and J. Lorsch. Differentiation and integration in complex organizations. *Administrative Science Quarterly*, (12):1–47, 1967.

D. B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995. URL `citeseer.ist.psu.edu/lenat95cyc.html`.

R. Levitt, G. Cohen, J. Kunz, C. Nass, T. Christiansen, and Y. Jin. *The "Virtual Design Team": Simulating How Organization Structure and Information Processing Tools Affect Team Performance*, chapter 1, pages 1–18. Computational Organization Theory, K.M. Carley and M.J. Prietula, eds. Lawrence Erlbaum, 1994.

Z. Lin and K. Carley. Organizational response: The cost performance tradeoff. *Management Science*, 43(2):217–234, 1997.

Z. Lin and K. M. Carley. *Designing Stress Resistant Organizations: Computational Theorizing and Crisis Applications*. Kluwer, Boston, MA, 2003.

L.P.Gerlach and V.H.Hine. *People, Power, Change: Movements of Social Transformation*. Bobbs-Merrill, Indianapolis, IN, 1970.

B. Machta and J. Machta. Parallel dynamics and computational complexity of network growth models. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 71(2):026704, 2005. URL `http://link.aps.org/abstract/PRE/v71/e026704`.

J. March. *Decisions and Organizations*. Basil Blackwell, Oxford, 1988.

J. March and H. Simon. *Organizations*. Wiley, New York, 1958.

M. Mateski. Red teaming terrorism (method paper 1.03). *Red Team Journal*, June 2003.

D. McKay, T. Finin, and A. O'Hare. The intelligent database interface: Integrating AI and database systems. In T. Dietterich and W. Swartout, editors, *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, pages 677–684, Boston, MA, USA, 29– 3 1990. AAAI Press. URL `citeseer.ist.psu.edu/mckay90intelligent.html`.

J. Melton. Sql language summary. *ACM Computing Surveys*, 28(1), March 1996.

M. L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, N. J., 1967.

J. W. Mohr. Measuring meaning structures. *Annual Review of Sociology*, 24: 345–370, 1998.

J. Neville and D. Jensen". Supporting relational knowledge discovery: Lessons in architecture and algorithm design. *Papers of the ICML 2002 Workshop on Data Mining Lessons Learned*, 2002. URL `citeseer.ist.psu.edu/654471.html`.

L. Numerical Algorithms Group. Iris explorer reference. URL: `www.nag.co.uk`, 2004.

ObjectStyle. Ashwood java library. http://www.objectstyle.org/ashwood, 2005.

R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physics Review Letters*, 86(14), april 2001.

PostgreSQL. www.postgresql.org, 2005. www.postgresql.org.

S. Reed and D. Lenat. Mapping ontologies into cyc. In *AAAI 2002 Conference Workshop on Ontologies For The Semantic Web*, Edmonton, Canada, July 2002.

R. S. Renfro. *Modelling and Analysis of Social Networks*. PhD thesis, Department of Air Force, Air Force Institute of Technology, 2003.

L. Ritter. *Tools and methods for embedded system design using Ada*. 1989.

J. Robb. Scale-free terrorist networks. Jef Allbrights Web Files; URL: `www.jefallbright.net/node/view/2632`, 2004.

D. Ronfeldt and J. Arquilla. Networks, netwars and the fight for the future. *First Monday*, 6(10), 2001.

R. Rothenberg. From whole cloth: Making up the terrorist netwrok. *Connections*, 24(3):36–42, 2002.

C. Ruby. The definition of terrorism. *Analyses of Social Issues and Public Policy*, pages 9–14, 2002.

G. Sabidussi. The centrality index of a graph. *Psychomatrika*, 31:581–603, 1966.

M. Sageman. *Understanding Terror Networks*. University of Pennsylvania Press, 2004.

A. Sanfeliu and K. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:353–362, 1983.

C. Schaffert, T. Cooper, B. Bullis, M. Killian, and C. Wilpolt. An introduction to trellis/owl. In *Proceedings of OOPSLA 1986*, pages 9–16, 1986.

C. Schreiber and K. M. Carley. Construct - a multi-agent network model for the co-evolution of agents and socio-cultural environments. Technical Report CMU-ISRI-04-109, Carnegie Mellon University, School of Computer Science, Institute for Software Research International, Pittsburgh, PA, 2004.

