Departamento de Engenharia Informática do
Instituto Superior de Engenharia do Porto

Semantic Social Network Analysis

Nuno Miguel Almeida Luz

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática

Área de Especialização em **Arquitectura, Sistemas e Redes**

Orientadores
Prof. Doutora Ana Maria Neves de Almeida Baptista Figueiredo
Prof. Doutor Nuno Alexandre Pinto da Silva

Júri
Presidente: Prof. Doutora Maria de Fátima Coutinho Rodrigues, Professora Coordenadora no Departamento de Engenharia Informática do Instituto Superior de Engenharia do Porto
Vogais: Prof. Doutor Alípio Mário Guedes Jorge, Professor Associado no Departamento de Ciência de Computadores da Faculdade de Ciências da Universidade do Porto
Prof. Doutora Ana Maria Neves de Almeida Baptista Figueiredo, Professora Coordenadora no Departamento de Engenharia Informática do Instituto Superior de Engenharia do Porto
Prof. Doutor Nuno Alexandre Pinto da Silva, Professor Coordenador no Departamento de Engenharia Informática do Instituto Superior de Engenharia do Porto

Porto, Setembro de 2010

Acknowledgements

I would like to thank all the people that in some way gave me their support and contribution during my entire work at GECAD and ISEP. Although I think I am a bit forgetful when trying to write it down, my appreciation and respect for the people who have helped and provided me with their knowledge and wisdom over the years doesn't fade.

First, I would like to thank both my advisors for all their support and advice. To Ana Almeida, especially for her support during the elaboration of this work, flexibility and given freedom of working the way I thought more productive and comfortable. And to Nuno Silva, especially for all his effort and time spent advising and constructively criticizing my work.

Also, I would like to thank GECAD for the opportunity of working in the Toursplan project (PTDC/EIA/74310/2006) and developing my MSc thesis in its context; and the Coalesce project (PTDC/EIA/74417/2006) for providing significant knowledge, necessary to the completion of this thesis.

A special thanks to my girlfriend, Soraia Ferreira, for pulling up my sanity during the most critical moments and for taking on the cumbersome task of reviewing this thesis; to my father Belmiro Almeida, my mother Glória Almeida, and my brother Carlos Luz, for their unconditional support.

Last but not least, I would like to thank all my friends for all the good moments away from work, especially to Ricardo Anacleto for all the years of friendship during my entire academic studies.

Resumo Alargado

Nos últimos anos as redes sociais online têm sido progressivamente adoptadas pela sociedade, influenciando a maneira como as pessoas interagem, partilham e distribuem ideias. A sua crescente popularidade levou ao aparecimento de múltiplas bases de dados com enormes quantidades de dados relativos a interacções sociais. Apesar de já existir uma vasta quantidade de métodos e algoritmos para análise de redes sociais propostos e escrutinados pela comunidade científica, a maior parte não se encontra preparada para a sua aplicação directa sobre a informação, rica e heterogénea, contida nas bases de dados das redes sociais online.

Com o aparecimento da Web Semântica, os dados sociais até agora enclausurados e protegidos por cada uma das entidades responsáveis pelos mesmos, estão a convergir para formar uma enorme massa de informação distribuida e semanticamente enriquecida, partilhada por todas as aplicações com funcionalidades sociais. Num ambiente como este, novos métodos e ferramentas são necessários para que exista uma ponte entre as novas tecnologias emergentes devido à Web Social e Semântica, e os já bem aceites e estabelecidos métodos e algoritmos para análise de redes sociais.

Sendo assim, o principal objectivo desta tese como parte do projecto Toursplan, é não só a adaptação da actual plataforma Toursplan de forma a que esta se possa encaixar na Web Social e Semântica, mas também o desenvolvimento de mecanismos que permitam analisar a semanticamente enriquecida base de dados Toursplan. Do trabalho desenvolvido resultaram:

- A análise do estado da arte das actuais redes sociais online, e de métodos e algoritmos de análise de redes sociais;
- A ontologia Toursplan, que descreve o domínio de conhecimento da plataforma Toursplan, incluindo o perfil dos utilizadores, pontos de interesse turísticos e planeamento de viagens;
- A ontologia SocioNet, que descreve o domínio de conhecimento relativo a métodos e algoritmos de análise de redes sociais, proporcionando um modelo para a persistência de dados resultantes da execução de múltiplos algoritmos previamente analisados e descritos;

- A implementação de uma base de triplos para a plataforma Toursplan, e a migração de toda a informação encontrada na base de dados relacional para a base de triplos;
- A implementação de uma camada de acesso a dados, com base na framework Jena para a Web Semântica, que permite o acesso à base de triplos através das ontologias Toursplan e SocioNet;
- A implementação de um protótipo que corresponde à nova aplicação Web da rede social Toursplan;
- A especificação e implementação de uma fase de normalização de dados provenientes de bases de triplos (ou grafos de triplos) com base em múltiplos padrões de modelação, usando a camada de acesso a dados anterior e a ontologia SocioNet;
- A implementação de alguns dos algoritmos de análise de redes sociais previamente analisados com base na camada SocioNet, a sua execução sobre informação normalizada, e a análise dos resultados obtidos.

Palavras Chave: Web Social, Web Semântica, Turismo, Análise de Redes Sociais

Abstract

Over the last few years, online social networks have become part of society, affecting the way people interact, and share and spread ideas. Their large and still increasing popularity led to the emergence of multiple huge social datasets. Although a vast set of social network analysis methods and algorithms have already been proposed and scrutinized by the research community, most of them aren't prepared for a direct application over the rich and heterogeneous information contained in the online social network datasets.

With the emergence of the Semantic Web, the up until now closed datasets are evolving to become one semantically enriched and distributed social dataset shared by all online social network applications. In such an environment, new methods and tools that fill the gap between the new social and semantic web technologies, and well established and accepted social network analysis methods are required.

In that sense, the main goal of this thesis work, as part of the Toursplan project, is to adapt the current Toursplan platform so it fits into the Social and Semantic Web environments, while also providing mechanisms to perform social network analysis over the Toursplan semantically enriched social dataset.

Keywords: Social Web, Semantic Web, Tourism, Social Network Analysis

Index

Figure Index

Table Index

Notation

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
BL	Business Layer
DAL	Data Access Layer
DL	Description Logics
FOAF	Friend Of A Friend
OWL	Web Ontology Language
POI	Point Of Interest
RDF	Resource Description Framework
RDF-S	RDF Schema
RIF	Rule Interchange Format
SNA	Social Network Analysis
SNO	SocioNet Ontology
SPARQL	Simple Protocol and RDF Query Language
SWRL	Semantic Web Rule Language
TPO	Toursplan Ontology
URI	Uniform Resource Identifier

WAL Web Application Layer

Chapter 1
Introduction

Although being subject of great research efforts over the last few decades, human society and its intrinsic aspects still form a complex puzzle (Levine & Kurzban 2006). As suggested in (Levine & Kurzban 2006), humans have developed sophisticated mechanisms to extract benefits from the social world. These mechanisms are expected to be the product of several adaptations designed to generate mutual benefits to the actors involved.

Humans not only are innately social (Porter 2008), but they also tend to form clustered structures of relationships like social circles or communities. Individuals inside these clusters have great impact on each others' lives, influencing ideas and behaviors. If we exclude mass communication media like the newspapers, radio, television or the internet, the information people has access to, is mostly the information flowing inside their social circles. Ultimately, our social circle has become more like a filter from the vast amounts of information and choices delivered to us every day.

The Web opened new doors regarding studies over human cooperation and behavior, since a lot of the information that flows through the Internet contains or represents, either explicitly or implicitly, social interactions. These interactions manifest themselves through technologies like e-mail, mailing list archives, hyperlink structure of homepages, co-authorship of documents, chat sessions and many others (Erétéo et al. 2009).

The Social Web, also known as Web 2.0, introduced social features in most Web applications and tools, giving birth to multiple explicit social networks for functional or entertainment purposes (Yun & Kim 2009). These applications connect people in a global scale, and allow them to reap the benefits of social life from online virtual environments. When widely used, online social networks garner significant amounts of important social data, which can be used to enhance social benefits.

One of the most straightforward benefits we extract from society comes from asking our friends for an opinion or advice. It is possible to apply a similar mechanism in online social networks by automatically filtering data, and providing the user with relevant and personalized results according to the opinions coming from his online social circle or community. The difference is that, unlike humans, machines can perform that procedure for millions of items, and for a wide and richer social circle.

Online social networks seem to be triggering yet another adaptation to the way humans cooperate, bringing forth new tools for interaction that help to either maintain existent social relationships or establish new ones. They also leverage the emergence of collective intelligence (Segaran 2007) from the combined behavior, ideas and preferences of thousands or even millions of people (Porter 2008), attracting even more participants on a world-wide scale.

An interesting example of the importance of collective intelligence is what Porter regards as the Amazon Effect (Porter 2008). To explain the Amazon Effect, he describes a usability study where people were asked to buy a product at a certain online store. A lot of people wanted to go to Amazon first, and when they were asked why, they just answered that they would like to do some research on the product, even if they were not buying it on Amazon.

But with the amounts of social data growing day by day, the emergence of problems related to handling, linking, analyzing and presenting data are inevitable. These vast sources of information are usually oblivious of each others' existence, thus containing redundant, and possibly inconsistent, data which is collected either implicitly or explicitly. For explicitly collected data, the social individual must submit and update these data multiple times in different data sources, a task that can easily become cumbersome. The Semantic Web, also referred to as Web 3.0, tackles this problem by promoting the creation of semantically-rich data models (e.g. ontologies) and linking of machine readable resource information distributed over the whole World Wide Web.

In the context of the Toursplan (Tours Planning Support System) project (PTDC/EIA/74310/2006), this work aims to enhance the current Toursplan platform through its integration in a Social and Semantic Web environment. Such an environment demands the existence of several components such as:

- A triple store (data must be in the form of triples). As a consequence, the current relational database data must be integrated;
- Social Network Analysis (SNA) methods and algorithms, so relevant information about the collected social interaction data can be extracted;
- A semantically-rich ontology model describing the Toursplan online social network domain of knowledge;
- A semantically-rich ontology model describing the SNA domain of knowledge, so the results of each SNA algorithm execution can be stored and analyzed in the future;
- A normalization process, which allows the execution of the provided SNA algorithms over any triple data source containing social interaction data (e.g. the Toursplan triple store).
- A front-office Web application with social interaction features;

- A back-office Web application with SNA features relying on several SNA algorithms and measures.

In order to successfully create and include all the previously mentioned components in Toursplan, two state of the art analyses are presented: one focuses in online social networks and the Semantic Web, while the other focuses in SNA.

1.1. Context and Organization

This research work was developed in the context of the Toursplan project (PTDC/EIA/74310/2006), developed in GECAD (Knowledge Engineering and Decision Support Group). Toursplan is a decision support system for the tourism domain. It provides tourists with personalized planning tools through adaptive recommendation strategies, relying on context and user profile information (Coelho et al. 2009). Currently, Toursplan features recommendation and planning modules feeding on user profile information collected both implicitly (through user interaction) and explicitly. Toursplan intends to interact with the tourist through a Web application and a mobile device, providing both planning support and real-time assistance.

GECAD is a world-wide known research unit settled in the School of Engineering – Polytechnic of Porto (ISEP/IPP) with the mission of promoting and developing scientific research in the knowledge and decision support domains. The group slogan is “Intelligence for a Sustainable, Safe, and Inclusive World”.

1.2. Thesis Overview

This thesis is organized as follows:

Chapter 1 presents the main objectives and motivations behind this research work. This chapter also gives a brief description of this works development context and organization.

Chapter 2 introduces the Social and Semantic Web, providing insights on the need for Social Web applications to become Semantic, and a state of the art in online social networks, describing their social features and collected social data. The state of the art presented in this chapter resulted in a still unpublished paper entitled *Collective Intelligence in Toursplan*.

Chapter 3 provides a brief description of the SNA domain, describing measures and algorithms for the analysis of social networks, with a special focus on community mining. Also, the SocioNet Ontology (SNO) is presented, which consolidates the presented SNA state of the art. The study presented in this chapter resulted in the article *A Survey of Social Network Analysis*, submitted to the Journal of Web Semantics.

Chapter 4 presents a framework, along with the Toursplan Ontology (TPO), allowing the extraction of relevant social interaction data from triple datasets, as well as the execution of SNA algorithms and the storage of their input and output data according to the SNO.

Chapter 5 presents the proposed Social and Semantic Web system architecture for Toursplan, describing the implementation efforts and some execution results regarding the evaluation of the framework presented in chapter 4.

Chapter 6 contains conclusions and future work based on current limitations and new features.

Chapter 2
Bridging the Social and Semantic Web

By now, the Web is well known for the social features introduced by online social networks. These features, when implemented and used adequately, have manifested themselves useful in many contexts, enhancing Web applications with domains that range from professional to recreational. This new Web, where people socialize and interact, is often referred to as the Social Web and came as a wave that swiftly transformed the old Web into a social environment.

The Semantic Web, on the other hand, is being slowly assimilated. Unlike the Social Web and its antecessor, which are about making human-readable linked documents available, the Semantic Web focuses on making semantically enriched and machine-readable linked data available. As machines become able to understand the semantics of data, a new wide range of possibilities for machines to actively assist us in multiple tasks emerge.

To achieve such a web of data, multiple technologies have been devised, belonging to different layers of the Semantic Web stack (see figure 1).

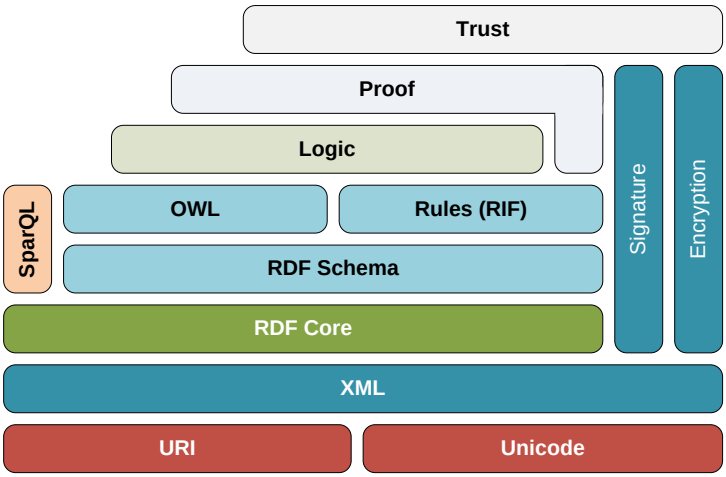

Figure 1 - The Semantic Web stack.

The foundation of the Semantic Web still relies on URIs (Uniform Resource Identifiers). However, multiple representation languages, with different roles, as RDF (Resource Description Framework), RDF-S (RDF Schema), OWL (Ontology Web Language), RIF (Rule Interchange Format) and SWRL (Semantic Web Rule Language) have been added to the stack:

- RDF is used to represent data in the form of triples. RDF also describes resources all over the web, identifying them through RDF URIs;
- Metadata is usually modeled in RDF-S or OWL, describing the domain of knowledge. OWL is a language capable of describing semantically richer models than RDF-S;
- Additionally, rules can be included to assist automatic reasoning processes. The languages used to represent such rules are usually RIF or SWRL.

As the Semantic Web merges with the Social Web, the social features and interactions begin to be connected and described through semantically enriched models with rules. All these data and metadata along with advanced reasoning processes can lead to collective intelligence in a whole new level. Figure 2 provides a view over the evolution of the Web, according to both the social and data connectivity dimensions.

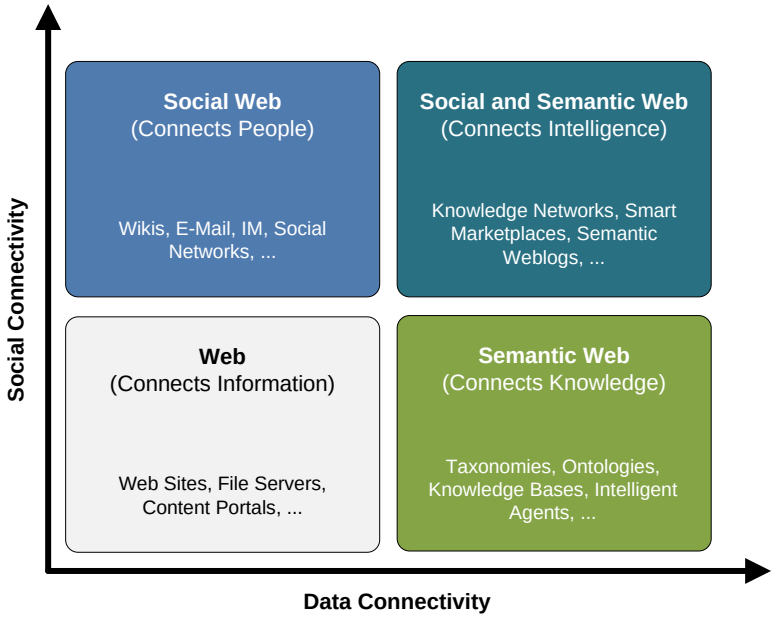

Figure 2 - Social and data connectivity and the World Wide Web (Spivack 2004).

2.1. Online Social Network Applications

Currently, some of the most popular online social networks are Facebook, Twitter, Hi5, MySpace and LinkedIn. While LinkedIn focuses on professional connections, the others focus on connecting friends and sharing information among them. Social features are also present in web applications with functional purposes like billing, invoicing, management, and traveling. Almost anything, if not everything, can be referenced or involved in social interactions. Traveling is of special interest for this work, since it unfolds in the context of the Toursplan project.

Lately, a lot of online tourism social networks have emerged or evolved to be the successors of the previous tourism information systems. Some examples are TripAdvisor, TripWolf, Dopplr, Wayn, TripSay, Driftr, Sair+, Real Travel, TravBuddy, Exploroo and TripConnect.

Most of the existent online social networks do not provide published scientific work regarding their structure and features. In that sense, most of the following state of the art description is based in guidelines by Porter (Porter 2008), and the observation of existent online social network interfaces and features.

2.1.1. Facebook

Facebook is an online social network for people to keep up with friends and share multimedia content about their lives or interests (e.g. links, videos, photos, events and notes). Facebook contains many social features worth analyzing (e.g. the news feed, profile wall, notifications and control features). For instance, the news feed, when provided with proper control features, is a powerful source of information that merges all recent activity from friends in only one place. Along with the notifications

area, the news feed page keeps the user up to date without the hurdle of having to go through multiple pages.

Facebook features third-party applications (e.g. from TripIt and TripAdvisor) that can exploit local and external social network data and enhance the profile of users for as long as they allow it.

2.1.2. LinkedIn

LinkedIn focuses on building professional profiles and connections. It promotes profile completion in multiple languages and features recommendations and messages between professionals. LinkedIn also allows the creation of groups, and includes features for seeking and posting jobs.

The first time a user visits LinkedIn, a small description of what it is and does is given. To join the network, only the first and last names, email and password are required.

Similarly to the Facebook news feed, the LinkedIn home page shows the users' network recent activity. It also suggests new people to add to the network, shows how many people has been visiting the users' profile, network connections, and pending invitations and messages.

A LinkedIn professional identity is given by a profile, which resembles a curriculum vitae enriched with external applications that allow the inclusion of external data (e.g. blog entries, files, slides and favorite books).

2.1.3. TripAdvisor

TripAdvisor is a social network that encourages the user to submit reviews and participate in the generation of collective intelligence. Its main focus is traveler reviews and opinions, receiving more than 32 million visitors each month.

When anonymous to TripAdvisor, a user is able to browse through the vast amounts of provided information: hotels, restaurants, vacation rentals and things to do. The user can either search using keywords, or browse all the POIs (Points of Interest) associated with a destination. Each POI contains information like photos, videos, reviews and descriptions that can be provided by users.

Hotels, restaurants and things to do are accessible through destinations. The destination page contains a lot of important information like aggregated multimedia content, top-rated POIs and frequently asked questions.

TripAdvisor allows the user to log-in using a Facebook account, and the loading of the users' friend list from an e-mail account like Google Mail, Hotmail and Yahoo. The registration process is simple, asking only for absolutely necessary information like e-mail and password. Also, the editable user profile contains little demographic information and preferences.

In TripAdvisor, a user can plan new trips and keep track of previous ones with trip folders. Trip folders contain notes, links and POIs. The planning process is manual, since there is no semi-automatic planning feature to assist the user.

TripAdvisor also features Travel Maps. This means that the user is able to create a map containing relevant destinations he would like to, or already has, visited.

2.1.4. TripWolf

Being similar to TripAdvisor, TripWolf focuses in travel tips and opinions, planning and booking. It claims to have trusted information for more than 50.000 destinations containing POIs such as accommodations, sights, beaches and transportations.

The trip planner and travel journal features, are probably the only ones that widely distinguish TripWolf from TripAdvisor. The trip planner is a small box that assists the user with keeping track of the POIs currently in his trips (see figure 3).

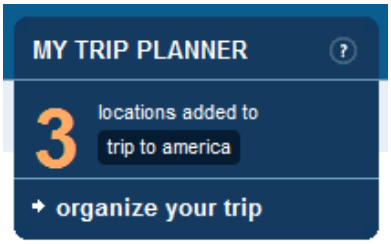

Figure 3 - TripWolfs' trip planner feature.

When registered, the user is able to create travel journals containing journal entries with geographic and multimedia information attached. Planned trips can be shared and rated by other users, and only include POIs from TripWolf. It is also possible to generate a trip guide PDF (Portable Document Format) with detailed information about nearby POIs. Although the generation of a PDF guide resembles, in a way, a semi-automatic planning process, the guide does not provide a personalized daily schedule of the trip.

Instead of focusing on friendship, TripWolf social relations reflect if the user's following another user activities (e.g. the submission of journal entries, ratings and comments) inside the network.

TripWolf also has a "tips from your friends" area, where recommendations are given according to the user social circle (followers or those the user is following), for a given destination.

The TripWolf data model is well structured, classifying places as continents, countries, regions, cities or villages. POIs belong to a place, and are organized by multiple categories: Places to Visit, Culture & Sights, Beaches, Nature & Adventure, Hotels & Accommodations, Eating, Shopping, Transportation, Nightlife & Entertainment, Working & Studying, and Sports & Wellness.

2.1.5. Dopplr

Dopplr claims to be a service for smart international travelers. It helps users with sharing personal and business travel plans privately within their social circle. To achieve this, Dopplr generates collective intelligence by analyzing travel trips and tips, providing travel patterns from the world's most frequent travelers.

While using OpenID accounts, Dopplr (as well as TripAdvisor, TripWolf and TripSay) is connected with some of the most popular online social networks like Flickr (from where it gets photos), Facebook, Twitter and LinkedIn.

Places include some interesting statistics, like a chart with the amount of trips over time. On the other hand, Dopplr doesn't seem to allow users to add new places or POIs, classified under the Eat, Stay or Explore categories.

Trips contain information like transportation means, dates, notes and a description. Later on, the user can add pictures from Flickr, and check out Dopplrs' recommendation for places to stay around the trip location.

Also, its adaptive procedures seem to rely mostly on user ratings and other possibly implicit feedback features.

2.1.6. TripSay

TripSay is an online travel social network for sharing trip information and experiences with friends. It mainly focuses on providing personalized recommendations according to the user (and like-minded users) interests.

In TripSay POIs and places are presented in a map. According to the zoom level of the map, the recommendation system filters POIs to avoid the presentation of excessive information. Figure 4 shows the TripSay map interface for rating places.

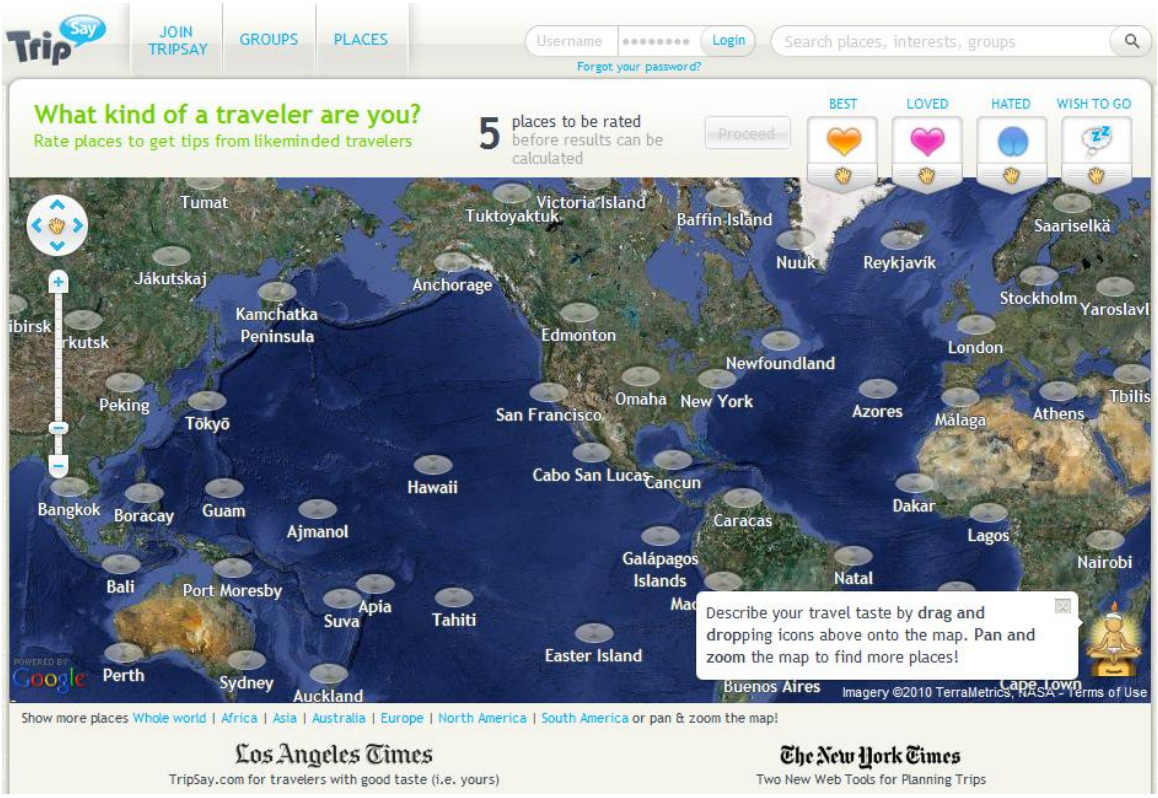

Figure 4 - TripSay rating map interface.

For the recommendation system to work properly, the user must rate several places. Subsequently, TripSay shows POIs that can be filtered according to multiple categories (e.g. airport, hotel, restaurant, bar, beach, diving, sight, shopping, mountain and fun).

TripSay supports follower relationships between users and allows the creation of groups. Inside groups, users can share POIs and get involved in discussions. Users can also submit messages with photos or links attached, to their profile wall.

2.1.7. Driftr

Driftr promotes user-generated content for travelers, and is still at its beta version. As well as in TripSay, it doesn't seem to make a distinction between places and POIs. Instead, users add trips to a specific place including information from personal or external sources about where to stay and eat, what to do, and how to get around.

Currently, Driftr does not seem to have many mechanisms for garnering collective intelligence since users can't provide feedback on the content submitted by others.

2.1.8. Sair+

Sair+ (Vieira & Lavos 2010) is a young online tourism social network that allows its users to share geo-referenced tourism experiences and opinions. While Sair+ does not provide any planning feature, it does provide recommendations and services for buying and booking online.

The users' personal area contains data about preferences and supports the interaction with other users through follower or friendship relations. There is also support for listings of user galleries and given ratings.

The recommendation system relies mainly on tags and ratings, filtering items according to the frequency of occurrence of the ratings given by the user neighbors, i.e. similar users with common tastes.

Sair+ seems to poly-classify POIs according to multiple classes (e.g. hotels, restaurants, places to go, shops, destinations, culture & leisure, kids, and nightlife).

2.2. Tourism and Travel Social Network Comparison

According to the key ideas for creating and designing for the social web presented in (Porter 2008), an empirical analysis of the previously mentioned online social network features is presented.

To gather interest, an application should provide a concrete and simple description of what it does, where and how it works, and why should someone use it (Porter 2008). Subsequently, interaction with the user should be easy and friendly, reducing friction and leading the user to the passionate use of the application.

In that sense, a set of key aspects, which need to be accounted for when building an online social network, can be identified (Porter 2008):

- **Communication** (how well the interface transmits the purpose and features of the application);
- **Friction and Engagement** (the application should only ask for absolutely necessary information at each step);
- **Motivation** (how well the user feels motivated to take advantage of the application);
- **Adaptation** (how dynamic is the content, and how well it evolves through time leading to collective intelligence);
- **Sharing** (if social objects can be shared inside and outside the network).

A motivated user takes advantage of the online social network to share information. Eventually, adaptation mechanisms like recommendation systems go after the submitted information and turn it into collective intelligence.

According to (Porter 2008), by focusing on the user, the online social network becomes more appellative. Some key ideas to leverage motivation are:

- **Identity** (e.g. dynamic and personalized profile pages);
- **Uniqueness** (e.g. the submission of opinions through ratings, reviews and comments);
- **Reciprocity** (e.g. notifications from friends' activity promoting replies);
- **Reputation** (e.g. the number of friends, number of reviews written or the number of fans);
- **Control** (e.g. controlling if personal content can be seen by everyone, friends of friends or only friends).

It's imperative to provide good features that greatly help the tourist when traveling while also avoiding complexity and feature creep. Therefore, specific features that fit the online social network domain and main activities should be included. The main activities of tourism/travel online social networks may differ slightly, but they all seem to focus on providing the user with features to find travel advice, plan trips, and share travel plans.

TripAdvisor, TripWolf and TripSay are strong in progressive engagement, and reduce friction by asking for a minimum amount of information when the user signs up. Although it does not reveal how it works in a straightforward manner, TripAdvisor transmits its main purpose in a very simple way through one statement: "Over 30 million trusted traveler reviews and opinions".

Reducing sign up friction involves adaptive sign up pages (e.g. supporting OpenID sign in), a good communication of the purpose and features of the application (to capture the immediate interest of the user) and progressive engagement (asking only for absolutely necessary information at each step).

Figure 5 illustrates the observed characteristics of each tourism/travel online social network. This figure represents the systematization of the performed empirical state of the art in tourism online social networks.

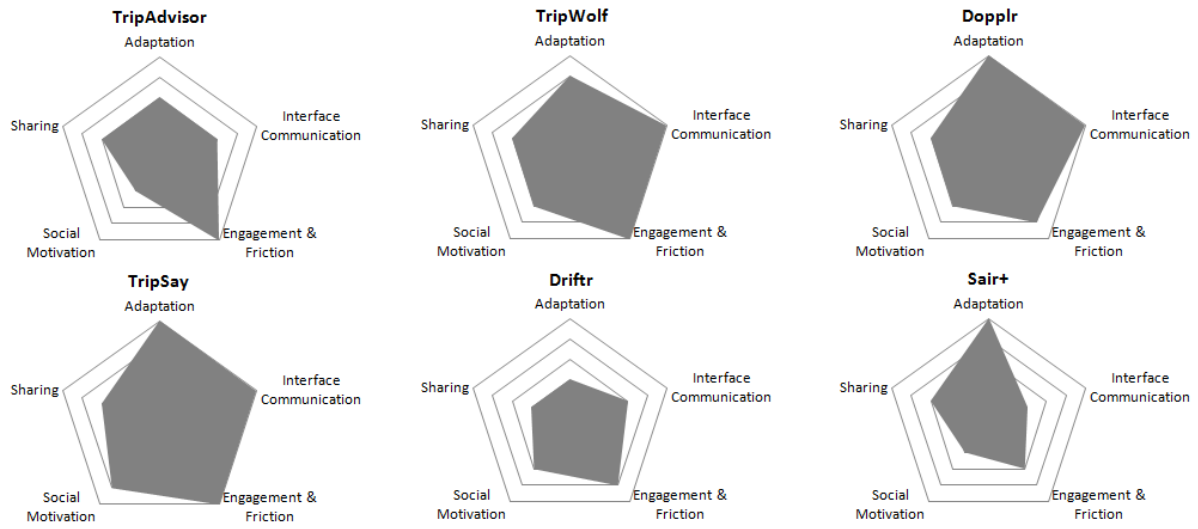

Figure 5 - Comparison of six popular and/or recent online travel social networks in a scale from 0 (not tackled at all) to 5 (very well tackled).

To easily choose and implement useful collective intelligence features in Toursplan, some of the most common features among tourism/travel online social networks that reflect collective intelligence were identified:

- Top items by week, month, category or overall;
- Best and worst revealed;
- Highly rated items nearby;
- "People here also go to…";
- Popular items in the same place as this one;
- Friend suggestions;
- Featured item.

Motivation levels not only have a huge impact over the number of users, but also show whether or not the domain features are working as expected. Table 1 describes features that can enhance motivation.

Table 1 - User motivation in tourism online social networks.

Motivational Aspect	Features
Identity	Profile with personal information, activity page, trips and trip journals plus multimedia information.
Uniqueness	Profile with reviews, ratings and given comments.

Reciprocity	Notifications resulting from friend activity, promoting further interaction.
Reputation	Profile with number of reviews, ratings and friends; profile with friend feedback.
Control	User can control who sees what in his profile: private, friends, friends of friends or public.

Most of these features, such as the user profile and activity page, are usually available after the user is authenticated. However, features like searching for places or points of interest using simplified filtering and recommendation mechanisms are available to anonymous users.

As well as points of interest, places are important social objects. Tourism online social networks use place pages to merge information from multiple points of interest in only one page. On the other hand, the POI pages usually contain reviews, ratings, comments, photos, videos and other information about the POI. They can also feature a Q&A system through which users can submit questions and answers about the POI. All questions and answers can be rated.

This analysis can serve as the basis for the implementation of a new Toursplan Social Web application user interface, and for the creation of the Toursplan knowledge domain model. The creation of the Web application user interface has been left outside the scope of this thesis.

Chapter 3
Survey of Social Network Analysis

The analysis of social networks gives answers to important social and behavioral science questions by providing precise formal definitions to aspects of the political, economic, or social structural environment (Wasserman & Faust 1994) (patterns or regularities among social relationships), allowing practical applications that go from elaborating specific marketing campaigns to finding terrorist cells (Mishra et al. 2007). But, the analysis of such large scale social networks is currently incomputable since they are often formed by millions of actors, a fact that also hinders a practical view over the network. This problem has led to the application of clustering techniques to social networks.

Social network clustering, also referred to as community mining, mimics the usual human behavior and tendencies to form clustered relationships and has been supported by multiple SNA measures (M. E. J. Newman & Girvan 2004; Girvan & M. E. J. Newman 2002). Identified communities can be further explored, resulting in the extraction of detailed and significant knowledge on practical scenarios. This is called a resume mining of communities problem (Wu et al. 2007), and involves fundamental questions like:

- What is the ultimate cause for the formation of the community?
- What are the unique community features?
- How did the network and its communities evolve through time?

The practical usability of such knowledge is still being investigated in multiple contexts. An example would be the elaboration of specific directed marketing campaigns, where the campaign focus meets the needs of certain groups of actors or communities (Kozinets 1999). Also notice that actors in a social network can be people, departments, enterprises, universities, countries, or even abstract

entities (e.g. hierarchy levels of a company). In addition to actors, social relations can be of any kind although the most common application relies on friendship (e.g. found in Facebook, MySpace, Orkut and Hi5) and professional relations (e.g. found in LinkedIn).

3.1. SNA Theory and Concepts

A social network (Handcock et al. 2007) is made of relations that represent not only the presence or absence of a relationship between social entities or actors, but also its strength or intensity. Social networks are usually modeled using graphs and adjacency matrices. Graph theory not only provides a vocabulary capable of addressing many social structures, but it also provides concepts and ideas for the quantification and measurement of social network properties (Wasserman & Faust 1994).

There are several different network graph models distinguished by their type of entities and relations (Wasserman & Faust 1994). Although most networks contain representations of only one type of entity, different types can co-exist inside a network to represent relations between entities (like people and banks), or even between entities and events. These last networks, with multiple entity types, can be referred to as *multi-mode networks* (see figure 6a).

Networks can also contain multiple types of relationships. These are called *multi-relational networks* (Cai et al. 2005), and can be modeled using a multigraph (see figure 6b).

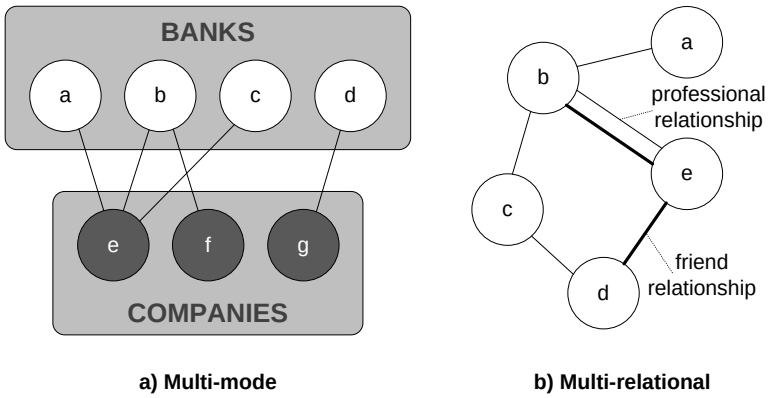

Figure 6 - Multi-mode and multi-relational networks.

Different representations of relationships result in different types of graphs like digraphs (directed graphs), signed graphs (where edges have positive or negative meaning), weighted graphs (with valued edges), fuzzy graphs (with fuzzy edges) or Markov chains (where the valued edges contain probabilities) (Yang et al. 2008).

A graph can also be characterized according to its evolution over time. A fixed graph is said to be fixed in time, while dynamic graphs change over time.

Some of the most important concepts and properties in graphs are:

- **Node degree**: the amount of direct edges connected to the node;
- **Length of a path** (or distance): the number of edges in the path;
- **Weight of a path** (or distance/length in weighted graphs): the sum of the weights in the path's edges;
- **Geodesic**: the shortest path between two nodes;
- **Graph density**: how close the number of edges in the graph is to the maximum allowed. Graphs with low density values are known as sparse graphs (see figure 7);
- **Cycle**: a path of at least three nodes where all edges and nodes are distinct, except for the first and last nodes, which are the same;
- **Semicycle**: a digraph cycle where the direction of the edges does not matter, as long as there is a connection between nodes;
- **Clique**: a graph or subgraph (undirected) with maximum density, meaning that it has all possible edges between nodes;
- **Graph diameter**: the longest path between any pair of nodes;
- **Graph average distance**: average distance of the shortest path (geodesic) between any pair of nodes in the graph.
- **Dyads and triads**: a set of two or three nodes respectively and the possible ties between them.

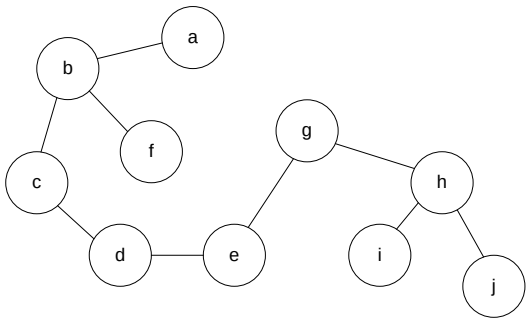

Figure 7 – Network graph with 9 edges over 45 maximum possible edges.

Notice the difference between graphs in figures 7 and 8. Although both contain the same amount of nodes (thus the same maximum number of possible edges), the graph in figure 8 is denser than the graph in figure 7 since it contains more edges. In other words, in figure 8, the number of actual edges is closer to the maximum number of possible edges.

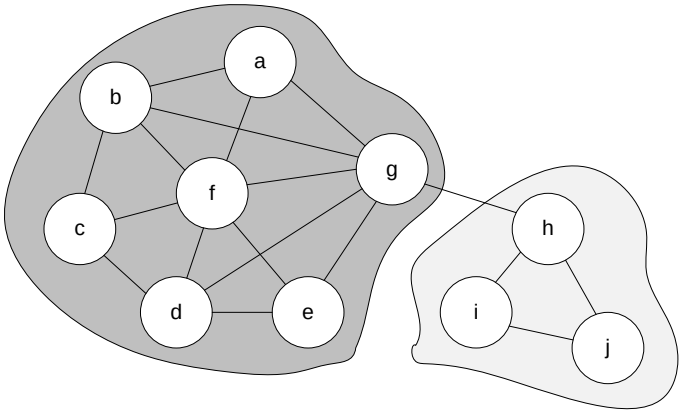

Figure 8 – Network graph with two dense clusters. The graph contains 18 edges over 45 maximum possible edges.

In directed graphs, as there are inward and outward edges, there are also two different node degree concepts, known as indegree and outdegree. In figure 9, node *a* contains more inward edges than node *b*, thus having a higher indegree. On the other hand, node *a* contains no outward edges while *b* contains two, meaning that node *b* has a higher outdegree.

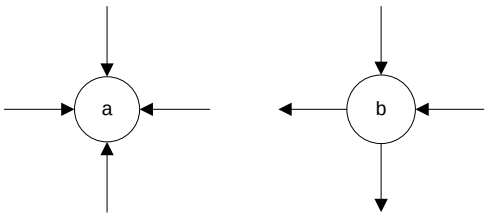

Figure 9 – Node degree in directed graphs.

Table 2 introduces symbols for the previously described graph concepts, which will be used further in this thesis.

Table 2 - Symbols for graph concepts.

Symbols	Definitions
G=(V,E)	A graph with nodes V and edges E.
n	The number of actors or nodes.
m	The number of relations.
v_i	Node i.
$d(v_i)$	Degree of node i.
$d_I(v_i)$	Indegree of node i.
$d_O(v_i)$	Outdegree of node i.

$\mathbf{g_{jk}}$	Geodesics between any pair of nodes j and k.
$\mathbf{g_{jk}(v_i)}$	Geodesics between any pair of nodes j and k that go through node i.
$\mathbf{\Gamma(v_i)}$	Neighborhood of node i (set of directly connected nodes).
$\mathbf{w(v_i, v_j)}$	Weight of the edge incident to nodes i to j (if it exists).
deg(A)	Sum of all node degrees within a subgraph or partition A.
$\mathbf{d(v_i, v_j)}$	Distance (or weight) of the geodesic path that connects the nodes i and j.

In social networks, the intrinsic aspects of human cooperation shape the network graph. This means that, by using a graph to store a set of actors and their social interactions, it is possible to check for behavioral patterns using the previously described concepts.

One of these aspects is the tendency to relate to those related to someone known. This leads to the creation of clustered relationship networks (Levine & Kurzban 2006), containing loosely connected groups of highly connected actors. Notice the triple {*h*,*i*,*j*} in figures 7 and 8. In figure 7, actor *h* knows both *i* and *j* but actors *i* and *j* don't know each other. Since *h* knows them both, there is a tendency for them to relate, thus forming a clique (as in figure 8).

Another factor that highly affects the creation of clustered relationships is the geographical distance between actors. Some research work (Kadushin 2004) showed that people are more likely to build relationships if they are geographically close to each other. This phenomenon, often referred as propinquity, can be defined as being at the same place at the same time (Kadushin 2004).

On the other hand, homophily which is the tendency to relate to similar individuals, not only affects the formation of clusters, but can also give them meaning. In cases of substantial homophily, it can be stated that individuals within a cluster have something in common, so clusters could be seen as types, categories or even stereotypes of individuals.

In his work, Jackson, M. (Jackson 2008) analyzed the effects of homophily over a network, concluding that homophily has direct influence over the density of the network. However, according to the same author, homophily does not affect the network graph diameter and average distance.

Clustering has a great impact over social networks. Information flows easily inside clusters, but communication between clusters is usually limited by weak ties, meaning that relationships between clusters are usually made by a very few individuals, known as brokers. This is the case of nodes *g* and *h* in figure 8.

Broker nodes/actors, that connect different clusters, are important since they control the flow of information between different clusters. Monitoring the flow of information over brokers can provide important knowledge about the data flowing between clusters.

Dense networks tend to have fewer weak ties, showing better results in the flowing of knowledge, suggesting easier knowledge acquisition. Sparse networks, which are more likely to contain weak ties, are usually better for the discovery of new knowledge, i.e. for the introduction of diversity among existent knowledge.

Since both types of networks give different advantages and disadvantages, a social network should be composed of both dense and sparse areas simultaneously (Levine & Kurzban 2006), providing balanced knowledge acquisition and discovery.

3.2. SNA Properties and Measures

Social network analysis measures can help the extraction of important information from the network. Most measures focus on either the characteristics of one node, dyads, triads, or subgraphs, and try to:

- **Identify prominent nodes** (Wasserman & Faust 1994). Typical measures include node degree, closeness, betweenness, information and rank (or status) measures. These measures are described further in the rest of this section;
- **Evaluate community quality**, like the modularity (M. E. J. Newman & Girvan 2004; Chen et al. 2009) and compactness (Duan et al. 2009) measures;
- **Perform positional analysis** (structural, regular and automorphic equivalence) (Fan et al. 2007). In other words, analyze the social position and roles of the social network actors;
- **Identify or exploit the nature of social relationships**, i.e. through balance, clusterability and transitivity (Wasserman & Faust 1994). An example of such a measure is the structural similarity proposed in SCAN (Xu et al. 2007).

Some of these measures (like the modularity measure) are actually the basis of many community mining algorithms, enhancing the quality (M. E. J. Newman & Girvan 2004) of the resulting communities, or providing additional knowledge about the network structure. Notice that because of this, some measures are covered later in section 3.3 (Community Mining).

3.2.1. Prominence Measures

There are multiple measures for prominent node identification (see table 3 for symbols), also known as *centrality* and *prestige* measures (Wasserman & Faust 1994). While centrality focuses on both inward and outward relationships, prestige only focuses over the inward connections of the node. In other words, a prestigious node is a recipient; object of an extensive quantity of inward relationships.

Table 3 - Symbols for SNA measures.

Symbol	Definition
$\mathbf{C_D(v_i)}$	Actor degree centrality.
$\mathbf{C_C(v_i)}$	Actor closeness centrality.

$C_B(v_i)$	Actor betweenness centrality.
$P_D(v_i)$	Actor degree prestige.

The simplest centrality and prestige definitions rely on node degree. Basically, as the amount of direct relationships of a node increases, the higher its centrality and prestige becomes.

In that sense, nodes *a* and *b*, of figure 9, have the same degree centrality but not the same prestige centrality. Node *a* degree prestige value is higher than that of *b*.

A proposed node degree centrality measure that allows the comparison of values across networks of different size is (Wasserman & Faust 1994):

$$C_D(v_i) = \frac{d(v_i)}{n-1} \quad (1)$$

Being the equivalent degree prestige measure represented as:

$$P_D(v_i) = \frac{d_I(v_i)}{n-1} \quad (2)$$

Closeness centrality focuses on how close a node is to all other nodes, and can be defined as the inverse of the sum of the distances from a node to all the other nodes:

$$C_C(v_i) = \frac{n-1}{\sum_{j=1}^{n} d(v_i, v_j)} \quad (3)$$

Betweenness centrality measures the influence of a node or edge over the flow of information over the network. Edge betweenness can be defined as the number of geodesics between any pair of nodes, which run through that specific edge. For networks containing loosely connected clusters, the edges (or nodes) with high betweenness centrality will be the ones connecting communities. Looking back at figure 8, the edge connecting nodes *g* and *h* has a high betweenness value, since it belongs to all the geodesics that connect nodes from one community to the other.

A node betweenness formalization based on Freemans' definition (Freeman 1977) is:

$$C_B(v_i) = \sum_{j<k} \frac{g_{jk}(v_i)}{g_{jk}} * \frac{2}{(n-1)(n-2)} \quad (4)$$

The betweenness centrality only focused on geodesics, but since in reality the flow of information does not always go through geodesics, an extension of this measure emerged, called information centrality. Information centrality considers all paths between pairs of nodes inside a network (Wasserman & Faust 1994), giving results according to the "information" of the paths, i.e. according to the inverse length of the paths.

Rank (or status) prestige follows the principle that a nodes' rank should be dictated by the rank of those which relate to the node. This is quite a difficult problem to solve, since it turns out to be an infinite regression. Some methods exist to give this idea some practical use, but they all impose some constraints on the network (Wasserman & Faust 1994).

3.2.2. Positional and Role Analysis

Positional and role analysis are used to evaluate how two actors are structurally similar or how relations in multi-relational networks show meaningful patterns of relationships. For example, to check if two actors both play the same role or have the same social position (Hanneman & Riddle 2005).

When performing positional analysis, an equivalence definition must be used to evaluate if two or more actors belong to the same equivalence class or position. For two actors to be structurally equivalent they must be identically related to identical other nodes, through their inward and outward relations (see figure 10).

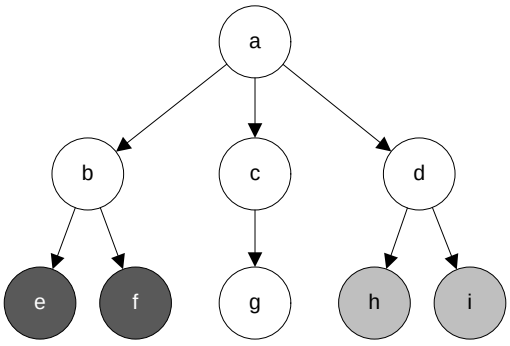

Figure 10 - Structural equivalence: nodes with the same fill color are structurally equivalent (belong to the same equivalence class). White nodes do not have any structurally equivalent nodes (Wasserman & Faust 1994).

Structurally equivalent actors are substitutable, meaning that they can be merged and represented as a single equivalence class without the loss of structural information (Wasserman & Faust 1994).

Another notion of equivalence comes from the idea of graph isomorphism and is called automorphic equivalence. Two graphs are isomorphic if there is a bijection (a one-to-one correspondence where no element is left unmapped) between the nodes in one graph and the nodes in the other, such that connections between nodes are preserved (Wasserman & Faust 1994). An automorphism, on the other hand, is an isomorphism that focuses over a single graph; it can be seen as a graph isomorphism over the original graph and its clone. Hence, automorphic equivalence exists between two actors if there is an automorphism that maps one of the actors to the other. Figure 11 contains an example of automorphic equivalence over the graph in figure 10.