S. C. Shapiro. Generalized augmented transition network grapmmars for generation from semantic networks. *American Journal of Computational Linguistics*, 8(1):12–25, 1982.

H. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–118, 1955.

B. L. Smith and K. R. Damphousse. The american terrorism study: Indictment database, 2002.

M. E. S.P. Borgatti and L. Freeman. Ucinet for windows. 2002.

E. Stacy. An overview of organization description language. Technical Report Technical Report AP-R-1154, Aptima, 2004.

C. Stohl and M. Stohl. The nexus and the organization: The communicative foundations of terrorist organizing. In *Communication in Action: The Communicative Constitution of Organization and its Implication for Teopry and Practice*, New Orleans, 2002.

G. Taentzer. Towards common exchange formats for graphs and graph transformation systems. *Electronic Notes in Theoretical Computer Science*, 44 (4), 2001. URL `citeseer.ist.psu.edu/taentzer01towards.html`.

A. Tate. Generating project networks. In *Proceedings of IJCAI-77*, pages 888–893, 1977.

J. Thomson. *Organizations in Action*. McGraw-Hill, New York, 1967.

M. Tsvetovat and K. Carley. Bouncing back: Recovery mechanisms of covert networks. 2003.

M. Tsvetovat and K. Carley. Modeling complex socio-technical systems using multi-agent simulation methods. *Kunstliche Intelligenz (Artificial Intelligence Journal)*, Special Issue on Applications of Intelligent Agents(2), 2004.

M. Tsvetovat and K. M. Carley. Structural knowledge and success of anti-terrorist activity: The downside of structural equivalence. *Journal of Social Structure (www.joss.org)*, forthcoming, 2005.

M. Tsvetovat, J. Diesner, and K. Carley. Netintel: A database for manipulation of rich social network data. Technical Report Technical Report CMU-ISRI-04-135, Carnegie Mellon University, School of Computer Science, Institute for Software Research International, 2005.

S. Tzu. *The Art of War (translation 1963, S.Griffith(translator))*. Oxford University Press, Oxford, 6th cent. B.C.

P. Wadler. A formal semantics of patterns in xslt, 1999. URL `citeseer.ist.psu.edu/wadler00formal.html`.

G. Wagner. Agent-oriented analysis and design of enterprise information systems. In *International Conference on Enterprise Information Systems*, pages 85–89, 2000. URL `citeseer.ist.psu.edu/wagner00agentoriented.html`.

D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, (393):440–442, 1998.

D. M. Wegner. Transactive memory: A contemporary analysis of the group mind. *Theories of group behavior, edited by B. Mullen and G. R. Goethals*, pages 185–208, 1987.

H. White, S. Boorman, and R. Breiger. Social structure from multiple networks: I. blockmodels of roles and positions. *American Journal of Sociology*, 81(4):730–80, 1976.

J. K. Wimisberg. Hamas - harakat al-muqawamah al-islamiyya (the islamic resistance movement). 2004.

A. Winter. Exchanging graphs with gxl. Technical Report 9–2001, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz, 2001. URL `citeseer.ist.psu.edu/article/winter01exchanging.html`.

W. A. Woods. Cascaded atn grammars. *American Journal of Computational Linguistics*, 6(1):1–12, 1980.

M. Wooldridge. Myworld: The logic of an agent-oriented dai testbed. URL `citeseer.ist.psu.edu/66749.html`.

XML.org. http://www.xml.org/, a.

XML.org. Document object model (dom). http://www.xml.org/xml/resources_focus_dom.shtml, b.

XML.org. Simple api for xml (sax). http://www.xml.org/xml/resources_focus_sax.shtml, c.

A. Yahja. Wizer: What-if analyzer for automated social model space exploration and validation. In *Proceedings of NAACSOS Conference*, 2003.

M. Ye and K. Carley. Radar-soar: Towards an artificial orgnization composed of intelligent agents. 20(2-3):219–246, 1995.

R. M. Young, M. E. Pollack, and J. D. Moore. Decomposition and causality in partial-order planning. In *Proceedings of Second International Conference on Artificial Intelligence and Planning Systems*, pages 189–193, Chicago, IL, 1994.

yWorks. yfiles: Java graph layout and visualization library. http://www.yworks.com/.