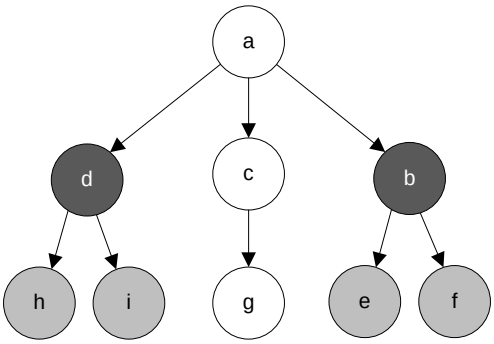

Figure 11 - Automorphic equivalence: nodes with the same color are *automorphically* equivalent. White nodes do not have any *automorphically* equivalent nodes (Wasserman & Faust 1994).

Regular equivalence provides another definition for positional analysis, and it is quite interesting since it explores the notion of relative roles. It can be defined as being a measure that evaluates how two actors equally relate to equivalent others (instead of identical others). This removes the restriction from the two previous equivalence notions, where ties to the original nodes had to be kept. If the edges in figure 12 were interpreted as “mother of” relationships and the nodes represent women, the network graph could represent a family tree containing only the women of the family, and the roles grandmother, mother and daughter could be distinguished after the application of a regular equivalence algorithm. Also, notice how nodes *g* and *h* both belong to the same equivalence class, “daughter”, even though they don’t have the same mother.

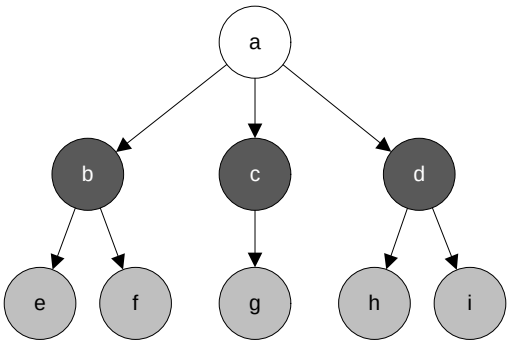

Figure 12 - Regular equivalence: nodes with the same color are regularly equivalent. White nodes do not have any regularly equivalent nodes (Wasserman & Faust 1994).

Regular equivalence has been generalized to fuzzy social networks (Fan et al. 2007). Fuzziness is a useful concept deeply rooted in nature, and can be applied to relations, thus representing both the qualitative relationship and the degrees of interaction between actors.

In (Fan et al. 2007), two alternative definitions of regular equivalence take place: regular similarity and generalized regular equivalence. Regular similarity is a fuzzy binary relation that specifies the degree of similarity between actors in a social network. Generalized regular equivalence is an equivalence relation that determines the role partition of the actors in a fuzzy social network.

3.2.3. Balance, Clusterability and Transitivity

The first steps toward social network clustering were taken when balance theory emerged (Wasserman & Faust 1994). The idea of balance requires a signed graph or digraph and studies how two actors perceive or react similarly toward the same object or actor. In figure 13a, triples are balanced, since actors *b* and *c* both either like or dislike actor *a* when they have a positive relationship between them, or disagree on their opinion of actor *a* when they have a negative relationship between them. Although in figure 13 only triples are shown, balance can be applied to cycles or semicycles of all sizes.

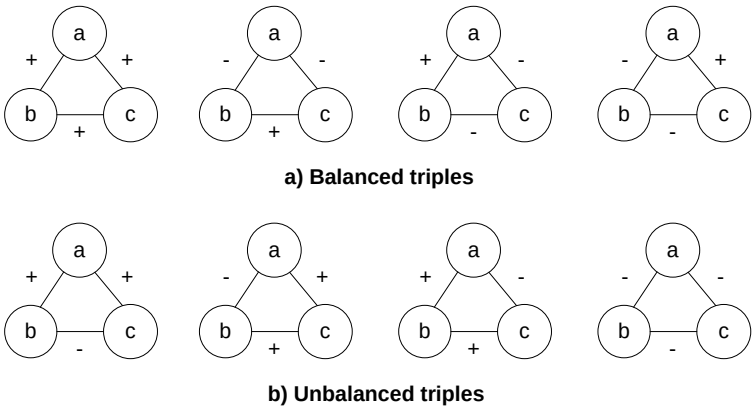

Figure 13 - Balance in all possible triples for signed undirected graphs (Wasserman & Faust 1994). Actors can be seen as people and their relations as how they like or dislike each other.

Balance can be seen as a measure of tension between a subset of actors (Wasserman & Faust 1994), and can be easily evaluated through the product of the signs found in the cycle edges. If the product is positive, then the cycle is balanced. Otherwise, it is an unbalanced cycle.

For balance to be extended to the whole graph, all cycles or semicycles within the graph must be considered.

An important characteristic of a balanced graph is that it can be partitioned in exactly two partitions connected by negative edges. Clusterability (Wasserman & Faust 1994) is an extension of balance that explored this characteristic by allowing the division of the signed graph in more than two

partitions. As far as clusterability is concerned, a graph is clusterable if it doesn't contain any cycle with only one negative edge.

Even so, clusterability cannot be used over unsigned graphs. To tackle this issue, a notion of transitivity (Wasserman & Faust 1994) emerged, stating that as long as every relationship is reciprocated, the graph is clusterable. This leads to disconnected partitions of reciprocally related nodes.

3.3. Community Mining

The importance of clustering in social network analysis has led to the emergence of the notion of community mining. In other words, it is the act of extracting communities from a social network, otherwise hidden or difficult to analyze.

The general definition for communities in social network analysis is that of loosely connected groups, each one containing highly connected nodes. Even so, researchers have proposed multiple different approaches that extend this general definition, and opinions on how to evaluate the quality of the discovered communities slightly vary. Also, the complexity and size of social networks contributed to the appearance of multiple community mining algorithms with quite different results in different practical scenarios.

Most of the classical partitioning methods (Chen et al. 2009) require input parameters, such as the number of communities to extract and/or their size. Typically in social networks, the quantity, size or even the characteristics of communities are unknown, a fact that usually makes these methods inappropriate for community mining. Attempts to solve some of these problems have been made, such as the normalized cut (Shi & Malik 2000) and the min-max cut (Ding et al. 2001), but they still favor the detection of roughly equally sized communities. This specially happens in scenarios where there are multiple optimal cuts and the fair cut is the one that provides an approximately similar amount of nodes to each community.

Classical partitioning methods did not adapt well to community mining in general. Being that graph partitioning is an NP-complete problem, the existence of several algorithms that are able to perform partitioning with reasonably good results and performance (Mishra et al. 2007; Xu et al. 2007; Duan et al. 2009; Yang et al. 2007; Zhou et al. 2002), has led to the emergence of many community mining approaches, each with advantages and disadvantages.

Clustering algorithms focus on object and/or relational data, i.e. relations between nodes/objects. Although community mining essentially relies on relational data, it can also explore the objects' data to improve the quality of the resulting clusters. Also, a criterion such as a similarity or proximity function *f* is usually present, so partitioning can be performed in a way that it correlates with *f* (Bansal et al. 2004). Suppose that you are given a graph G = (V, E) and a similarity function *f(a,b)* that returns "+" and "–" whether nodes *a* and *b* are similar or different. If the edges in E are labeled with the resulting

value of *f*, then the correlation algorithm is responsible for the creation of clusters that favor the existence of "+" edges inside the clusters, and "-" edges between the clusters.

Yang, B., Liu, J., Feng, J. and Liu, D. (Yang et al. 2008) classify community mining algorithms or methods as falling into two main categories: *optimization methods* and *heuristic based methods*. On the other hand, Chen, J., Zaïane, O. and Goebel, R. (Chen et al. 2009) classify community mining algorithms as being part of two groups defined as *graph partitioning* and *hierarchical clustering* (Girvan & M. E. J. Newman 2002; Chen et al. 2009; Xu et al. 2007). Similarly, a taxonomy by Jain, A., Murty, M. and Flynn, P. (Jain et al. 1999), classifies data clustering algorithms as falling under either *hierarchical* or *partitional clustering*.

In this thesis, a finer grain approach was taken to classify the methods and approaches as falling into the following classes:

- Cut-based Clustering;
- Multi-level Clustering;
- Flow-based Clustering;
- Hierarchical Clustering;
- Granular Clustering;
- Incremental Clustering.

Notice that these classes are not necessarily mutually exclusive, thus an incremental clustering algorithm can also be flow-based or hierarchical.

The following subsections elaborate on these classes.

3.3.1. Hierarchical Clustering

Hierarchical clustering can be defined as the traditional approach for detecting communities (Girvan & M. E. J. Newman 2002). A classical hierarchical clustering algorithm begins with the calculation of a weight for every pair of nodes in the network. This weight represents how closely connected the nodes are. Then, a new graph is created, where the edges are added according to their ordered weight values. As edges are added, communities emerge.

There are different definitions of weight. Two of the most popular definitions for weight rely on the total number of paths that run between two nodes. While one only considers node-independent paths (only paths that share none of the same nodes except for the first and last), the other considers all possible paths.

Hierarchical clustering methods allow the illustration of the cluster formation through *dendrograms* (see figure 14).

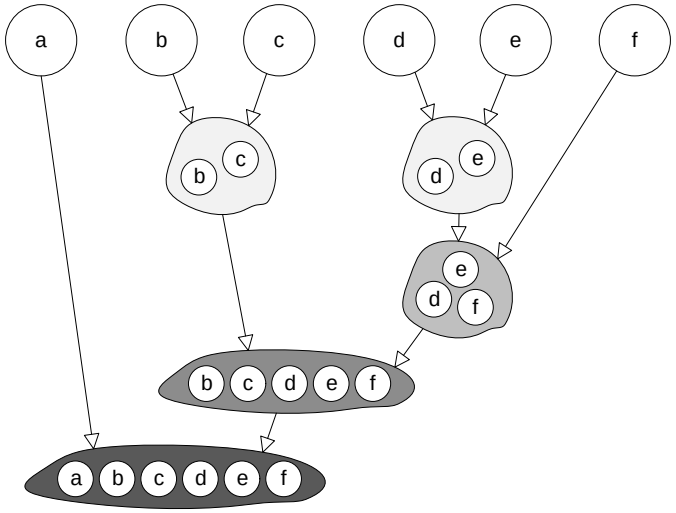

Figure 14 - Hierarchical clustering dendrogram[1].

From figure 14, for a hierarchical clustering algorithm that orders weight values from highest to lowest, it can be concluded that the highest weight values for the resulting community {*a*,*b*,*c*,*d*,*e*,*f*} fall into pairs {*b*,*c*} and {*d*,*e*}.

These traditional hierarchical algorithms usually give satisfying results in community mining, but they tend to separate peripheral nodes (e.g. a node connected to the network through only one edge) from communities since the weight value for these nodes is usually low. Also (Chen et al. 2009), although hierarchical clustering algorithms have the advantage of not requiring the size or number of groups to be found a priori, they are usually slow and their performance highly depends on the corresponding metrics, like the weight function.

To avoid some of the hierarchical flaws, a method is proposed in (Girvan & M. E. J. Newman 2002) that, instead of building communities from core nodes, uses nodes (or edges) that are most “between” communities. The algorithm first calculates a betweenness measure for all edges, removing the edge with the highest value. Then it recalculates the betweenness value for the affected edges and repeats the process until no edges remain. Since edges connecting communities have high edge betweenness, their removal will gradually cut the network graph into several partitions. The algorithm can calculate betweenness values for a graph with m edges and n vertices in $O(mn)$, thus giving the resulting hierarchical clustering algorithm a worst-case complexity of $O(m^2n)$.

Girvan, M. and Newman, M. also propose a measure to evaluate the structure and quality of a particular cluster, called *modularity* (M. E. J. Newman & Girvan 2004). Modularity follows the principle that a good division of a network is one in which the number of edges inside communities must be higher than in a case where the graph nodes were randomly connected.

[1] http://upload.wikimedia.org/wikipedia/commons/thumb/a/ad/Hierarchical_clustering_simple_diagram.svg/418px-Hierarchical_clustering_simple_diagram.svg.png

In hierarchical clustering, the modularity measure can be used to check which dendrogram level provides the best network partitioning, since the resulting modularity value for each possible partitioning usually indicates one or two peaks (M. E. J. Newman & Girvan 2004).

The original modularity measure has been subject to further research, which led to the identification of significant weaknesses. For example, the need to evaluate the entire graph structure, flaws when identifying communities smaller that a certain scale, and the impossibility to measure absent links inside communities. This last weakness is critical for community structure comparison between different graphs (see figure 15).

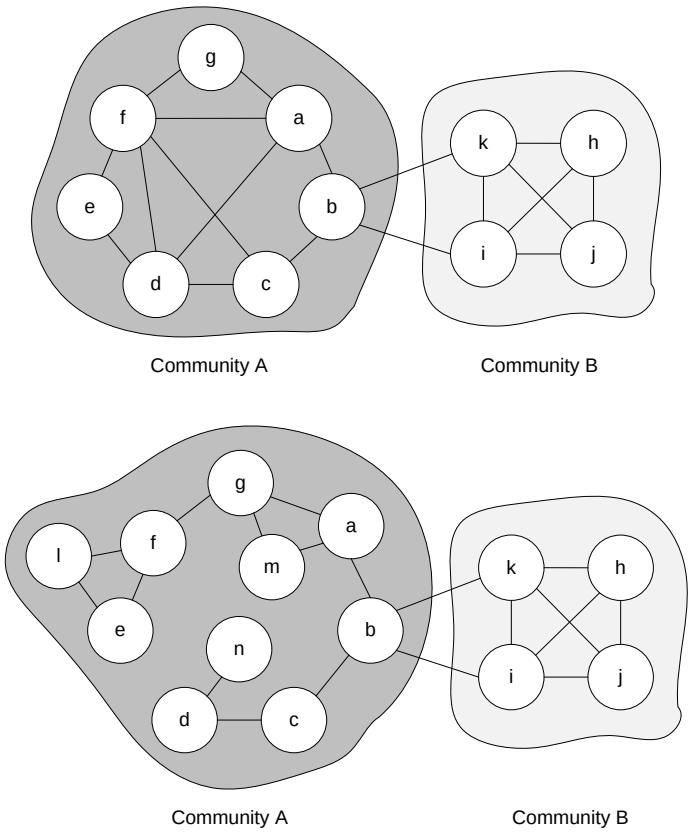

Figure 15 - Two networks with different structure, but with the same modularity values (Chen et al. 2009).

To allow the comparison of community structure quality between different graphs, a new improved modularity measure is proposed, namely the *max-min modularity* (Chen et al. 2009). The principle of the max-min modularity is minimizing the number of connections between clusters while maximizing those within clusters.

The Max-Min modularity not only rewards connections between nodes inside communities, but also penalizes disconnected nodes. However, although disconnected nodes do not have a positive impact over the community, they might not decrease the community quality value. Since disconnected pairs of nodes might represent unobserved connections, the Max-Min algorithm only evaluates certain unrelated pairs of nodes to inflict a negative influence over the quality value.

Most of the existent clustering algorithms do not distinguish the importance of nodes (like brokers) within networks. To tackle this issue, *SCAN* (Structural Clustering Algorithm for Networks) (Xu et al. 2007) brings the concept of hubs and outliers (see figure 16). Hubs (also previously referred as broker nodes) are nodes that connect different clusters but don't belong to any. On the other hand, outliers are nodes that only have a weak association with their cluster.

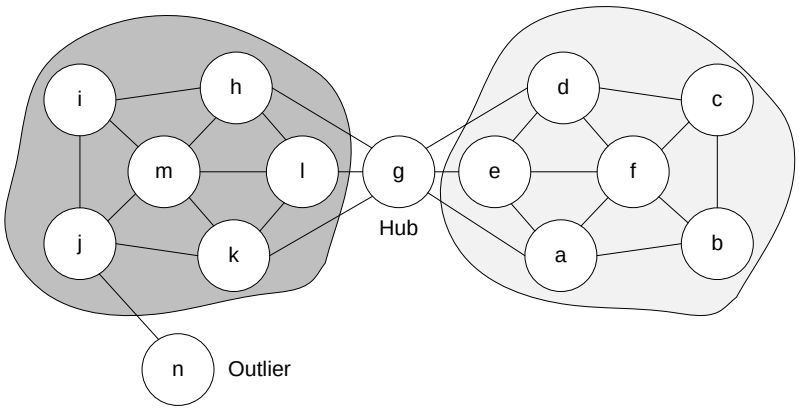

Figure 16 - SCAN: clusters, hubs and outliers (Xu et al. 2007).

SCAN focuses over undirected and non-weighted graphs and identifies isolated hubs and outliers, so both connectivity and structure is used to evaluate cluster quality.

Also, SCAN operates according to an extended notion of cluster, formalized as structure-connected clusters. This notion is an extension to that of density-based clusters and focuses on the neighborhood around two nodes. The intersection of the neighborhoods of two nodes, reveals common neighbors. This perfectly makes sense in a social network scenario, since the more a community resembles a clique, the more intimate it is.

Table 4 - Symbols for SCAN measures.

Symbol	Definition
$\sigma(v,w)$	Structural similarity between nodes *v* and *w*.
$DReach_{\varepsilon,\mu}(v,w)$	Structure reachability of node *w* from node *v*.
$Connect_{\varepsilon,\mu}(v,w)$	Structure connectivity between nodes *v* and *w*.
$N_{\varepsilon}(v)$	ε-neighborhood of node v.

Along with SCAN come multiple measures (see table 4), one of them being the structural similarity. The structural similarity described in (Xu et al. 2007), measures how similar the neighborhood of two nodes is. Given the adjacent nodes *v* and *w*, and their correspondent set of neighborhood nodes Γ(*v*) and Γ(*w*), the structural similarity between *v* and *w* is given by:

$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}} \quad (5)$$

The SCAN algorithm begins by finding core nodes. Core nodes must have at least *μ* neighbors with whom they share a structural similarity that exceeds a value equal to *ε*. The set of adjacent nodes that share a structural similarity *ε* or greater are called the *ε-neighborhood*. Also, clusters grow from core nodes, forming a cluster along with their ε-neighborhood nodes.

The concept of ε-neighborhood of a node *v,* $N_\varepsilon(v)$, is further explored in SCAN, where definitions like structure reachability and structure connectivity are used to cluster. A node *w* is reachable in a direct way from a node *v*, if *v* is a core and *w* belongs to *v*'s ε-neighborhood. Structure reachability is transitive, but it is only symmetric for a pair of core nodes.

$$D\,\mathrm{Re}\,ach_{\varepsilon,\mu}(v,w) \Leftrightarrow IsCore_{\varepsilon,\mu}(v) \wedge w \in N_\varepsilon(v) \quad (6)$$

On the other hand, two nodes *v* and *w* are structure-connected if they are reachable from node *u*.

$$Connect_{\varepsilon,\mu}(v,w) \Leftrightarrow D\,\mathrm{Re}\,ach_{\varepsilon,\mu}(u,v) \wedge D\,\mathrm{Re}\,ach_{\varepsilon,\mu}(u,w) \quad (7)$$

This leads to the formulation of a cluster *C*, where all nodes are structure-connected and structure-reachable. From this definition of cluster hubs can be easily detected by checking if a node contains neighbors inside two or more different clusters, and outliers by checking if they are not hubs and don't belong to any cluster.

As stated in (Xu et al. 2007), the SCAN algorithm is faster than the *FastModularity* algorithm proposed in (Clauset et al. 2004). The SCAN complexity is of O(*m*) while the modularity based algorithm runs in O(*md* log *n*), where *d* is the depth of the dendrogram describing the hierarchical cluster structure. Also, results from (Xu et al. 2007) demonstrate that the resemblance between the real world network clustering and the produced clustering is higher in SCAN than in FastModularity.

3.3.2. Cut-Based Clustering

One popular way of discovering clusters is the use of cut-based clustering algorithms. These algorithms try to split the network until a certain number of clusters (one of the input parameters) are found (see figure 17).

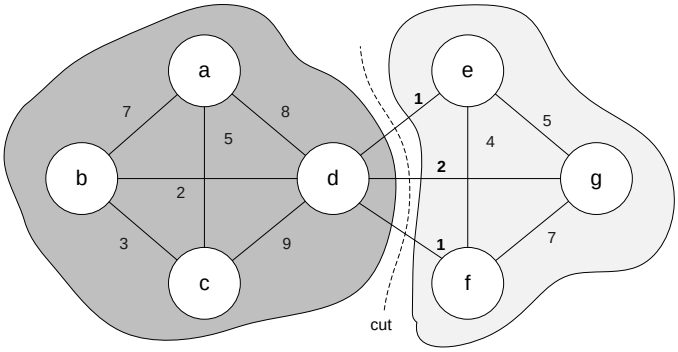

Figure 17 - Cut-based clustering, using the minimum cut criterion.

There are multiple cut criteria in literature, like the minimum cut, the normalized cut (Shi & Malik 2000) and the min-max cut (Ding et al. 2001). These criteria are described next.

Table 5 - Symbols for cut techniques.

Symbol	Definition
cut(A,B)	(Minimum) cut value between partitions A and B.
Ncut(A,B)	Normalized cut value between partitions A and B.
Mcut(A,B)	Min-max cut value between partitions A and B.

The minimum cut focuses on minimizing the cost of the cut, defined as the sum of weights of the edges that connect two clusters.

$$cut(A,B) = \sum_{i \in A,\, j \in B} w(v_i, v_j) \quad (8)$$

Since the minimum cut favored the discovery of small clusters, with isolated nodes, the normalized cut was proposed (Shi & Malik 2000). The normalized cut tackles this issue by evaluating the cost of the cut as a fraction of the sum of the weight of the edges that connect the cluster to the whole graph (or the sum of all node degrees within the cluster).

$$Ncut(A,B) = \frac{cut(A,B)}{\deg(A)} + \frac{cut(A,B)}{\deg(B)} \quad (9)$$

The min-max cut (Ding et al. 2001) follows the principle that similarity should be minimized between clusters and maximized within clusters. Thus, the cut cost is evaluated as a fraction of the sum of the weights of the edges within the cluster.

$$Mcut(A,B) = \frac{cut(A,B)}{\sum_{i,j \in A} w(v_i, v_j)} + \frac{cut(A,B)}{\sum_{k,u \in B} w(v_k, v_u)} \quad (10)$$

Cut-based criteria have been used in a variety of algorithms, some of which can be classified as Spectral algorithms (Mishra et al. 2007). Spectral algorithms can be easily implemented using standard linear algebra techniques, and their application relies on the eigenvectors and eigenvalues of the network Laplacian matrix to find adequate cuts and extract disjoint partitions (Von Luxburg 2007).

The specific minimum cut problem can also be solved using algorithms that have a close relationship to the famous “max flow-min cut” theorem by Ford and Fulkerson (Kleinberg & Tardos 2005), which proves that the maximum flow of a network is identical to the minimum cut that separates *s* and *t*. Nodes *s* and *t* are the source and the sink nodes respectively, and belong to two different partitions. In figure 17, nodes *b* and *g* are possible ideal source and sink nodes.

Another cut-based clustering algorithm is the *simple min-cut algorithm* (Stoer & Wagner 1997), which is able to identify the minimum cut for undirected and weighted graphs in an overall running time of $O(nm + n^2 \log n)$. Although the algorithm uses a parameterized start node, it can be modified so that the start node can be arbitrarily selected.

3.3.3. Granular Clustering

In reality, communities might not be delimited by strict boundaries, thus being overlapped to some extent. Besides, in some cases, the existence of overlapped clusters can also help the extraction of important nodes like brokers (see node *d* in figure 18).

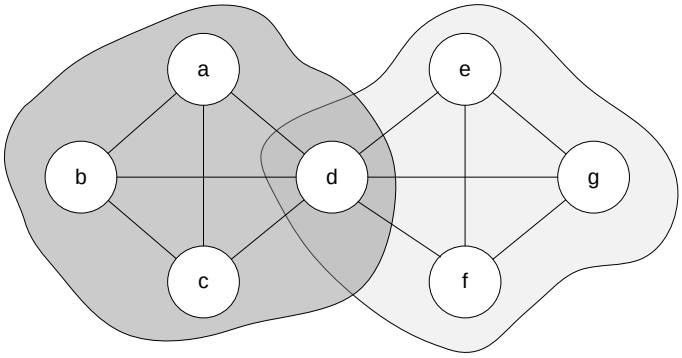

Figure 18 - Overlapped clusters.

The process of finding overlapped clusters is often referred as a granular computing problem. The notion of granules (Fan et al. 2007) is often defined as being fuzzy partitions that allow a small degree of overlap.

In the *concentric-circle* model (Zhou et al. 2002), a community is described as being a set of concentric-circles (see figure 19) that can overlap.

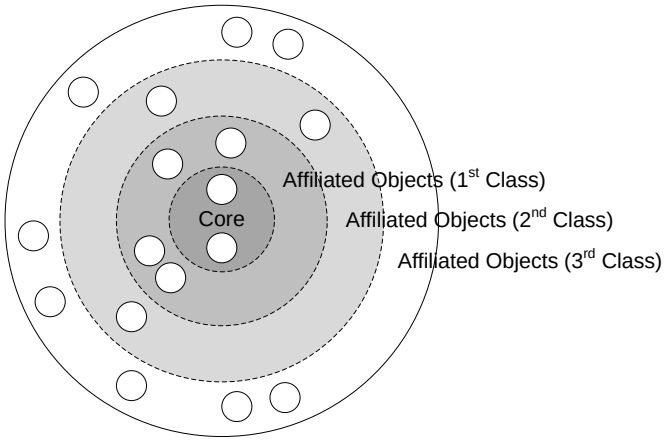

Figure 19 - A concentric-circle model for communities (Zhou et al. 2002).

The concentric-circle model relies on the following principles:

- Core objects and affiliated objects in a community are distinguished;
- The core of a community is made up of one or more objects;
- Hub objects (nodes pointing to many others) are not included in the core;
- It is not required that objects in the core of a community are tightly linked, since it might prevent the formation of some communities;
- Affiliated objects are expanded gradually according to their local hub value.

The concentric-circle communities can be defined by their core. Including multiple nodes inside the core is useful, since it helps characterize and distinguish the community with more detail. Core objects

are selected according to their prestige (indegree) value. Other core candidates to an already existent community can be selected by evaluating their explicit or implicit (connectivity to core members through other objects) relationships with core members. Affiliated nodes are then used to populate the communities according to the core inward connections.

To perform community mining using the concentric-circle model, a two phase algorithm (i) generates core sets and (ii) expands them with affiliated objects. Although the algorithm successfully generates communities, some granularity refinement is needed so that the final result does not contain a large amount of highly overlapped communities. This issue is solved by merging these communities.

The *(α,β)-clustering* method (Mishra et al. 2007) is also capable of finding overlapped clusters, though with a quite different approach. The (α,β)-clusters are internally dense according to *β*, and externally sparse according to *α*. More specifically, each vertex in the cluster is adjacent to at least a *β* fraction of the cluster, and any vertex outside of the cluster is adjacent to at most an *α* fraction of the vertices in the cluster. Since vertices outside the cluster should have less connections to vertices inside than those that belong to the cluster itself, *α* and *β* values are given according to:

$$0 \leq \alpha < \beta \leq 1 \quad (11)$$

Given the nature of the (α,β)-clustering problem, a restricted form must be specified so it can be applied to community mining. This restriction is applied to the *α* and *β* values so that a balance between internal density and external sparsity exists.

The adjacent (α,β)-clustering algorithm also introduces the notion of cluster *ρ-champions*. A node *v* is a ρ-champion of a cluster C, if the number of *v*'s neighbors not in C, is less or equal to a *ρ* fraction of the number of nodes in C:

$$\left|\Gamma(v) \cap V \setminus C\right| \leq \rho\left|C\right| \quad (12)$$

Basically, a ρ-champion is a node that has greater affinity with nodes inside the cluster than with those outside.

In a restricted α and β scenario, where all clusters are of the same size, the algorithm is able to find all clusters with ρ-champions in time $O(m^{0.7}n^{1.2}+n^{2+o(1)})$.

Although both the concentric-circle and (α,β)-clustering methods find overlapped clusters, there is no explicit focus over broker nodes, so it isn't clear if brokers are among nodes shared by multiple concentric-circle clusters or (α,β)-clusters.

3.3.4. Multi-Level Clustering

Multilevel partitioning (Fowler & Greenough 1998) approaches have been proposed to reduce the partitioning complexity of algorithms such as the spectral algorithm. Instead of trying to cut and partition the graph directly, multilevel algorithms first condense the graph through a quantity of levels. This condensation is achieved by merging nodes using a definition of distance between nodes, which varies from algorithm to algorithm.

The condensation process results in new nodes with weighted edges reflecting the number of edges that existed in the original graph. After condensation, cut-based methods like spectral bisection are used to perform the graph partitioning. Finally, the partitioning information needs to be transferred up through the condensation levels, so that a partitioning of the original graph is found.

An example of a multilevel partitioning algorithm is the multilevel k-way partitioning scheme for irregular graphs proposed in (Karypis & Kumar 1998). This algorithm generates disjoint and roughly equal-sized partitions. The process begins with a condensation of the graph until only *k* nodes are left, so the k-way partitioning routine can execute over a smaller version of the original graph. It is possible that more than *k* nodes are left in the final condensed graph, since the condensation process stops if the reduction in size of the graph in successive condensation phases becomes too small. Also, since the condensation process favors the merging of nodes connected by heavier edges, with higher weight values, the resulting condensed graph can be seen as the initial k-way partitioning of the original graph.

Finally, after obtaining the partitions of the condensed graph, the inverse of the condensation process is performed. By going back through the condensation levels, the merged nodes become separated and are assigned to the same partition. Since it is now possible that a new better assignment of nodes to partitions exists, a refinement algorithm is executed in each condensation level, up until the original graph is reached.

3.3.5. Flow-Based Clustering

Another way of clustering networks is through the use of flow based methods. In (Flake et al. 2000), the authors propose an algorithm capable of crawling the web and clustering a graph of indexed web pages based on the *s-t maximum flow* problem. The s-t maximum flow problem can be defined as finding the maximum flow that can be routed from one node (*s*) to another (*t*) and can be solved with Ford and Fulkerson “max flow-min cut” theorem (Kleinberg & Tardos 2005).

In (Yang et al. 2007), the authors propose an algorithm called FEC, that can discover hidden communities in signed and weighted network graphs or digraphs. Although it isn’t explicitly flow based, it follows the intrinsic characteristics of flow inside social networks.

For partitionable signed graphs, the clustering problem could be solved simply by cutting negative links between nodes. But in a scenario where edges are not only signed but also weighted, using such a method would be as good as ignoring the edge weight values (see figure 20).

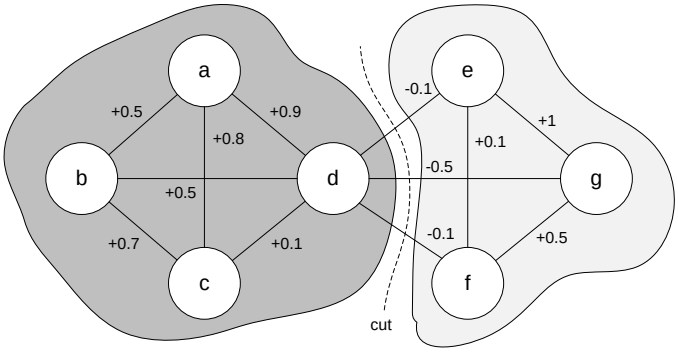

Figure 20 - Simple cut in a signed graph. Clusters with positive edges inside, connected by negative edges. The cut does not regard the weight values.

FEC contains two main phases: the FC phase where a random-walk agent computes the transition probability vectors from the adjacency matrix, and the EC phase that applies a cutoff criterion over the resulting matrix from the FC phase.

The random-walk agent starts from one node and proceeds with its walk for a number of steps, trying to reach the sink node. At each step, the decision on where to go next is based on the transition probability distribution that depends on the node degree of connectivity. Since the flow inside communities is greater than between communities, there is a better aggregated probability for the agent to remain inside the same community after a certain number of steps.

Following this general definition of community, the random walk agent generates an adjacency matrix (see figure 21) with a column containing the resulting probability of the agent reaching a specified sink node, starting from any other node in the network.

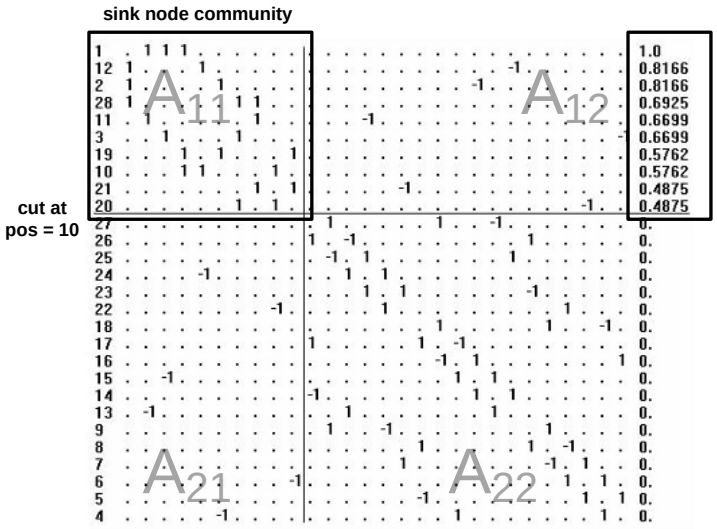

Figure 21 – A transformed adjacency matrix after the random walk algorithm (Yang et al. 2007), ordered by the probability of the random walk agent reaching the sink node 1 from any other node.

The reordered matrix not only has a diagonal structure, but it also denotes an existent (but hidden) community in the network (top-left part). To extract these communities, a new criterion called signed cut is applied during the EC phase. The motivation behind the proposal of a new cut criterion is that the minimum, average and normalized cut do not provide acceptable results in many scenarios, including those of signed networks (signs are ignored).

According to the signed cut, there should be a small number of negative edges, and a high number of positive edges, both within communities. The opposite stands for edges between communities, where negative signs should be more abundant than positive ones.

The signed cut acts over a matrix, A, resulting from the FC phase, and extracts the sink node community (represented by the block matrix A_{11} in figure 21) by satisfying the following two conditions:

$$\underset{1\le i\le pos}{\forall}\left(\sum_{1\le j\le pos}A_{ij}\ge\sum_{pos<j\le n}A_{ij}\right)\wedge\underset{pos<i\le n}{\forall}\left(\sum_{1\le j\le pos}A_{ij}\le\sum_{pos<j\le n}A_{ij}\right) \quad (13)$$

Where *pos* is the position of the signed cut in the matrix A. Notice that the sums in both conditions are performed over the block matrices A_{11}, A_{12}, A_{21} and A_{22} respectively.

After the application of the signed cut, FEC continues to recursively split the network, as long as the clustering error before the cut is higher than the clustering error after the cut. The proposed error measure works effectively for signed networks since it measures the negative links inside communities and positive links between communities, against the overall network links. However, this error measure reveals to be ineffective for unsigned networks, since they don't have any negative links.

The FEC algorithm is fast, with a complexity of $O(KI(n+m))$, where K is the number of communities found, I the number of steps, n the number of nodes, and m the number of edges.

3.4. Resume Mining of Communities

Although community mining algorithms are able to successfully discover communities in social networks, they usually don't provide significant information over their nature. Resume mining of communities tackles this problem and focuses on finding community specific features, the causes for their formation and how they evolve through time.

Such an analysis has come to leverage community mining to a world of immense practical scenarios. In (Duan et al. 2009), the authors explore dynamic and weighted digraph representations to discover community structure and perform the detection of significant changes in community structure, as the network evolves (change-point detection).

The proposed algorithm, called *Stream-Group*, contains three phases: the first discovers clusters using a random walk with restart technique. The second merges the resulting clusters, maximizing their modularity values. Finally, the third phase applies a similarity measure between graph partitions from the current and previous time slices, so that changes over time can be detected.

The Stream-Group algorithm allows the incremental update of the network graph representation, and although it is mainly a community mining algorithm future research may leverage change-point detection, so further analysis over the evolution (and respective causes) of the network can be performed.

The resume mining of communities approach in (Wu et al. 2007) not only includes a method to describe communities using node attributes and network structure, but also to determine if two communities from different graphs are successive by core members, i.e. if one community is the predecessor of another. Core members are important to the classification of communities and are actors with significant centrality values.

Community evolution and history analysis can provide important knowledge and monitoring information. One of the possible analyses relies on community succession, meaning that different communities that exist in different time spans might actually be related by succession. To obtain these relations, a historical centrality is calculated for core nodes of a community. This historical centrality uses multiple community snapshots, taken through time, to finally evaluate if different communities are successive.

For a given community, it is expected that every node has some similarity. By exploring and analyzing similarities in the attribute values of nodes inside a community, it is possible to classify that very same community. To do this, a new measure of commonness is presented in (Wu et al. 2007). Commonness has a great impact over the community's' ability to remain stable over time, and since core members are responsible for the creation of the community, their compatibility (or consistence) has a lot of impact on the community. For core members to be considered consistent upon an attribute their ultimate cause has to converge or the community is more likely to be torn apart. Additionally, community members are consistent on an attribute if core members are consistent, and if the difference between core members and the others is not significant.

However, evaluating the commonness of community nodes is not enough to discriminate one community from another. To improve characterization and distinction, the algorithm calculates the distribution of attribute values along community nodes. For example, in figure 22, the mean age value of members for both communities is the same, but their distribution is clearly different. In one community, core members are older than peripheral members while the opposite happens in the second community.

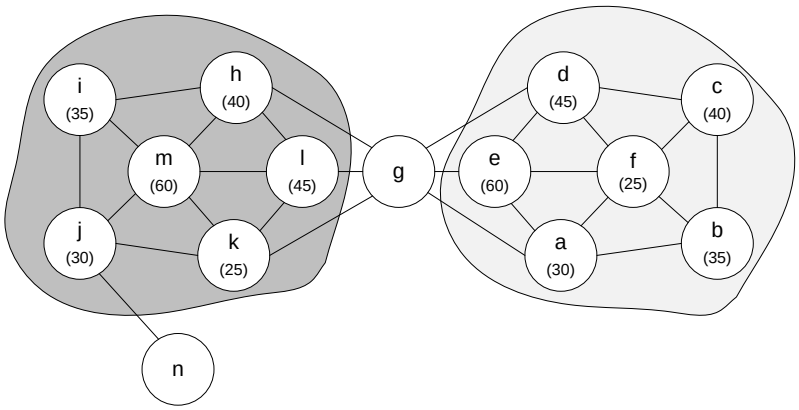

Figure 22 - Resume mining of communities: age attribute commonness.

3.5. The SocioNet Ontology (SNO)

The classification of the previously presented SNA algorithms is in fact a complex task. They involve different methods, require different parameters, and work under different scenarios.

Although the state of the art description is in an easy human-interpretable fashion, it is not machine-readable. Also, the inherent poly-classification of the SNA algorithms is hard to picture in a wide scale.

To tackle the inherent complexity associated with the wide variety of SNA algorithms and methods, an OWL DL, SHOIN-expressive, ontology has been modeled in the context of this thesis: the SocioNet Ontology (SNO)[2]. The SNO exploits the SNA concepts modeled in the SemSNA ontology (Erétéo et al. 2009). This last ontology describes SNA static concepts such as Betweenness and Degree, but is not meant to characterize nor classify SNA algorithms.

The SNO is a semantically-rich ontology that captures the necessary, and necessary and sufficient conditions of different classes of SNA algorithms (e.g. hierarchical algorithms) in a way that each one of them can be poly-classified according to their input and output. This ontology is partially serialized in table 6, and its partial asserted and inferred hierarchy is graphically depicted in figure 23.

Table 6 - Manchester Syntax serialization for the partial SocioNet ontology.

Class: **SCANAlgorithm**
Annotations:
rdfs:comment "Xu, Yuruk, Feng and Schweiger's Structural Clustering Algorithm for Networks"@en
SubClassOf:
NamedAlgorithm,
hasOutput some Clustering,
hasOutput some Dendrogram,
hasOutput some StructuralConnectivity,
hasOutput some StructuralReachability,

[2] This ontology was initially proposed in the paper: "Luz, N., Anacleto, R., Silva, N. and Almeida, A. 2010. A Survey of Social Network Analysis (Submitted)".

hasOutput some StructuralSimilarity

Class: **StructureAlgorithm**
Annotations:
rdfs:comment "An algorithm that outputs a StructuralIndice"@en
EquivalentTo:
SNAAlgorithm
and (hasOutput some StructuralIndice)

Class: **HierarchicalClusteringAlgorithm**
Annotations:
rdfs:comment "A clustering algorithm that is innately hierarchical; its process produces a dendrogram"@en
EquivalentTo:
ClusteringAlgorithm
and (hasOutput some Dendrogram)

Notice that all SNA algorithms are under the NamedAlgorithm class. It is only after inference that SNA algorithms are automatically poly-classified according to their input and output, resulting in the inferred ontology. For example, SCANAlgorithm is simultaneously (poly-) classified as StructureAlgorithm (because it *hasOutput* StructuralConnectivity, which is a sub-class of StructuralIndice) and HierarchicalClusteringAlgorithm (because it *hasOutput* Dendrogram).

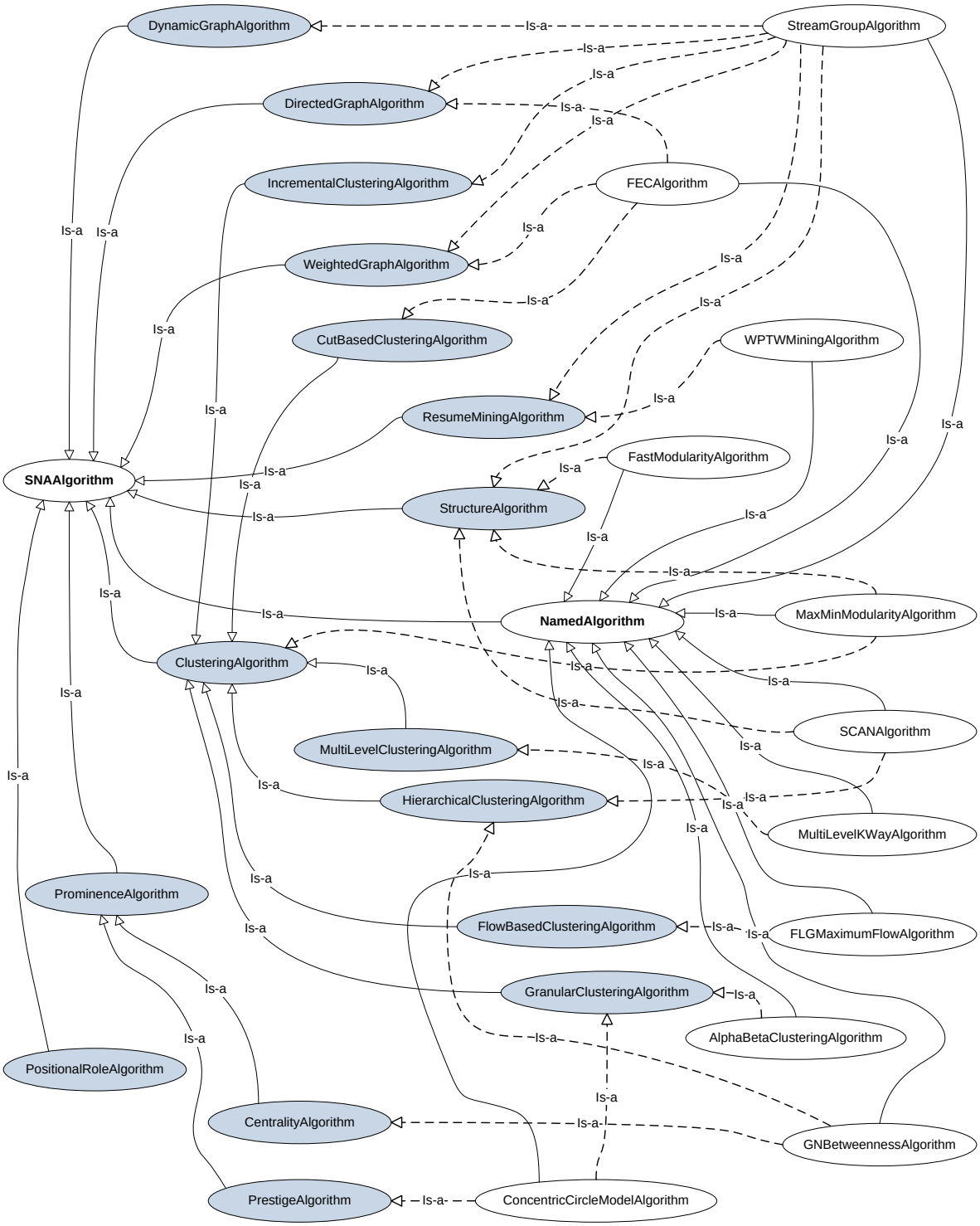

Figure 23 - Inferred partial SNA ontology. Dashed arrows represent inferred axioms.

The SocioNet ontology not only provides a classification of SNA algorithms, but also the means to express SNA concepts like graphs, measures and clusters that represent the input or output of an SNA algorithm. Thus, it is possible to apply multiple SNA algorithms to a social network, storing their results and associated execution context. Figure 24 contains the main SNA concepts in SNO.

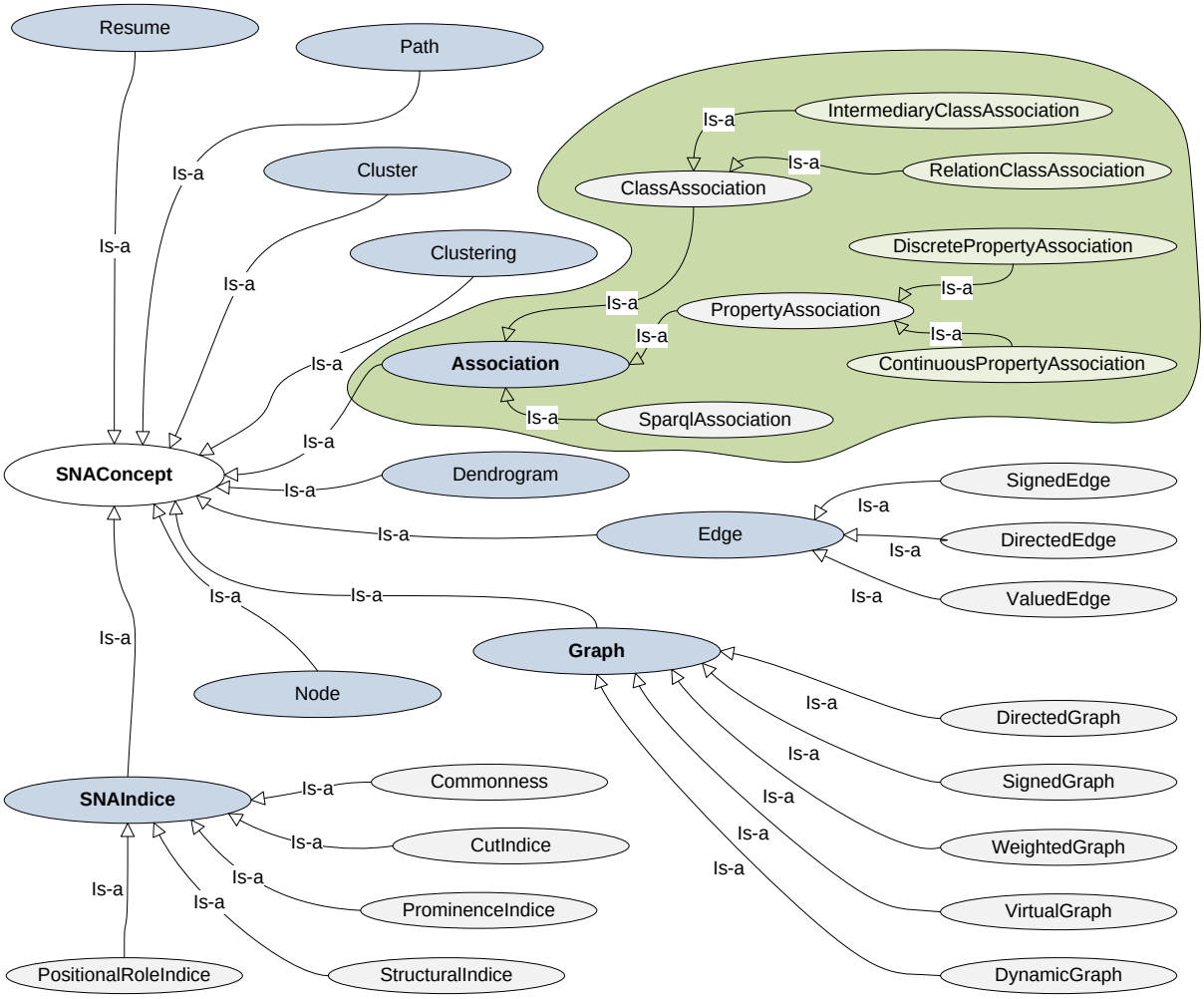

Figure 24 – Partial SNAConcept class hierarchy.

Notice that an instance of SNAAlgorithm is interpreted as an execution of the algorithm at some point in time. Some of the SNAConcepts in SNO (depicted in figure 24) are:

- **Graph**: used to annotate important information regarding the social network graph;
- **Node**: nodes of the graph (e.g. TPO Person instances);
- **Edge**: an edge connecting two nodes;
- **Association**: describes how a VirtualGraph is formed, i.e. the original meaning of the association between two actors, hidden behind the edges of the graph;
- **SNAIndice**: an index or measure resulting from the execution of a SNA algorithm;
- **Path**: used to annotate relevant paths between actors;
- **Clustering**: defines a clustering resulting from the execution of a community mining algorithm;
- **Dendrogram**: a dendrogram resulting from a hierarchical community mining algorithm;
- **Cluster**: a cluster which belongs to a Clustering;
- **Resume**: an output resulting from a resume mining of communities algorithm.

A SNO VirtualGraph is built from mining other triple graphs. The mining process extracts associations between actors according to a small set of modeling patterns, named Associations. Associations define how two actors possibly are related, and are required for the triple graph to be mined accordingly. Figure 25 depicts multiple SNOConcepts related to the Graph and VirtualGraph classes, including the Association class.

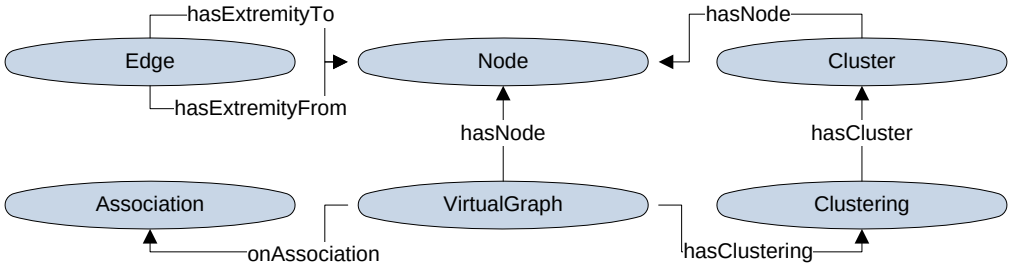

Figure 25 - SNO VirtualGraph, Clustering, Cluster, Node, Edge and Association classes.

A detailed description of the different types of associations as well as some usage examples are provided further in chapter 4.

The complete and ever-evolving ontology is serialized in RDF/XML and available at http://www.gecad.isep.ipp.pt:9090/toursplan/sna.owl.

Chapter 4
Social Network Analysis over Triple Datasets

The multiple heterogeneous and semantically enriched data sources emerging in the Semantic Web have led to the existence of significant amounts of social data modeled and interpreted in different ways. Also, triple graphs may contain social interaction data (linking two or more actors) that can only be efficiently extracted after the analysis of multiple paths, instead of the usual direct edge analysis. Although a SPARQL approach to SNA is proposed in (Erétéo et al. 2009), most existing SNA algorithms are not optimized to run directly over (or use) the emerging Semantic Web triple data representations and technologies like RDF, OWL, SWRL or SPARQL.

This chapter presents a process for ontology-based social network data normalization and SNA algorithm execution that bridges the gap between the Semantic Web data models and current SNA implementations, independently of:

- The knowledge domain (e.g. tourism, cinema);
- The domain application, i.e. independently of data or the data model;
- The SNA algorithms available in the SNA framework;
- The desired SNA measures, as requested by the application manager.

The proposed approach is depicted in figure 26.

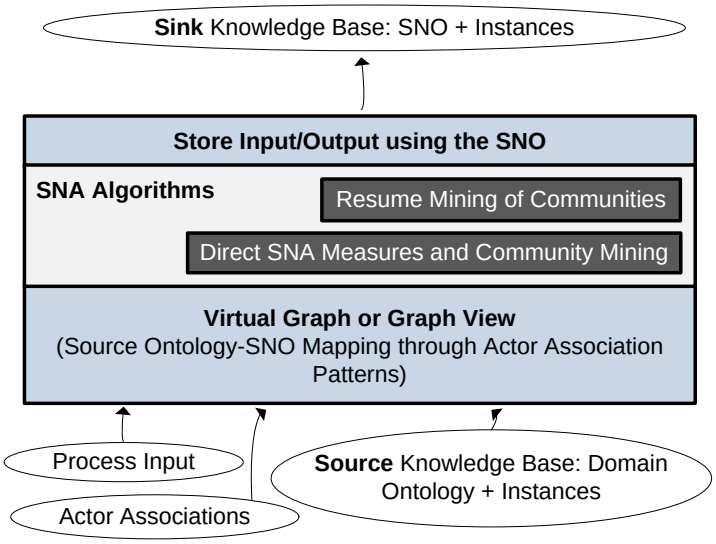

Figure 26 – Semantic SNA stack.

For that, one has to provide the following:

- A common representation model of the domain (e.g. social tourism). In the context of the Semantic Web, this role will be played by a domain ontology;
- A normalization process providing an abstraction over the specificities of the data sources and their models, giving rise to a Virtual Graph;
- A configurable SNA process for the execution of a sequence of SNA algorithms, responsible for calculating measures, performing community mining and/or resume mining of communities. After the SNA algorithm execution, results can be stored in a new triple graph (e.g. modeled by SNO).

These principles are addressed in the following sections. While other domains could be adopted, the tourism domain has been chosen for obvious reasons.

4.1. The Toursplan Ontology (TPO)

To begin the SNA of triple data, a Virtual Graph or view must be created over the knowledge base (or graph of triples). A knowledge base consists in a triple dataset containing both the ontology model (metadata) and its instances (data). This is useful in situations where meaningful social interactions or relations exist in the triple graph, but are modeled in a complex or indirect way.

To push Toursplan into the Semantic Web, a semantically-rich ontology based on the current relational database was built and optimized. From the previously presented state of the art in chapter 2, not only it is possible to build a GUI (Graphical User Interface) to interact with tourism social network actors, but also a model capable of capturing data generated in a tourism online social network environment.

Currently, the TPO partially describes three knowledge domains:

- The tourism domain;
- The social interaction domain;
- The actor/user profile domain, extended from the FOAF (Friend Of A Friend) ontology.

Although each one of these knowledge domains can be described by its own ontology model, only the currently necessary concepts from all of the three domains were modeled leading to a single ontology: the TPO. As the ontology evolves, it might be reasonable to divide the TPO into two or more separate models.

Multiple questions emerged when building the Toursplan ontology. One of them is related to what is known in relational databases as the associate or linking table, used to represent many-to-many relationships. Considering the Semantic Web context, the TPO had to be developed using a Semantic Web ontology language, namely RDF-S or OWL. Since the RDF-S and OWL languages only support binary relations, it is not possible to model linking tables (e.g. containing valued and temporal relations) in a straightforward way. The solution can be achieved through multiple methods:

- A model with linking classes, each one representing a different relation;
- The reification of triples, so that each triple is a statement, which can be used as the subject of another triple;
- Named graphs, since they provide the means to define multiple named graphs which act as special containers for triples (quads). This not only allows the triple to be identified by the named graph, but also the existence of other triples (e.g. the triple that defines the weight value) inside the named graph.

To model valued and temporal relations, linking classes were used. This method was chosen since named graphs revealed to be adequate for containing relatively larger sets of triples, than those produced by a single valued or temporal relation. Also, reification was avoided since reified triples obfuscate the semantics of the relation[3].

Another question relies on the modeling paradigm, i.e. Description Logics (DL) versus property-centric. While the (RDF-S) property-centric paradigm forces properties to have a certain domain and range, OWL DL follows a different modeling approach where classes can be described through axioms.

In the TPO, the *child* object property can be applied to different domain and range classes (e.g. Place and PoiCategory) which do not share a super class adequate for representing the property domain or range. To model such a scenario in a property-centric approach, multiple child properties would have to exist (e.g. a *childPlace* for Place and a *childPoiCategory* for PoiCategory).

[3] http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/

OWL DL avoids the creation of multiple similar properties with different domain and/or range, since the property is not defined by a domain or range. Instead, classes are defined by axioms, which describe the properties and corresponding property ranges of the class.

As a consequence, OWL DL has been adopted as the TPO representation language and modeling paradigm. The resulting ontology has SHIQ(D) expressivity, and figure 27 contains its partial class hierarchy.

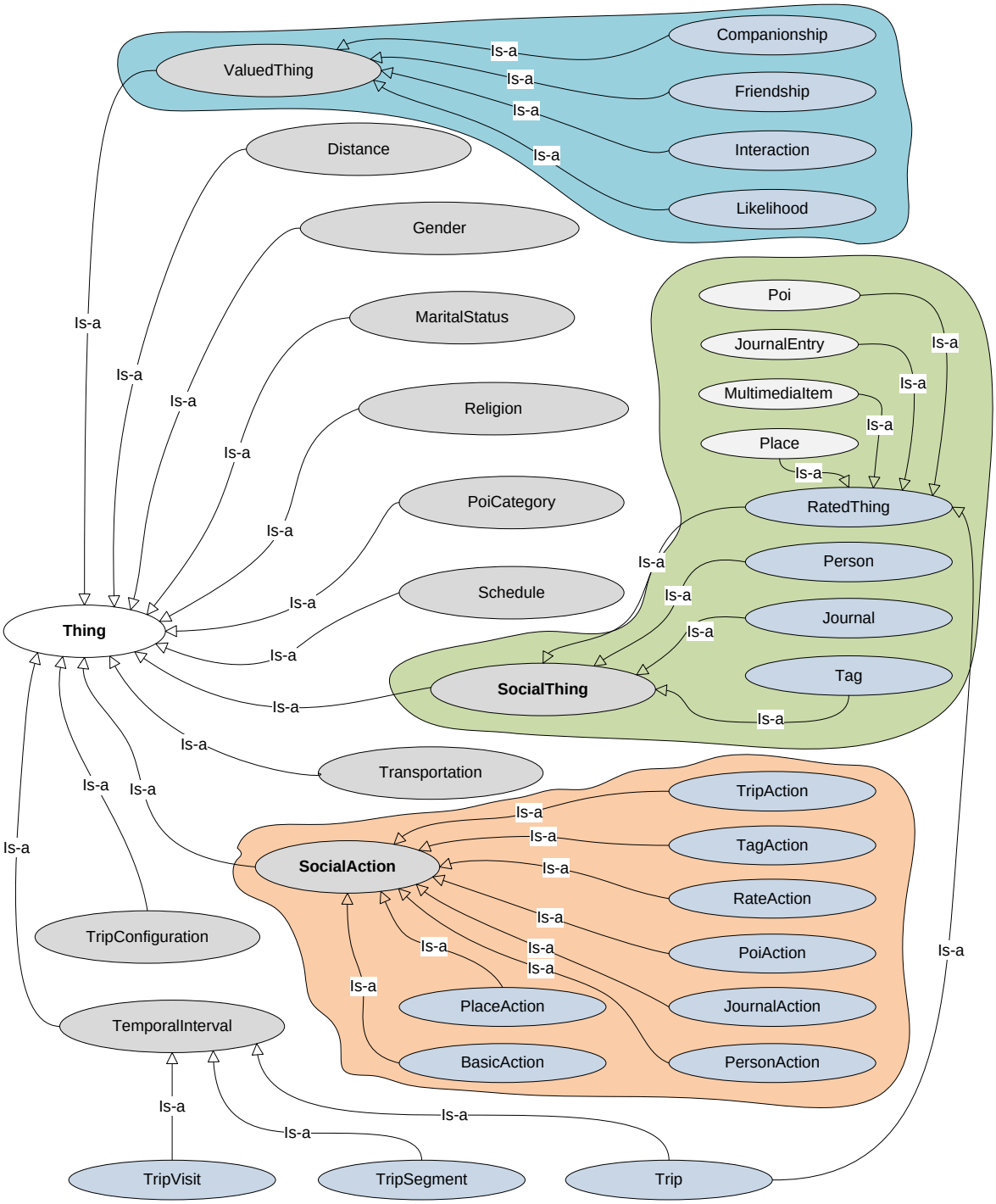

Figure 27 – Partial class hierarchy of the TPO.

Although, excluding the SPARQL (Simple Protocol and RDF Query Language) endpoint, the TPO is only available for internal use, it extends the FOAF ontology as it is currently one of the most popular ontologies in the Web for social relations and actor/user profile representation.

4.1.1. Social Actors (Person)

The actor profile in the TPO is modeled according to the current Toursplan recommendation and planning needs. In figure 27, some classes that represent actor attributes can be identified, such as Gender, MaritalStatus and Religion. This way, a set of editable instances can be used as the range of the attribute. Table 7 contains the Manchester syntax serialization for the Person class, describing its object and datatype properties.

Table 7 - Manchester Syntax serialization for the Person class.

Class: **Person**
SubClassOf:
foaf:Person,
SocialThing,
account some Account,
companionship some Companionship,
friendship some Friendship,
gender some Gender,
location some Place,
maritalStatus some MaritalStatus,
performed some SocialAction,
religion some Religion,
address some Literal,
creativity max 1 double,
hearingHandicap max 1 double,
liveliness max 1 double,
mobilityHandicap max 1 double,
outdoorness max 1 double,
perfectionism max 1 double,
sightHandicap max 1 double

Other datatype properties such as *foaf:mbox*, *foaf:firstName* and *foaf:birthday* come from the FOAF ontology.

The TPO stores two types of social relationships, one being Companionship and the other Friendship. While the friend list is built directly by the actors through the application GUI, companionship relations between actors are defined by the application engine, when two or more actors choose to travel together. Since this extra relationship is deemed to represent actual social interaction in the real world, it can be used, up to some extent, to either check the veracity of the actors' friend list or strengthen the friendship relation in filtering algorithms. Figure 28 shows both classes Friendship and Companionship as being subclasses of ValuedThing. Both relationship types are valued (or weighted), with Companionship being undirected and Friendship being directed.

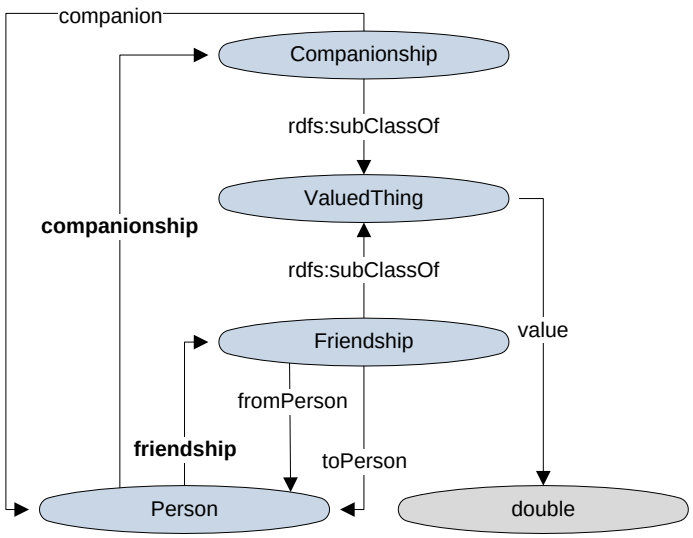

Figure 28 - Actor relationships: Friendship and Companionship.

4.1.2. Social Things and Actions

Social networks are built from social interactions between different actors (e.g. friendship invitations and sending of messages). In that sense, the TPO models these interactions through the SocialAction class. Every SocialAction is performed by a Person and has an associated time, which represents the moment the action was performed. Also, every SocialAction (e.g. LikeAction and BefriendAction) involves at least one SocialThing (e.g. Person, Tag and Poi). The ontology includes general concepts that can be applied to any online social network (e.g. PersonAction and RateAction), and others that are specific to the tourism domain (e.g. JournalAction and PlaceAction). Figure 29 depicts the SocialAction class in the TPO.

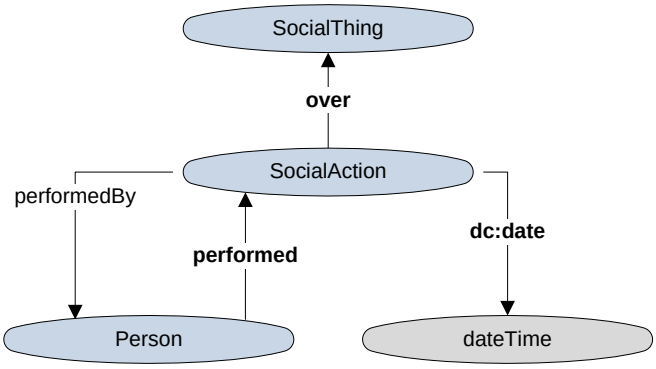

Figure 29 - Social actions in the Toursplan Ontology.

The different actions inside the Toursplan ontology are (see figure 30):

- **SocialAction**: the root action class. Every social action is a subclass of SocialAction;
- **RateAction**: the actor rates a SocialThing instance;
- **JournalAction**: an action over a Journal instance;

- **TagAction**: an action over a Tag instance;
- **PoiAction**: an action over a POI instance;
- **TripAction**: an action over a Trip instance;
- **PersonAction**: an action over another Person (e.g. sending a message or inviting to friend list);
- **PlaceAction**: an action over a Place instance;
- **BasicAction**: basic actions, i.e. add, remove and update. BasicAction instances can also be instances of other action classes like JournalAction. An action instance with both types AddAction and JournalAction, represents the creation of a new Journal instance.

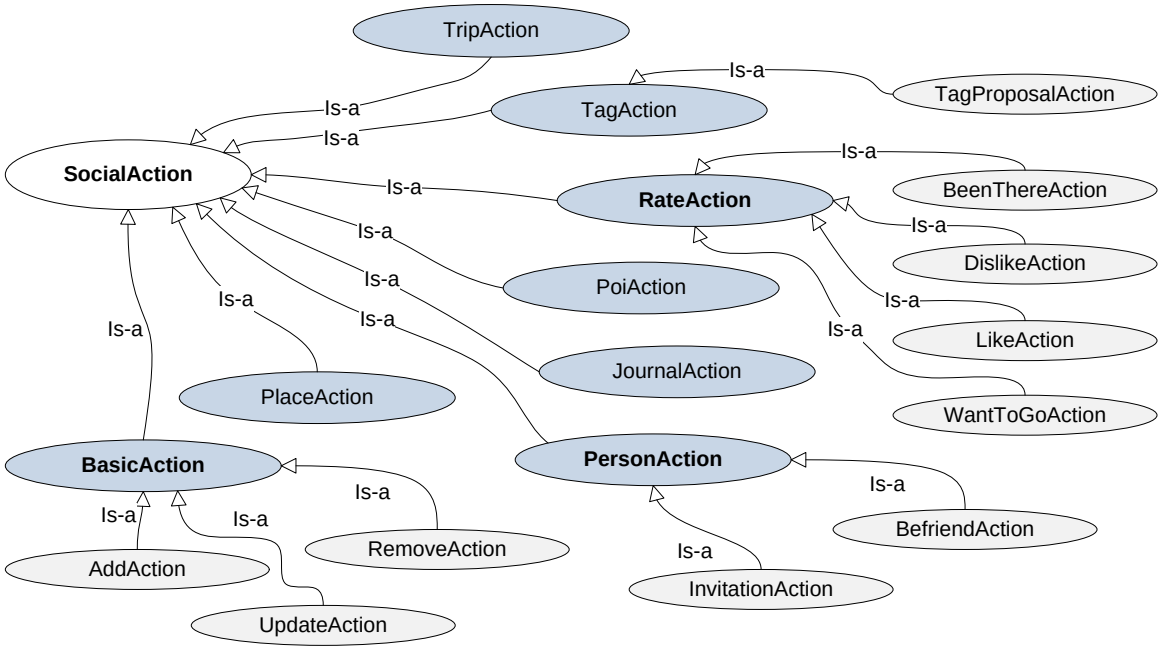

Figure 30 - Toursplan ontology: social actions.

4.1.3. Points Of Interest

POI is an important concept in the TPO, related to multiple instances of different classes. For instance, a POI has a category which is an instance of PoiCategory. Although these categories (e.g. Hotel, Restaurant, Monument) can be modeled as classes, their definition as instances seemed a more straightforward solution than their definition as subclasses of POI. Also, the current Toursplan recommendation system feeds on a bi-dimensional likelihood matrix with actors and POI classes as dimensions. To represent the matrix using the TPO, instances of POI categories are needed so instances of Likelihood can link instances of PoiCategory to instances of Person. In case POI categories were modeled as subclasses of Poi, these subclasses would play the role of instances of PoiCategory in the likelihood matrix. This turns the ontology undecidable and therefore incompatible with OWL DL (becomes OWL Full). Figure 31 contains both classes PoiCategory and Poi linked through the transitive property *category*.

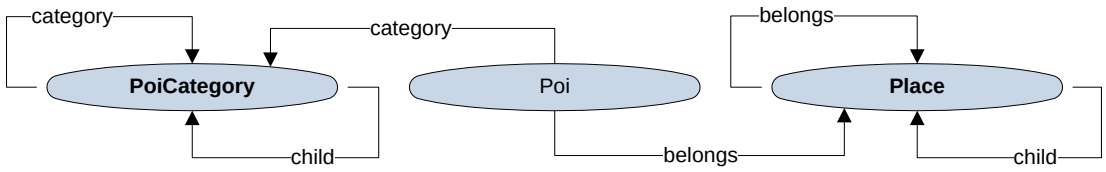

Figure 31 – PoiCategory and Place properties: category, belongs and child.

A POI is also linked to a Place through the *belongs* transitive property. With a reasoning mechanism applied to property *belongs*, all the POIs in a Continent can be obtained, even if they are only explicitly linked to a Country inside the Continent.

From the granularity of the classification of places observed in the analyzed tourism/travel online social networks, the TPO has been modeled to classify places as being a Village, Region, City, Country or Continent (see figure 32).

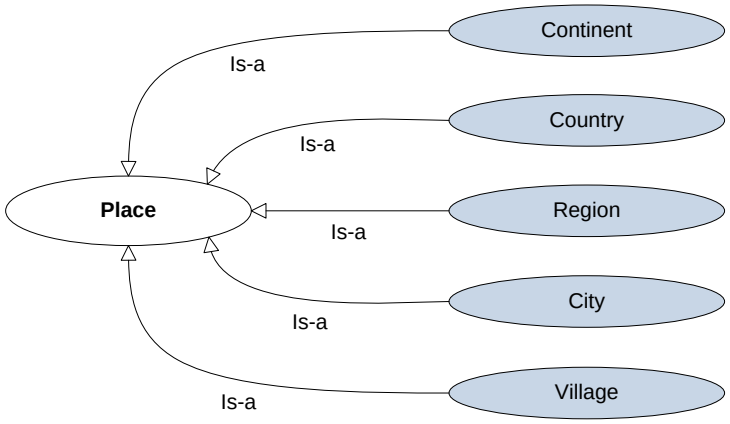

Figure 32 – Toursplan ontology: Place subclass hierarchy.

4.1.4. Trips

Trips have been modeled according to the planning system needs of the Toursplan application. Each Trip contains one or more TripSegments. TripSegments allow the actor to specify which periods of time the planning system is going to fill with visits to POIs. All the planning configuration properties fed to the planning system must be set in a TripConfiguration instance that can be associated with the Trip (for a global configuration) and/or the TripSegment (for a specific configuration of the segment). The planning procedure is then responsible for creating TripVisit instances (visits to POI) associated with the Trips' TripSegments (see figure 33).

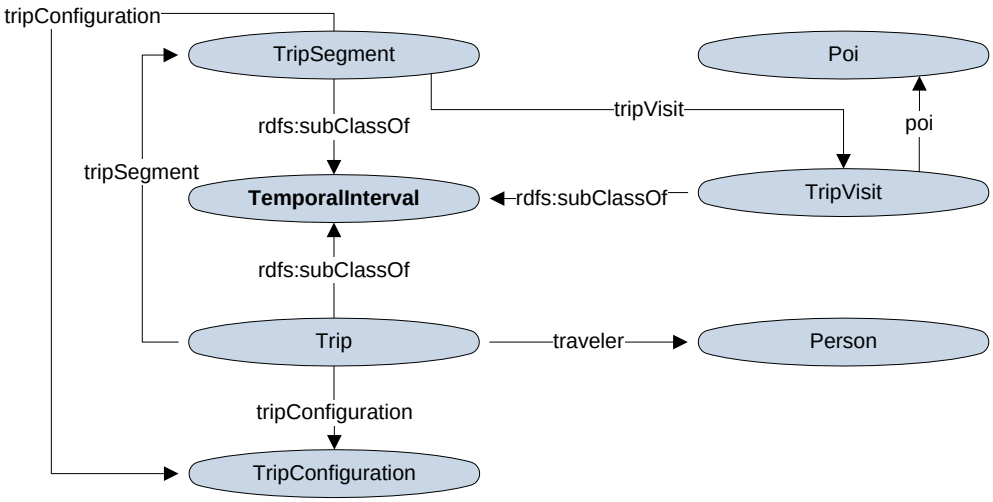

Figure 33 - Trip, TripSegment and TripVisit object properties.

The TripConfiguration class contains properties like *dinnerPoi*, *lunchPoi*, *morningTransportation*, *walkingLimit*, *morningFeetMaxDistance* and *moneyLimit*. These values are used as input parameters by the planning algorithms.

4.2. Virtual Graph

The TPO is an example where relations and interactions are not modeled in a straightforward way, i.e. there are cases where relations and interactions are modeled as classes, and other cases where they can be found only after traversing the triple graph for more than one step. In that sense, a small set of modeling patterns is identified so that associations between actors can be obtained from them. This way, the possible (types of) associations between actors can be specified when creating the Virtual Graph.

The Virtual Graph (or Graph View) represents an abstraction over a source triple graph, reflecting links between two nodes with a specific type (e.g. Person), which may not be directly linked in the source triple graph. These links are extracted using a set of different types of associations, i.e. the Discrete Property Association, the Continuous Property Association, the Relation or Linking Class Association, and the Intermediate Class Association, picked from identified modeling patterns. This means that several SNA algorithms can be executed over a single unified graph formed from multiple and different actor related data.

The edges of a Virtual Graph denote the existence or absence (or strength for valued edges) of an association between two nodes. In valued edges, values usually range from 0 (not associated) to 1 (totally associated). This way, most SNA algorithms can be fed with the resulting graph and used to measure or cluster the network.

Virtual Graph instances persist in the sink graph, which is always built using the SNO. The source graph, however, is not restricted to a specific model. In the Toursplan scenario, the source graph is built from the TPO (see figure 34).

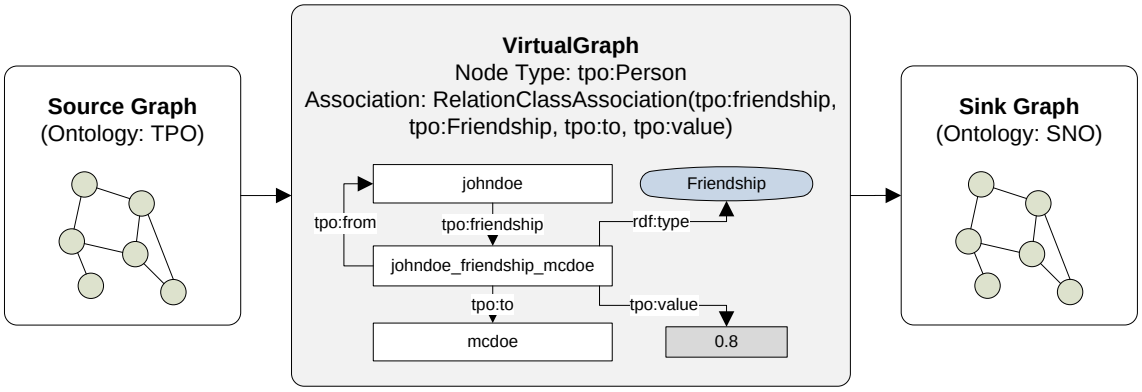

Figure 34 - Virtual Graph usage example.

Figure 34 depicts an example where a Virtual Graph is created over a TPO source graph. The association between johndoe and mcdoe is found through a Relation Class Association and represented in the sink graph with instances of sno:Node and sno:ValuedEdge with the *sno:hasValue* property value equal to 0.8. The currently identified and implemented associations are represented in figure 24, inside the green area.

4.2.1. Finding Associations

Properties found in the source graph ontology need to be mapped differently according to their characteristics (e.g. range and type). Although it is possible to consider all object properties as being discrete, i.e. their range is a discrete set of objects, it is not the case for data type properties since they can be either discrete (e.g. name) or continuous (e.g. age).

In order to represent relevant data (e.g. a weight or value) about subject-predicate-object triples, TPO contains properties modeled as classes (e.g. Friendship, Companionship). Besides situations where modeling issues demand the existence of this kind of classes, denominated as relationship classes, there are those where a deeper analysis of the graph would be useful. The TPO social actions represent such a case. Although they don't represent social relationships, clustering from them might be useful to detect patterns among the actions of the social network actors.

Figure 35 depicts instances of multiple TPO classes, where four different association patterns can be identified.

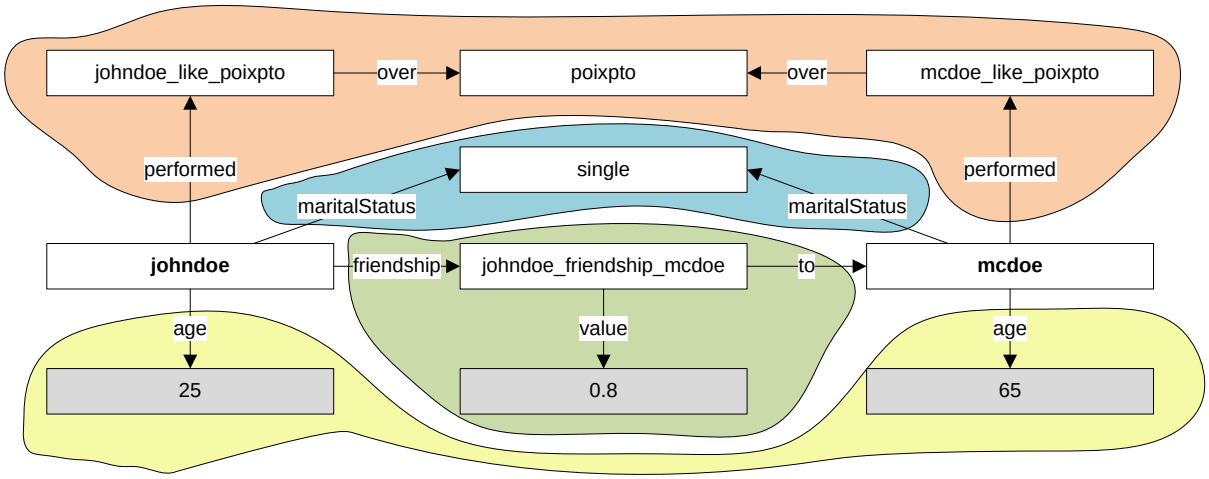

Figure 35 - An example of four different actor association patterns. In orange: an Intermediate Class Association. In blue: a Discrete Property Association. In green: a Relation Class Association. In yellow: a Continuous Property Association.

From figure 35, the simplest association found between actors johndoe and mcdoe would be through the discrete object property *maritalStatus* (in blue) or continuous data type property *age* (in yellow). On the other hand, the probably most useful association in SNA would be through the johndoe_friendship_mcdoe instance (in green), which is not as easy to extract as the previous two associations. The fourth and most complex association in figure 35 is through both actors johndoe and mcdoe actions (in orange).

4.2.2. Property Associations

There are two types of property associations: discrete and continuous. Discrete property associations are obtained by comparing the discrete property values of two actors. Let *p(a)* be the value set of property *p* for instance *a*, then the discrete association between the actor instances *a* and *b* over property *p* is:

$$discAssoc(a,b,p) = \begin{cases} 1 : |p(a) \cap p(b)| > 0 \\ 0 \ \ otherwise \end{cases} \quad (14)$$

However, there might be a situation where a cardinality-based approach is needed. In such a scenario, the discrete association between instance *a* and *b*, over property *p* is:

$$discAssocCard(a,b,p) = \frac{|p(a) \cap p(b)|}{|p(a) \cup p(b)|} \quad (15)$$

Table 8 shows the results of a discrete property association over property *maritalStatus* for actors johndoe and mcdoe, depicted in figure 35. In this example, either using the cardinality-based or the non-cardinality-based approach, association values are either 0 or 1. This happens because it is assumed that each Person can only have one *maritalStatus*.

Table 8 - Association matrix from the *maritalStatus* property in figure 35.

maritalStatus association	***Johndoe***	***mcdoe***
Johndoe	-	1
Mcdoe	1	-

Continuous properties (e.g. *age*) (Wu et al. 2007) have a range of values that can be represented by an interval. Let $p_u(a)$ be the unique value of property p for instance a, and C be the actor class, then the continuous association between actor instances a and b of type C over property p is:

$$contAssoc(a,b,p,C) = 1 - \frac{|p_u(a) - p_u(b)|}{\max P(p,C) - \min P(p,C)} \quad (16)$$

where

$$\max P(p,C) = \max_{\forall x: C \in type(x)} (p_u(x)) \quad (17)$$

and

$$\min P(p,C) = \min_{\forall x: C \in type(x)} (p_u(x)) \quad (18)$$

Table 9 contains the continuous association values of the *age* property for the actors in figure 35.

Table 9 - Association matrix from the *age* property in figure 35. Maximum and minimum property values were considered to be 70 and 15, respectively.

age association	***johndoe***	***Mcdoe***
Johndoe	-	0.27
Mcdoe	0.27	-

Notice that, although both examples contain symmetric matrices, there might be cases where they aren't (e.g. *foaf:knows*).

4.2.3. Relation Class Associations

In some cases, there might be a need to obtain associations not only through properties, but also through instances of relation classes. In figure 35, a directed valued *friendship* relation is depicted through an instance of the Friendship class, where johndoe considers mcdoe a friend. To extract such an association, one has to consider the characteristics of the relation, i.e. if the relation is valued or unvalued, directed or undirected. In that sense, the following input parameters are needed:

- Relation class (e.g. Friendship);
- First level property, connecting the first actor to the relation class instance (e.g. *friendship*);
- Second level property, connecting the relation class instance to the second actor (e.g. *to*);
- Value property, if the relation is valued (e.g. *value*).

This way the association graph is built using the value of the value property if a valued relation class instance exists between two actors, or using the values 1 (association exists) or 0 (association does not exist) if the relation class instance is not valued.

For a first level property *p1*, a second level property *p2*, a relation class C_r, and a value property *v*, the relation class association is given by:

$$relClassAssoc(a,b,p1,C_f,p2,v) = \underset{\forall i \in p1(a):b \in p2(i) \wedge C_f \in type(i)}{avg}(relClassAssocVal(i,v)) \quad (19)$$

where

$$relClassAssocVal(r,v) = \begin{cases} v_u(r) : |v(r)| > 0 \\ 1 : |v(r)| = 0 \end{cases} \quad (20)$$

The cardinality-based relation class association is given by:

$$relClassAssocCard(a,b,p1,C_f,p2,v) = \frac{\sum_{i \in p1(a):b \in p2(i) \wedge C_f \in type(i)} relClassAssocVal(i,v)}{\sum_{i \in p1(a):C_f \in type(i)} relClassAssocVal(i,v)} \quad (21)$$

4.2.4. Intermediate Class Associations

Intermediate class associations are different from relation class associations since their second level property can lead to a non actor instance, i.e. there must be two similar paths from two actors *a* and *b*, to an instance of type C_o. Such a scenario is depicted in figure 35 (in orange). Both actors johndoe and mcdoe like the Poi poixpto, and an analysis of the intermediate instances johndoe_like_poixpto and mcdoe_like_poixpto can link both actors in the association graph.

The intermediate class association is given by the following equation:

$$inClassAssoc(a,b,p1,C_o,p2) = \begin{cases} 1 : \sum_{\substack{r1 \in p1(a):C_o \in type(r1) \\ r2 \in p1(b):C_o \in type(r2)}} |p2(r1) \cap p2(r2)| > 0 \\ 0 \ \ otherwise \end{cases} \quad (22)$$

The cardinality-based intermediary class association is given by:

$$inClassAssocCard(a,b,p1,C_o,p2) = \frac{\sum_{\substack{r1\in p1(a):C_o\in type(r1)\\ r2\in p1(b):C_o\in type(r2)}} \left| p2(r1)\cap p2(r2)\right|}{\sum_{\substack{r1\in p1(a):C_o\in type(r1)\\ r2\in p1(b):C_o\in type(r2)}} \left| p2(r1)\cup p2(r2)\right|} \quad (23)$$

4.2.5. Custom Association

In some situations, the previously identified associations may not be enough to extract the necessary knowledge from the triple graph. To provide a mechanism able to work under such a scenario, a new kind of flexible association is needed: a SPARQL association. Still, the SPARQL standard is not powerful enough to easily extract cardinality-based values or continuous property associations. In most cases the resulting association values will be either 0 or 1, or obtained from a value property in the source graph.

However, the discrete property association on *maritalStatus* could be defined through a SPARQL association using the query in table 10.

Table 10 - SPARQL association query on property *maritalStatus*.

```
PREFIX tpo: <http://gecad.isep.ipp.pt/toursplan#>
ASK
{
  ?a tpo:maritalStatus ?obj .
  ?b tpo:maritalStatus ?obj
}
```

The character sequences "*?a*" and "*?b*" represent two actors in the source graph.

4.2.6. Multiple Association Values

The Virtual Graph can be built from one or more associations. This way, the resulting association graph can reflect actor associations according to the multiple attributes or relationships that define the actors. There are multiple approaches to merge the association values between two actors:

- Mean association value;
- Weighted mean association value (weights specified as an association input parameter);
- Cardinality-based mean association value (weights extracted from the cardinality values of properties).

Greater cardinality values often mean more information, thus more accurate results. In that sense, properties with higher cardinality values for an actor should have more influence in the final association value. Using weights extracted from cardinality values might be useful when dealing with concepts like social actions or interactions, but not so much for properties reflecting age or religion. That is, its use depends on the semantics of the domain model.

Table 11 contains an example of an association matrix, with resulting mean association values for the actors johndoe and mcdoe. The mined properties and classes were *maritalStatus* and *age*, Friendship and LikeAction:

- **discAssoc** (johndoe, mcdoe, *maritalStatus,* Person) = 1
- **contAssoc** (johndoe, mcdoe, *age,* Person) = 0.27
- **relClassAssoc** (johndoe, mcdoe, *friendship,* Friendship, *to, value,* Person) = 0.8
- **relClassAssoc** (mcdoe, johndoe, *friendship,* Friendship, *to, value,* Person) = 0
- **inClassAssoc** (johndoe, mcdoe, *performed,* LikeAction, *over,* Person) = 1

Table 11 - Association matrix from multiple properties and instances in figure 35.

mean association	***johndoe***	***Mcdoe***
johndoe	-	0.78
mcdoe	0.57	-

Notice that in this very small universe of data, although there is no Friendship association from mcdoe to johndoe, they both have the same tastes and *maritalStatus*.

As denoted in figure 34, data extracted through a Virtual Graph can be represented as a triple graph according to the SNO. In that sense, the results in table 11, based on the source graph depicted in figure 35, can be represented as the sink graph in figure 36.

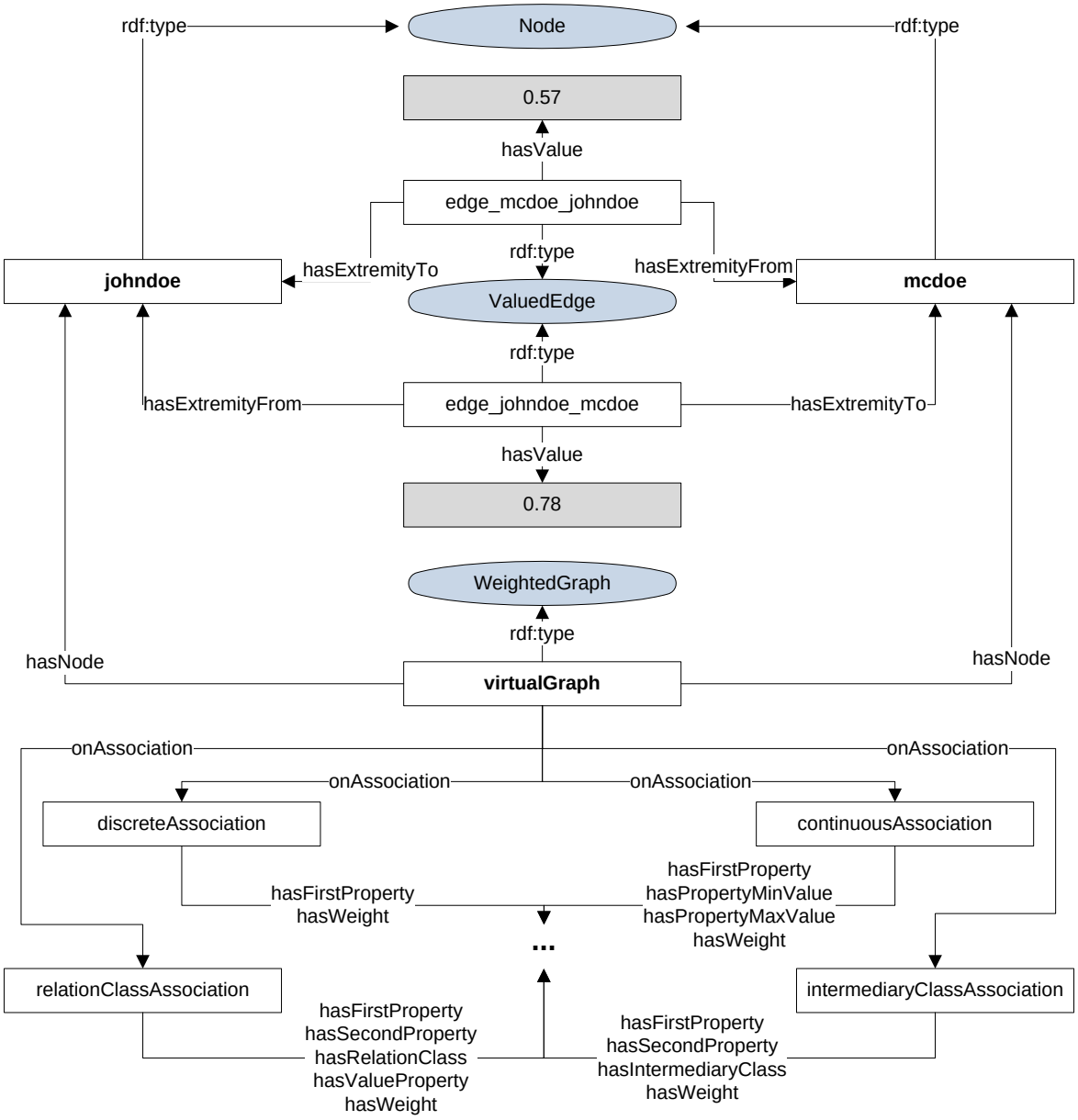

Figure 36 - Virtual Graph sink model example.

4.3. Algorithm Configuration and Execution

After a Virtual Graph is generated, a sequence of SNA algorithms fed with the Virtual Graph can be executed. The final sink graph will not only contain the Virtual Graph SNO representation, but also the input and output (the execution context) of all the executed SNA algorithms.

All SNA algorithm data, including algorithm instances, configurations, and input parameters must be present in the sink triple graph before the algorithm runs. The necessary input parameters are defined in the SNO, and can be set explicitly before the algorithm sequence runs, or set by previous SNA algorithms (as their output).

The SNA algorithm sequence is implemented as a SNA algorithm itself, containing a list of SNA algorithms and a Virtual Graph as its input. Alternatively, and in order to improve performance, a faster and optimized in-memory graph structure (e.g. using Hash Maps or Linked Lists) might be used to represent the Virtual Graph, being fed to the SNA algorithm sequence separately.

In figure 37, a Virtual Graph is fed to a SNA algorithm sequence with two SNA algorithms: the first calculates betweenness centrality for all Virtual Graph nodes, and the last generates a clustering of the Virtual Graph. Since both share the sink graph, the clustering algorithm has access to the previously calculated nodal betweenness values outputted by the betweenness centrality algorithm.

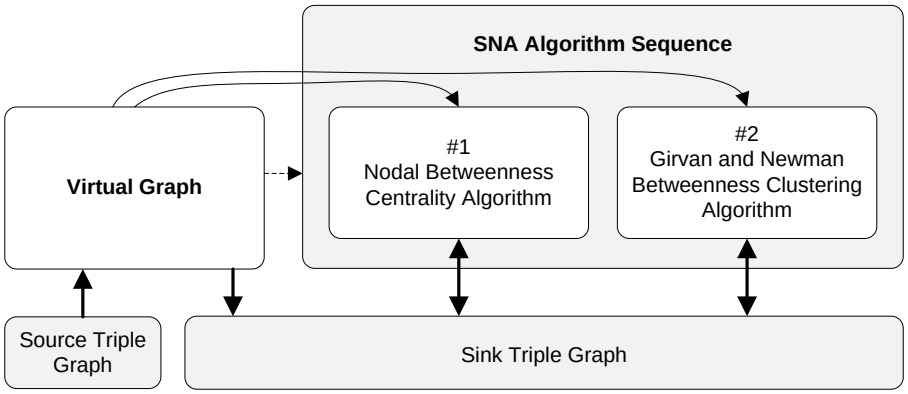

Figure 37 - SNA algorithm execution from a Virtual Graph.

Algorithm instances in the sink graph have timestamp properties containing their execution time and date. This is useful for implementations of incremental clustering or community evolution analysis algorithms.

Currently, the execution sequence is defined as an ordered list of SNA algorithms. Eventually it can evolve into a SNA workflow (e.g. featuring parallel algorithm execution).

Chapter 5
Development and Results

Toursplan is a tourism Web application that helps tourists planning their ideal trips in a simple and personalized way. It is a highly adaptive application featuring a hybrid recommendation system (Coelho et al. 2009) that relies in a strong user model.

Currently, Toursplan is neither a Social Web nor a Semantic Web application and does not allow users to interact and share experiences in a straightforward way. To embed Toursplan in a Social and Semantic Web environment, a new layered architecture has been devised. As presented in chapters 3 and 4, two ontologies have been built: the Toursplan Ontology (TPO) and the SocioNet Ontology (SNO). These models capture two important dimensions of the Toursplan application, describing user interactions, social network data and the results of several SNA algorithms. Additionally, to bridge both the SNA and Semantic Web domains, a normalization process is proposed using both these models (see chapter 4). This way, the application can be enhanced through the implementation of useful and well established social network algorithms to the tourism and travel domains.

The new Toursplan core architecture features four main layers: the Triple Store, the Data Access Layer (DAL), the Business Layer (BL), and the Web Application Layer (WAL). This analysis results from the need to add extra layers to the architecture stack as it deems necessary. An example would be a mobile application middleware[4]. Figure 38 illustrates the Toursplan logical architecture.

[4] This subject is further explored in the papers: "Luz, N., Anacleto, R. and Almeida, A. 2010. Tourism Mobile and Recommendation Systems – A State of the Art. EEE2010", "Anacleto, R., Luz, N. and Figueiredo, L. 2010. PSiS Mobile. ICWN2010" and "Anacleto, R., Luz, N. and Figueiredo, L. 2010. Personalized Sightseeing Tours Support Using Mobile Devices. WCC2010".

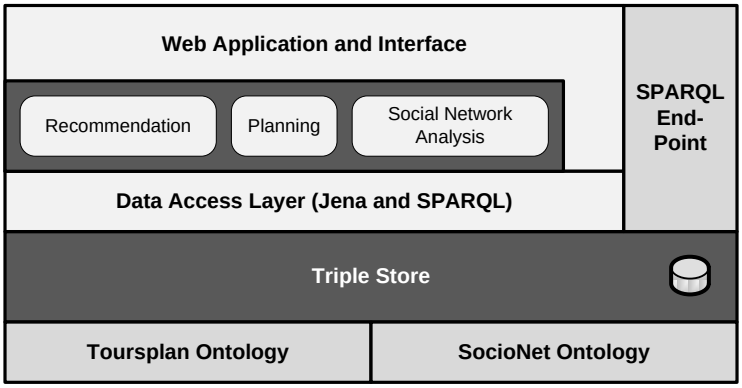

Figure 38 - Toursplan logical architecture.

At the bottom of the architecture stack lie both the TPO and SNO. The Triple Store layer contains all the data gathered and generated by the Toursplan Web application, which can be made accessible to external agents through a SPARQL end-point. On top of the triple store is the DAL, which provides data objects to deal with triples in a more straightforward fashion. Over the DAL is the BL, containing the recommendation, planning and SNA modules. This thesis focuses on the SNA module (described in chapter 4), providing a framework to analyze social data found in the triple store.

Several implementation efforts were made to enhance Toursplan and place it as a Semantic Web application, which heavily relied on the Jena Semantic Web Framework[5]. Jena is a Java framework for building Semantic Web applications, which provides a programmatic environment for RDF, RDF-S, OWL and SPARQL, also including a rule-based inference engine.

5.1. Toursplan and the Triple Store

For years, relational databases have been incredible useful to store tabular sets of data. However, there are situations where the ability of relational databases is limited (Eifrem 2009). This is the case for social network data, where a query asking for three to six degrees of separation from a person on the friendship relation (friend of a friend of a friend of…) is perfectly common, but extremely time consuming in relational databases. Graph representations fit quite well in this kind of situations.

Since one of the current main concerns regarding triple stores relies on scalability and performance, three different triple store implementations were set up namely TDB, SDB and NG4J (Named Graphs API for Jena)[6].

TDB is a high-performance, native persistence engine using custom indexing and storage, faster and simpler to set up than SDB. SDB is a persistence layer that uses an SQL database and supports full ACID (Atomicity, Consistency, Isolation, Durability) transactions. NG4J is an extension to Jena for parsing, manipulating and serializing sets of named graphs. While TDB stores triple data in binary

[5] http://jena.sourceforge.net/

[6] http://www4.wiwiss.fu-berlin.de/bizer/ng4j/

files, both SDB and NG4J were set up with a MySQL server instance. As its description suggests, TDB provided a significantly better performance (see figure 39). A simple test scenario was devised, and results are not intended to represent the performance capabilities of SDB, TDB and NG4J, but only to present a performance comparison between the different triple store implementations. A detailed analysis of performance and scalability has been left out of the scope of this thesis.

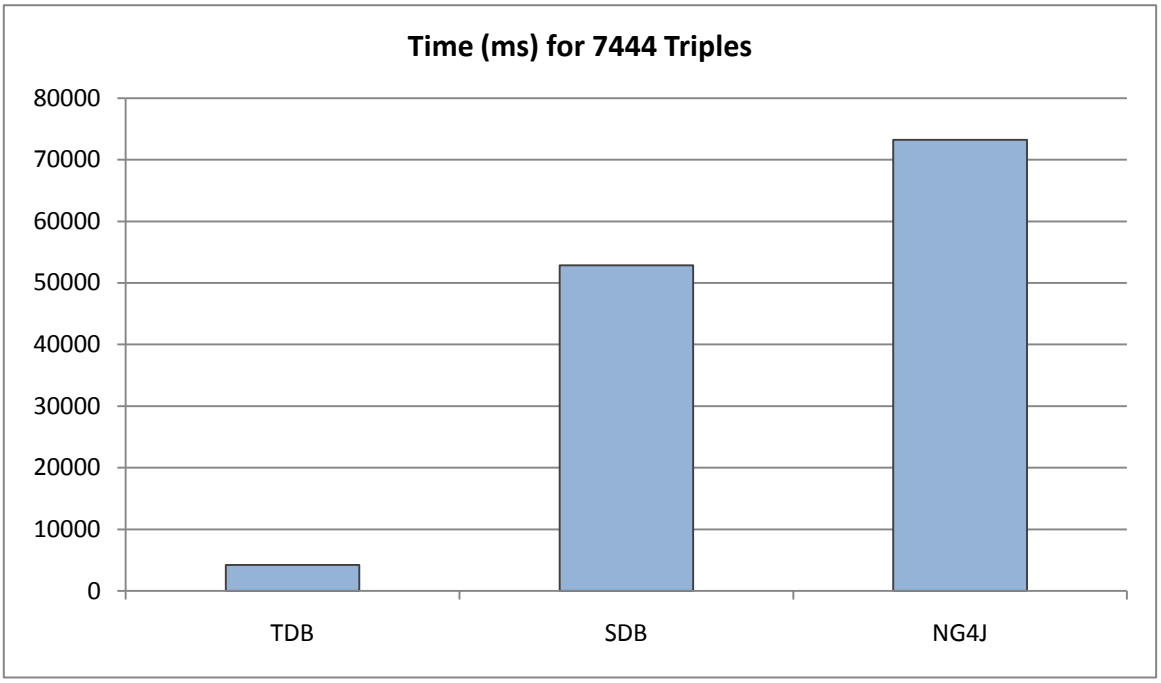

Figure 39 – Mean time in milliseconds to import a total of 7444 triples using different persistence mechanisms without optimization.

As all data in Toursplan was stored in a relational database, a migration from the Microsoft SQL Server database to the Jena TDB triple store was performed. The 7444 triples in figure 39 represent part of the migrated data, i.e. all Places, Categories and POIs imported from the relational database. However, since Toursplan is still in its development stage, no usage data could be collected. In that sense, user and interaction data had to be generated using randomly selected instances (e.g. Gender instances) or values (e.g. age values from 10 to 90). In cases where an instance (e.g. of type Person) can have more than one value for the same property (e.g. friendship), the amount of property values is also randomly picked from a restricted interval. The generated and imported data was then used to test the Toursplan SNA layer.

5.2. The Toursplan Data Access Layer

The Toursplan DAL is a Java library which provides access to Toursplan resources according to the TPO semantics. Through this model, all resources, as modeled in the TPO, are accessible as long as they exist in the underlying triple graph. Mainly, the Toursplan DAL contains a set of Java classes representing the TPO classes, where property values can be manipulated through class methods.

In Jena, a Model provides a high level abstraction over a triple graph, i.e. a perspective. While the triple graph itself can only be seen as a graph with nodes and edges, a Model class provides a meaningful and semantically enriched perspective over the triple graph resources according to an ontology model (e.g. a TPO dataset with persons and friendship relations). The Toursplan DAL provides a TPO perspective through the ToursplanModel (see figure 40), which also contains methods to create, search and remove TPO class instances (e.g. PointOfInterest, Place and Person).

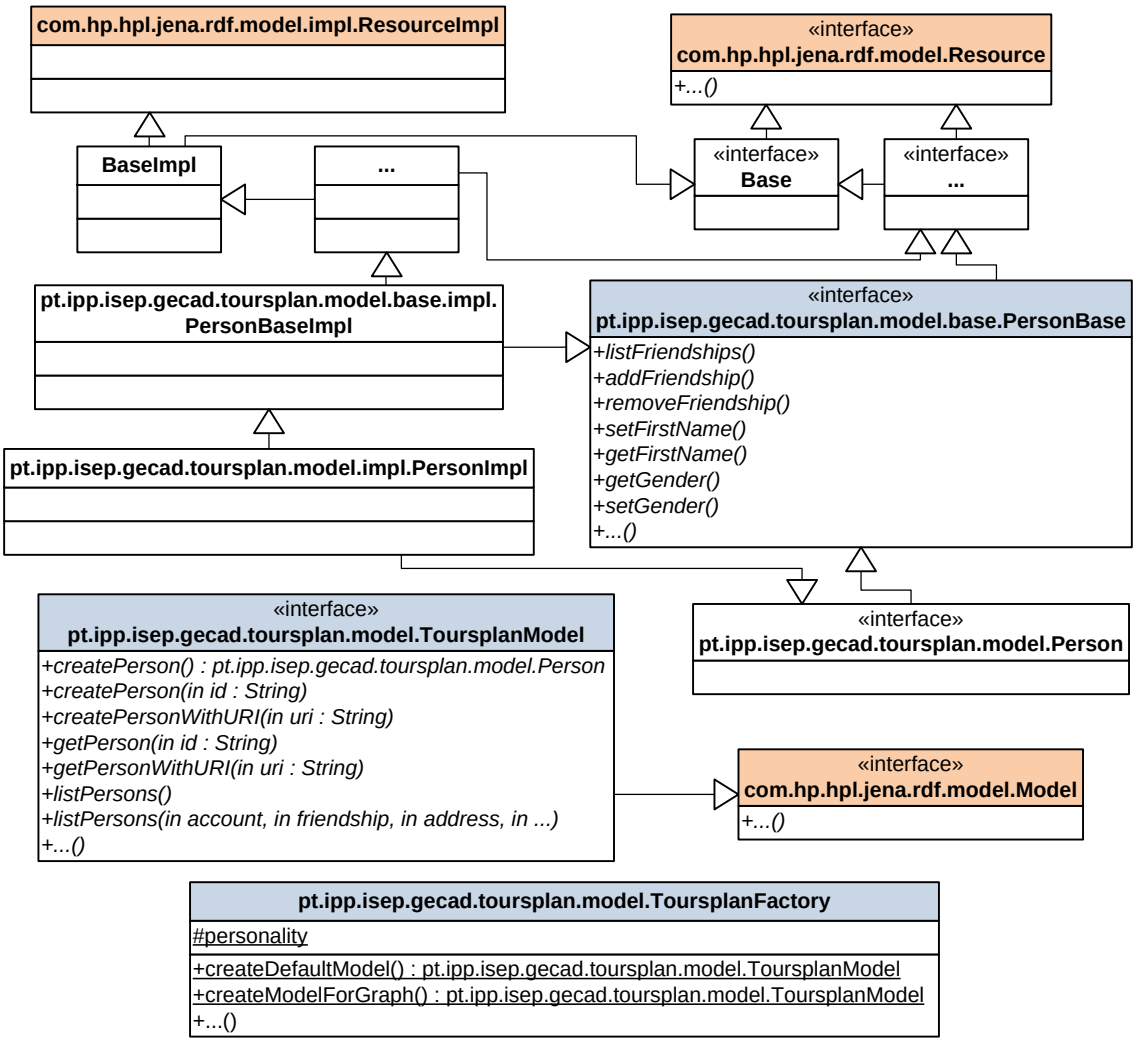

Figure 40 - Jena interface extension in the Toursplan DAL.

The ToursplanFactory is responsible for creating new ToursplanModel objects. The process is simple, since the ToursplanModel acts as a wrapper for Graph objects. Thus, the factory needs to create either a new in-memory graph and a ToursplanModel afterwards, or a ToursplanModel from an already existent graph (e.g. a TDB or NG4J graph).

Implementing new perspectives requires the use of the Jena Enhanced Node API[7]. Although the task mainly consists in creating a Java representation of the ontology classes and the model itself, it has revealed to be very time consuming and repetitive. To avoid the manual Java implementation of every ontology class, the JePerGen tool was created. It generates Jena-base perspectives (factory, model and node classes) extending the Jena Enhanced Node API, from ontologies modeled in DL. However, in the scope of this thesis, JePerGen is regarded as a secondary but necessary effort. In that sense, it is still incomplete and implemented only according to both the TPO and SNO DAL generation requirements. Further details can be found in annex 1.

As the ToursplanModel provides a TPO perspective over a triple graph, node classes such as Person provide a perspective over the graph nodes.

Although poly-hierarchy isn't allowed in Java, Jena provides a mechanism for dynamically obtaining different perspectives for the same node.

Table 12 - Jena poly-classification: node perspectives.

```
ToursplanModel m = ToursplanFactory.createDefaultModel();
//insert or load triple data to m…
for (ResIterator it = m.listResourcesWithProperty(RDF.type, TPO.Person); it.hasNext(); ) {
        Resource r = it.next(); //node as Resource
        Person p = r.as(Person.class); //get node as Person
}
```

5.3. The SocioNet Library

Similarly to the Toursplan DAL, the SocioNet library is a Java library built over Jena and generated with JePerGen (refer to annex 1 for further details). The SocioNet library also contains the manual implementation of the Virtual Graph and different SNA algorithms and measures. The currently implemented algorithms are:

- **NodalDegreeAlgorithm**: a straightforward implementation following the equations in (Wasserman & Faust 1994);
- **NodalDegreeCentralityAlgorithm**: a straightforward implementation following the equations in (Wasserman & Faust 1994);
- **NodalBetweennessCentralityAlgorithm**: an implementation of the Faster Algorithm for Betweenness Centrality presented in (Brandes 2001);
- **NodalClosenessCentralityAlgorithm**: an adaptation of the Faster Algorithm for Betweenness Centrality presented in (Brandes 2001);

[7] http://jena.sourceforge.net/enh-node-howto.html

- **FECAlgorithm**: based on multiple SNA methods (e.g. random walk, signed cut, modularity measure). Since the error measure proposed in (Yang et al. 2007) is only effective for signed graphs, an extended modularity measure for directed and/or weighted graphs (Duan et al. 2009) was needed for the FEC algorithm to properly cluster unsigned graphs;
- **SequenceAlgorithm**: a special algorithm representing a sequence of SNA algorithms. Its execution results in the ordered execution of a chain of SNA algorithms.

Although it currently represents the Toursplan SNA layer, the SocioNet library can be used independently from Toursplan, i.e. it can be used to perform SNA over any kind of model. The SocioNet contains a new special type of Model called SocioNetModel, as well as a set of classes representing different SNO node perspectives.

The SocioNet library contains SNA algorithms that can be sequentially executed and fed with a Virtual Graph. Also, it's easy to implement and add new SNA algorithms, extending the base algorithm implementation (see figure 41).

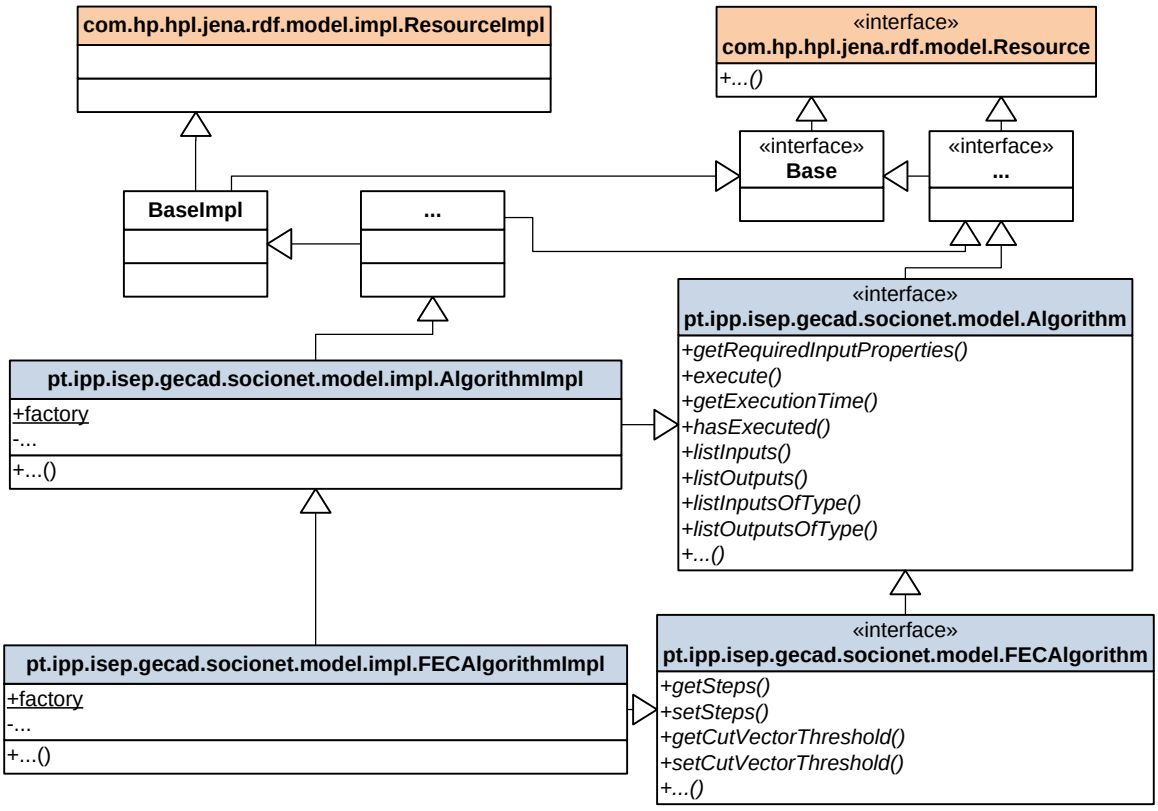

Figure 41 - SocioNet library SNA algorithm API.

Table 13 contains an example of how to use a source and sink model to execute a sequence of SNA algorithms using the SocioNet library. In the example depicted in table 13, the NodalBetweennessCentralityAlgorithm runs first and the FECAlgorithm afterwards. Notice how the sink Model is shared by all SNA algorithms, thus allowing access to the output of previously executed

algorithms. Although the Virtual Graph values are stored in the sink Model, it is possible to use an in-memory HashMap based graph implementation to feed the SNA algorithms in order to improve performance.

Table 13 - SocioNet SNA algorithm execution example using a TPO source model.

```
ToursplanModel sourceModel = ToursplanFactory.createDefaultModel();
//insert or load triple data to sourceModel…
SocioNetModel sinkModel = SocioNetFactory.createDefaultModel();

//create the Virtual Graph
VirtualGraph vGraph = sinkModel.createVirtualGraph();
vGraph.setNodeClass(TPO.Person.getURI());
vGraph.setSourceModel(sourceModel);
//cardinality-based actor associations from the Friendship relation class
RelationClassAssociation rClassAssoc = sinkModel.createRelationClassAssociation();
rClassAssoc.setHasFirstProperty(TPO.friendship.getURI(), null);
rClassAssoc.setHasSecondProperty(TPO.toPerson.getURI(), null);
rClassAssoc.setHasClass(TPO.Friendship.getURI(), null);
rClassAssoc.setHasWeight(1d);
rClassAssoc.setUseCardinality(false);
rClassAssoc.setHasValueProperty(TPO.value.getURI(), null);
vGraph.addOnAssociationAssociation(rClassAssoc);

vGraph.storeEdgesAndNodes(); //create virtual graph nodes and edges in the sink Model

//create the SNA algorithm sequence
SequenceAlgorithm seqAlg = sinkModel.createSequenceAlgorithm();
NodalBetweennessCentralityAlgorithm nBetweennessAlg =
sinkModel.createNodalBetweennessCentralityAlgorithm();
FECAlgorithm fecAlg = sinkModel.createFECAlgorithm();
fecAlg.setSteps(10);
seqAlg.addAlgorithm(nBetweennessAlg);
seqAlg.addAlgorithm(fecAlg);

//execute from HashGraph (faster)
try {
        seqAlg.run(vGraph, vGraph.asCachedHashGraphResource());
} catch(AlreadyExecutedException e) {
        e.printStackTrace();
```

```
} catch(MissingInputException e) {
        e.printStackTrace();
} catch (AlgorithmRuntimeException e) {
        e.printStackTrace();
}

//show sink Model triples
sinkModel.write(System.out, "N3");
```

All data involving SNO classes (created through the SocioNetModel) persist inside the SocioNetModel graph (e.g. the SequenceAlgorithm, its execution time, the chain SNA algorithms and their input and output).

5.4. Evaluation and Results

This thesis work has led to the development/implementation of several components:

- The TPO and SNO;
- The SocioNet library, which includes the Virtual Graph management and multiple SNA algorithms;
- The Toursplan DAL;
- The JePerGen tool;
- A Social and Semantic Web application prototype (refer to annex 2 for further details).

Despite the importance of every item in the previous list, the evaluation efforts focus on the implemented Virtual Graph and SNA algorithms of the SocioNet library.

Several datasets were used, starting from simple small graphs to medium sized network datasets. In all tests, the datasets were loaded into a ToursplanModel instance with each node being a resource of type Person with multiple randomly generated property values.

The next sub-sections describe the evaluations.

5.4.1. The Kite Graph

The Kite graph (see figure 42) contains the following characteristics:

- Undirected and Unweighted
- Node Count: 10
- Edge Count: 18
- Mean Indegree/Outdegree: 3.6
- Minimum Indegree/Outdegree: 1.0

- Maximum Indegree/Outdegree: 6.0
- Mean Betweenness: 0.122
- Mean Closeness: 0.528

As the Kite graph only represents friendship relations between actors, other relevant data about the actors had to be randomly generated and added to the triple store (see table 14). Notice that data about the actor actions is not presented, since it represents a heavy and hard to depict subset of data, especially in a bi-dimensional table format.

Table 14 - Generated data for the Kite graph dataset tests.

Person	maritalStatus	religion	creativity
toursplan_admin	-	-	-
luna_kevyn	Married	Hinduism	0.3295
jillian_mclaughlin	Married	Budism	0.1643
anika_dillard	Single	Islam	0.9619
estes_norris	Divorced	Budism	0.2097
berger_leo	Married	Budism	0.8477
kadeem_brody	Single	Christianism	0.3849
alden_teagan	Married	Judaism	0.1763
mcknight_compton	Single	Hinduism	0.7551
aimee_colorado	Married	Islam	0.5539

The following figures 42, 43, 44, 45 and 46 contain a FEC clustering of different Virtual Graphs with different Associations.

The Virtual Graph in figure 42 contains a cardinality-based Relation Class Association over the *friendship* relation of the TPO. The application of the cardinality-based association results in a directed graph since the cardinality values for the *friendship* property are different for different nodes, even though the original Kite graph is undirected.

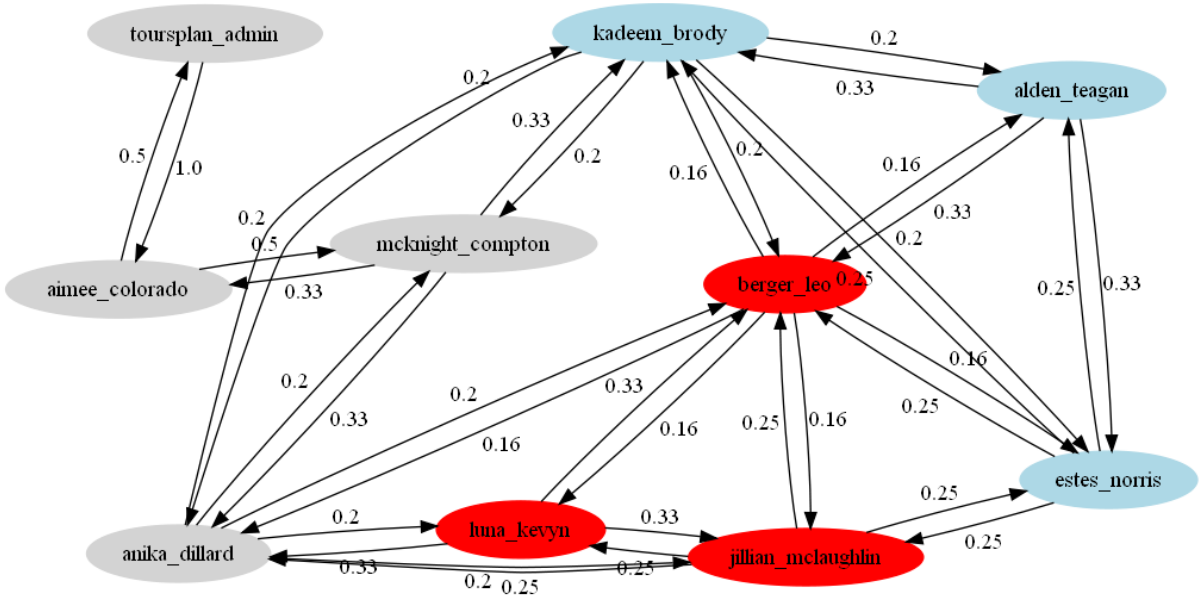

Figure 42 - Kite dataset clustering from the FEC algorithm using a cardinality-based Relation Class Association over the Friendship class.

Figure 43 depicts a Virtual Graph with a cardinality-based Intermediary Class Association over the LikeAction class. For a link to exist in the graph, both actors must like one or more POIs in common. Since the actor toursplan_admin has no generated data besides the Kite graph *friendship* relations, the node appears isolated.

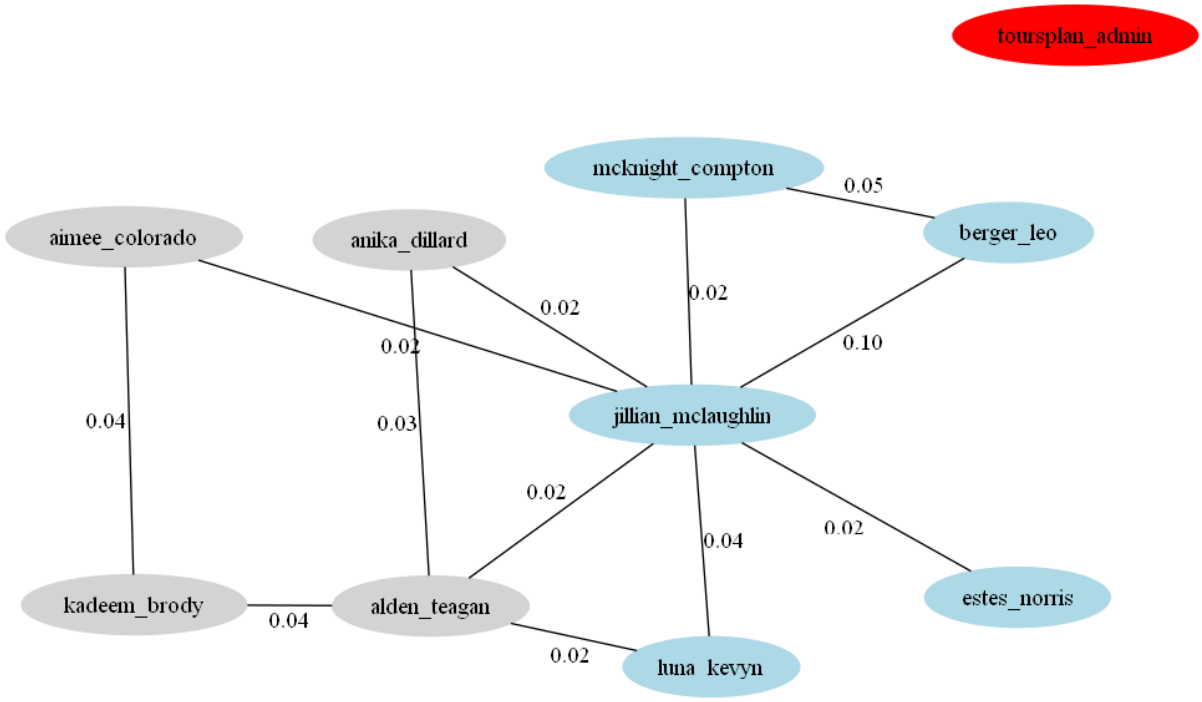

Figure 43 - Kite dataset clustering using a cardinality-based Intermediary Class Association over the LikeAction class.

The Virtual Graph in figure 44 contains a simple Discrete Property Association over the *maritalStatus* property. Only actors with equal *maritalStatus* values are linked.

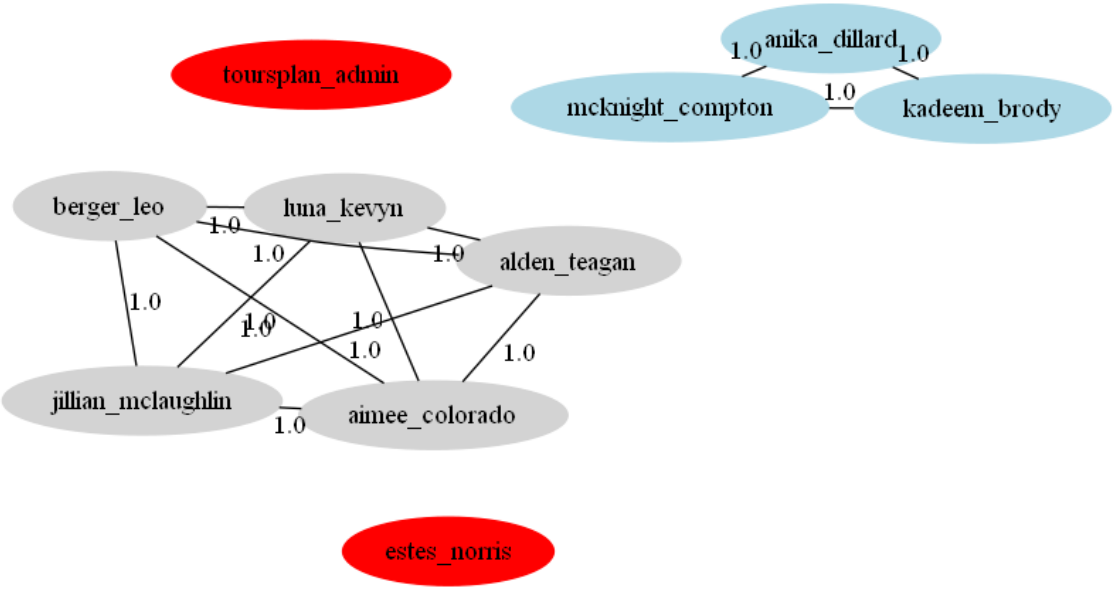

Figure 44 - Kite graph dataset clustering using a Discrete Property Association over the *maritalStatus* property.

Figure 45 depicts a Virtual Graph with a Continuous Property Association over the *creativity* property. Excluding the toursplan_admin node the resulting graph is complete, with values ranging from 0 to 1 (there are no 0 values since edges would be missing). Higher values mean similar creativity indices.

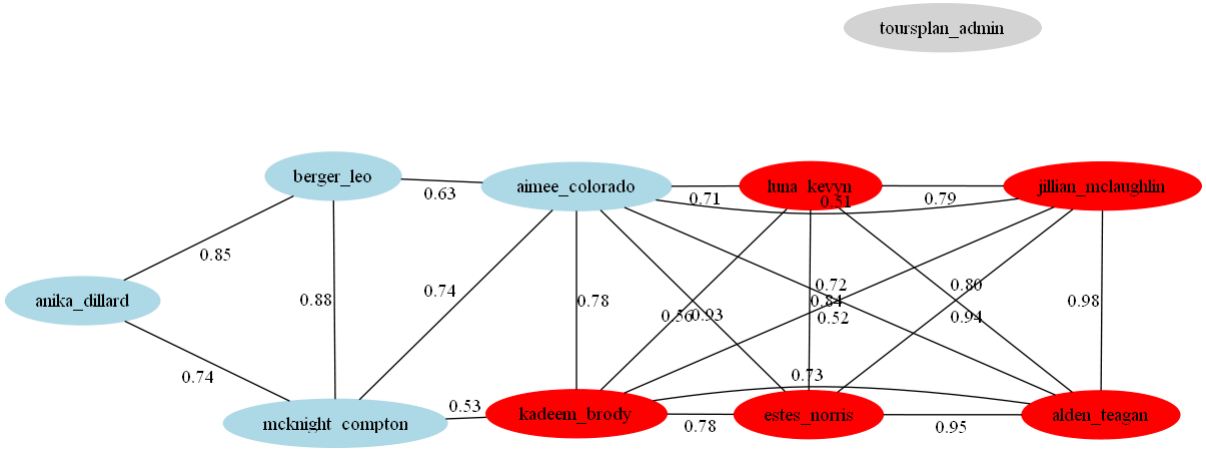

Figure 45 - Kite graph dataset clustering using a Continuous Property Association over the *creativity* property. As a matter of simplification, and although the graph is complete, edges with values under 0.5 were removed.

In Figure 46, a Virtual Graph with both a cardinality-based Relation Class Association over the Friendship class, and a cardinality-based Intermediary Class Association over the LikeAction class is depicted. Results represent a merging of the graphs in figures 42 and 43.

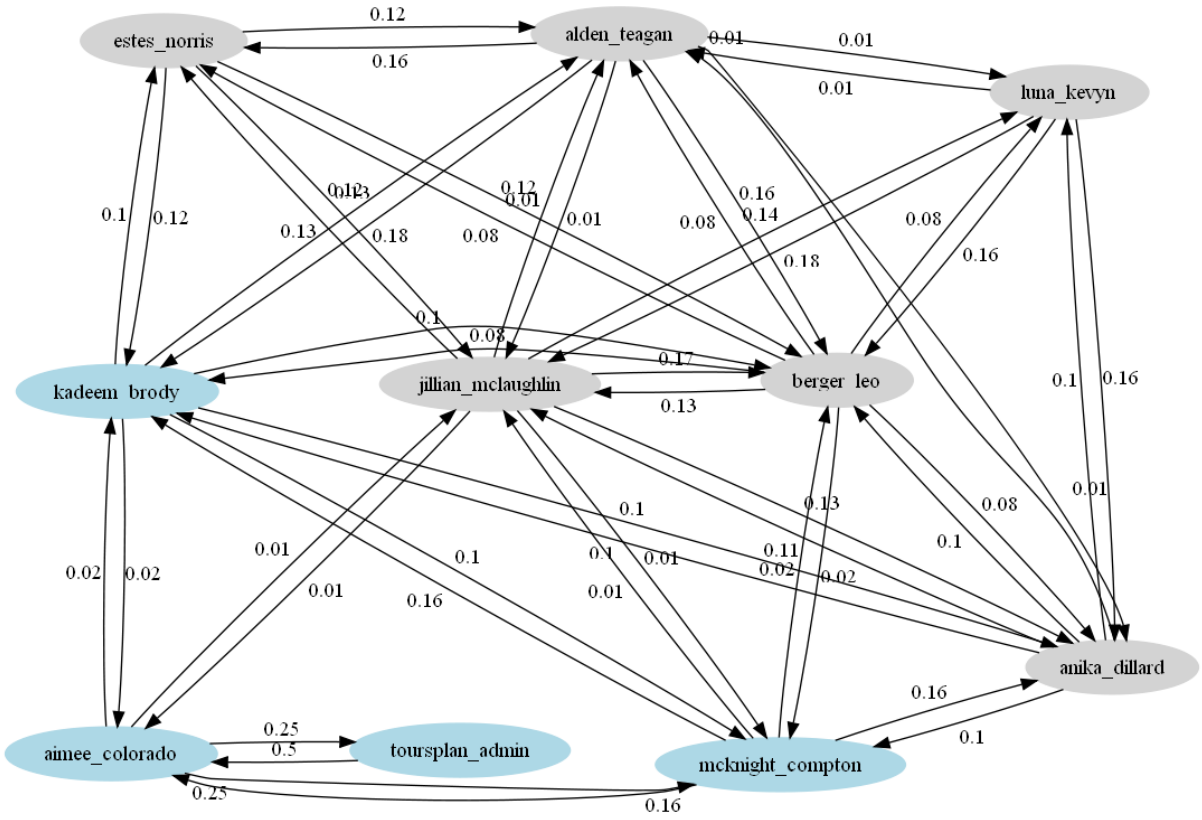

Figure 46 - Kite graph dataset using a cardinality-based Relation Class Association over the Friendship class and a cardinality-based Intermediary Class Association over the LikeAction class.

5.4.2. The Zachary's Karate Club Network

The Zachary's Karate Club Network dataset[8] (see figure 47) contains the following characteristics:

- Undirected and Unweighted
- Node Count: 34
- Edge Count: 78
- Mean Indegree/Outdegree: 4.588
- Minimum Indegree/Outdegree: 1.0
- Maximum Indegree/Outdegree: 17.0
- Mean Betweenness: 0.044
- Mean Closeness: 0.426

Table 15 contains relevant randomly generated information about actors in the dataset.

[8] http://networkdata.ics.uci.edu/data.php?id=105

Table 15 - Generated data for the Zachary's Karate Club Network dataset tests.

Person	maritalStatus	religion	creativity
toursplan_admin	-	-	-
flynn_cameron	Single	Judaism	0.8245
latifah_bertha	Divorced	Christianism	0.1586
angela_ora	Divorced	Christianism	0.7796
lawrence_walsh	Single	Judaism	0.0565
lacy_blanchard	Divorced	Hinduism	0.5608
ballard_golden	Divorced	Hinduism	0.1814
conan_crane	Single	Atheism	0.1300
amelia_melyssa	Single	Hinduism	0.2813
cohen_phillip	Single	Atheism	0.5163
aladdin_yvette	Married	Christianism	0.0363
jingles_emmanuel	Single	Judaism	0.6132
wilkerson_lenore	Married	Atheism	0.9817
cole_fuller	Married	Hinduism	0.9235
beasley_lloyd	Single	Christianism	0.0509
bradford_aguirre	Divorced	Atheism	0.2979
drew_hayden	Single	Atheism	0.4276
wanda_burton	Married	Budism	0.7692
mann_houston	Divorced	Judaism	0.9046
susan_garrison	Married	Christianism	0.4005
kyle_kaye	Married	Hinduism	0.9752
dickson_timothy	Divorced	Judaism	0.2332
carpenter_davenport	Single	Atheism	0.6734
lopez_wilcox	Single	Judaism	0.6326
bo_erica	Single	Islam	0.4999
sandra_quinn	Divorced	Hinduism	0.2723
kieran_sylvia	Married	Atheism	0.3366
eugenia_russo	Married	Hinduism	0.2881
houston_cadman	Married	Judaism	0.7582
brutus_leah	Divorced	Budism	0.8434

jaquelyn_wanda	Married	Hinduism	0.8291
barlow_rodriquez	Divorced	Judaism	0.9820
hoover_halla	Married	Judaism	0.0333
willa_strickland	Divorced	Islam	0.7061

Figures 47, 48 and 49 depict respectively, a FEC clustering of different Virtual Graphs with a Relation Class Association, a Discrete Property Association and a Continuous Property Association.

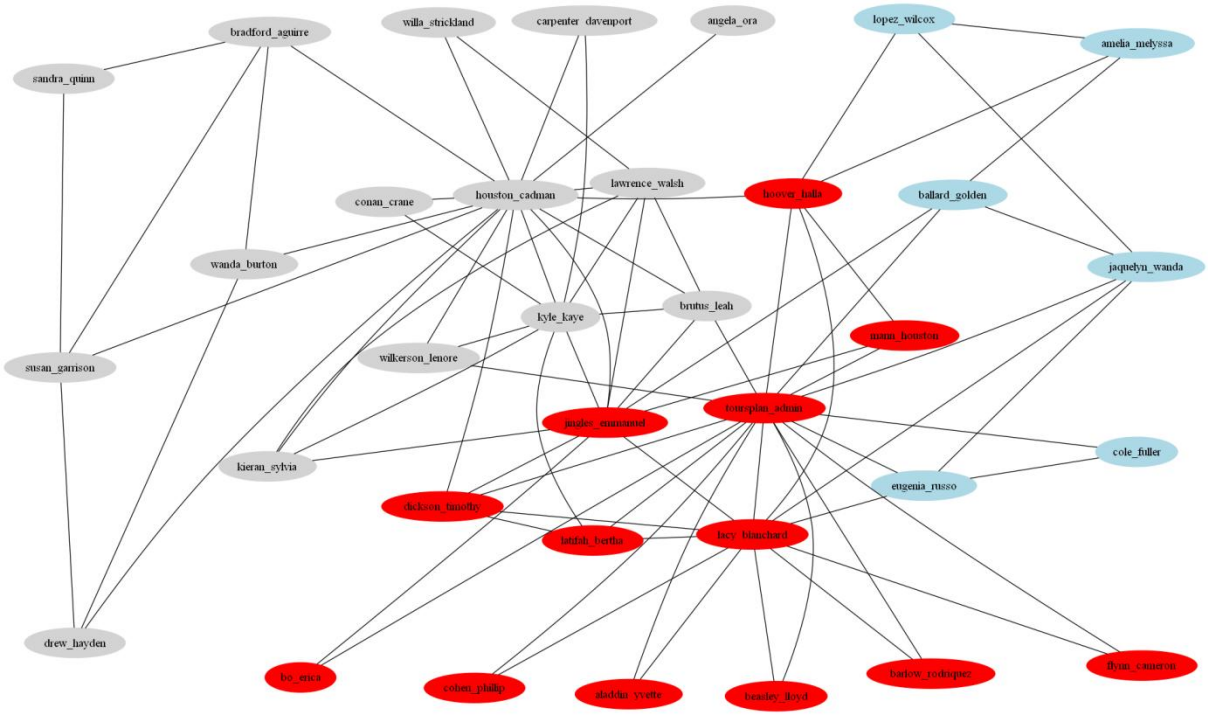

Figure 47 – Karate dataset FEC clustering from the FEC algorithm using a Relation Class Association over the Friendship class.

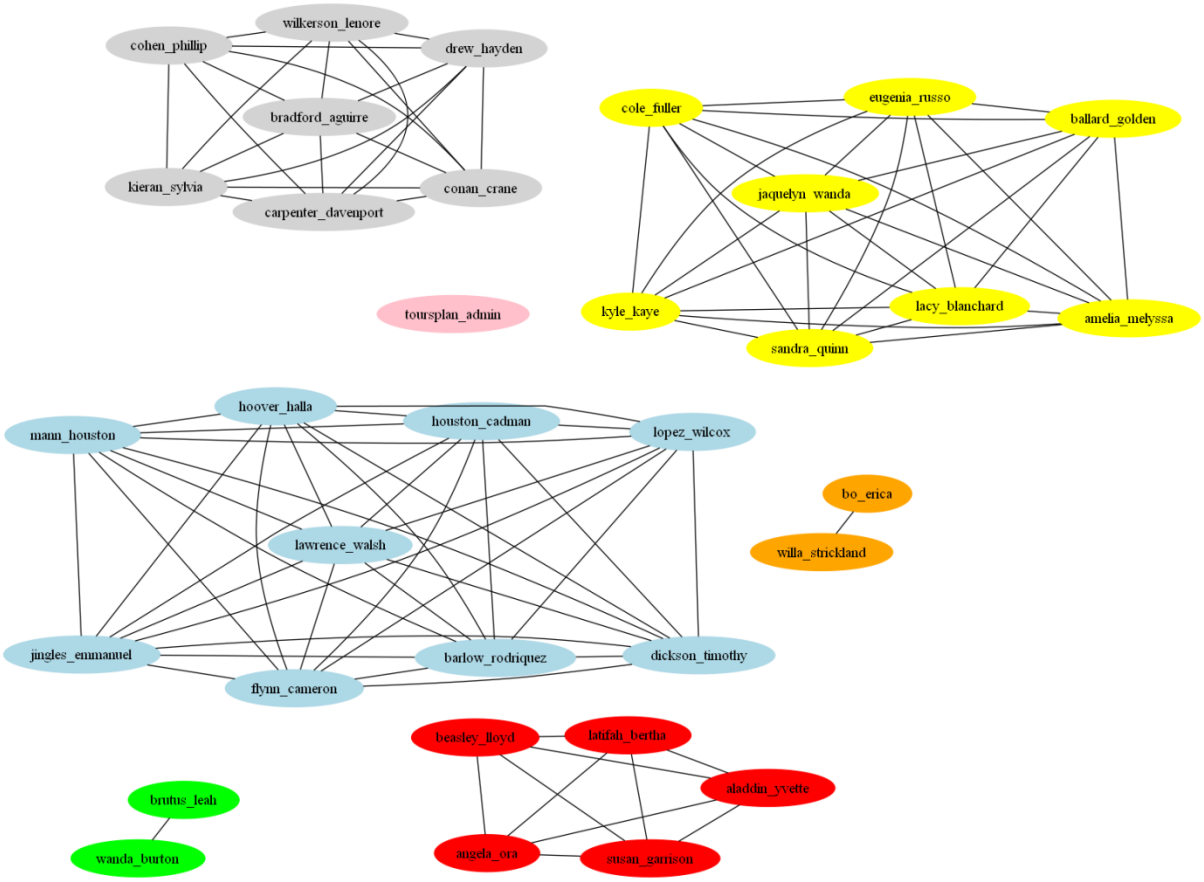

Figure 48 - Karate dataset FEC clustering using a Discrete Property Association over the *religion* property.

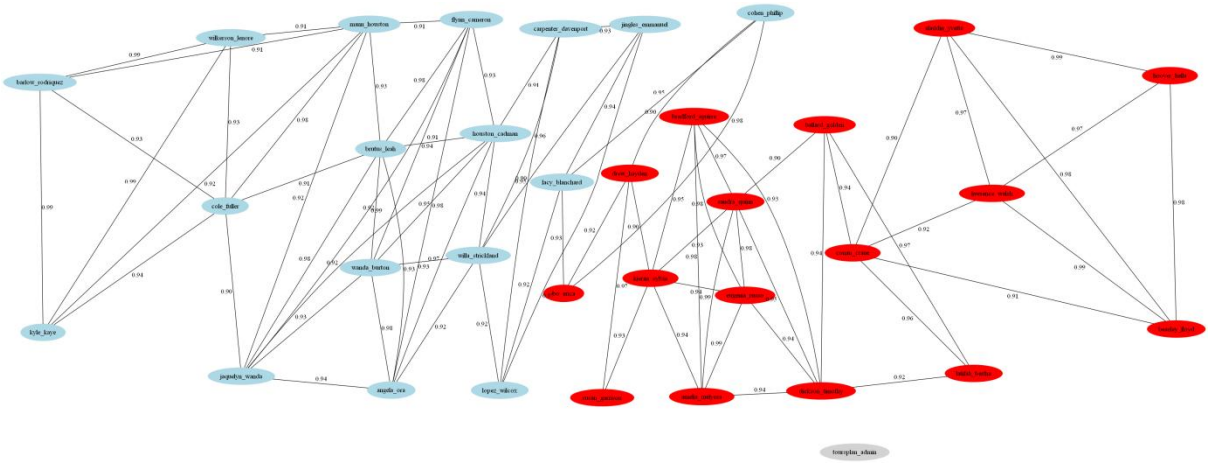

Figure 49 - Karate dataset FEC clustering using a Continuous Property Association over the *creativity* property. As a matter of simplification, and although the graph is complete, edges with values under 0.9 were removed.

5.4.3. A Bernoulli Distribution Weighted Random Graph

This graph was randomly generated using a Bernoulli distribution were valued edges had a 10% chance of having values higher than 0.

The resulting graph characteristics are:

- Directed and Weighted
- Node Count: 20
- Edge Count: 40
- Mean Indegree/Outdegree: 0.953
- Minimum Indegree/Outdegree: 0.0
- Maximum Indegree/Outdegree: 2.345/3.025
- Mean Weighted Betweenness: 0.081
- Mean InCloseness/OutCloseness: 0.211/0.136

Results for the random graph are presented in figures 50 and 51, from Virtual Graphs with a Relation Class Association and a cardinality-based Intermediary Class Association, respectively.

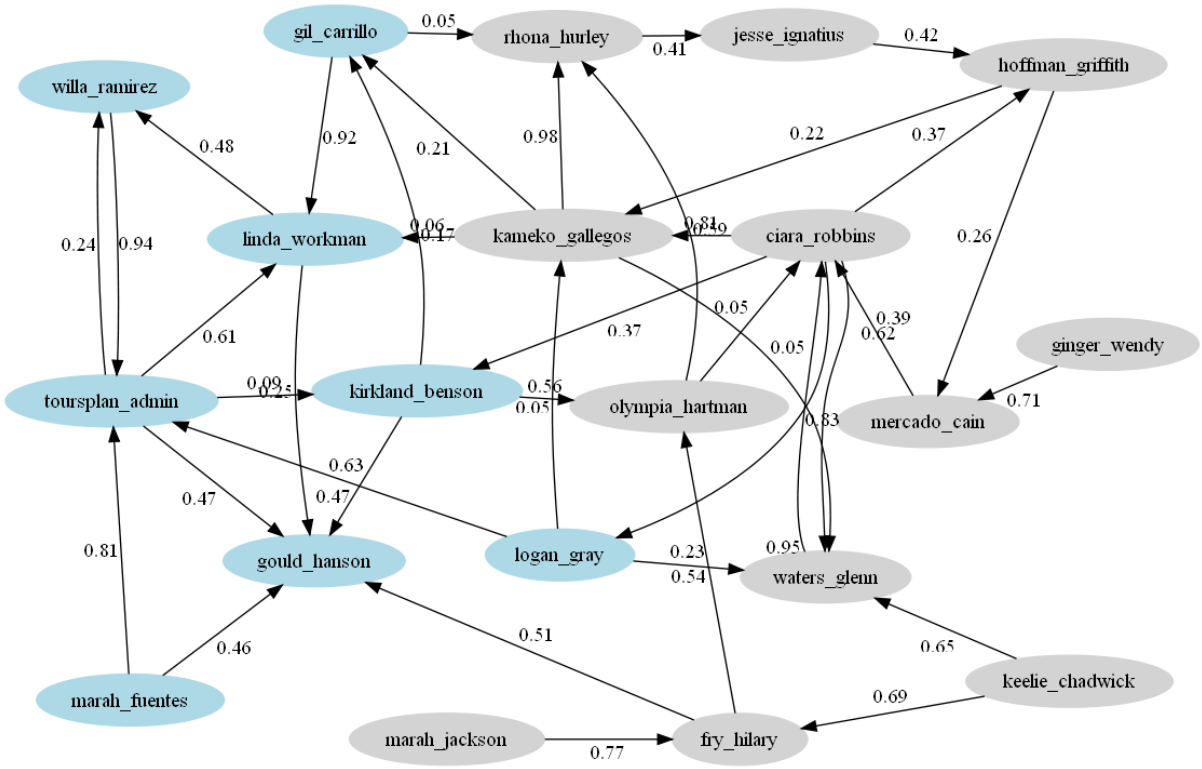

Figure 50 - Random graph dataset FEC clustering using a Relation Class Association over the Friendship class.

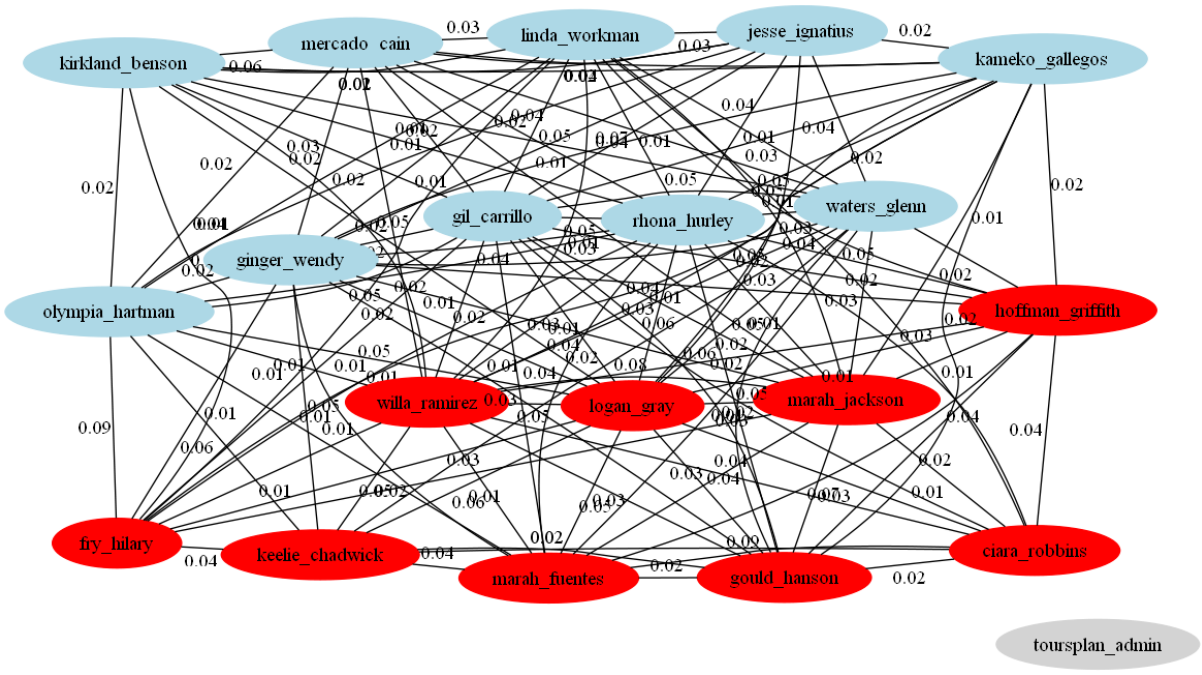

Figure 51 - Random graph dataset FEC clustering using a cardinality-based Intermediary Class Association over the BeenThereAction class.

5.5. Summary

This chapter presented several implementation efforts necessary for Toursplan to become social and semantically enriched.

Evaluation results validate the approach presented in chapter 4, proving that the Virtual Graph represents indeed an abstraction over complex triple graphs built from semantically enriched models. Also, multiple examples of different triple graph abstractions according to different modeling patterns are presented, along with its FEC algorithm clustering results.

The depicted results show that Virtual Graph abstractions can be used to extract relevant information from social network triple data, and correlate different actor attributes and properties using multiple SNA algorithms, providing answers to multiple sociological aspects of the network (e.g. if actors with the same marital status tend to form friendship relations between them).

Chapter 6
Conclusions and Future Work

Over the last five years, online social networks have been the focus of large and still growing amounts of attention. But with the advent of the Semantic Web, new ways and technologies for modeling, storing, sharing and linking data emerged. This thesis goal was to bring Toursplan into the Social and Semantic Web environment through the use of such methods and technologies.

In that sense, two different state of the art analyses were performed. The first focused on online social networks, evaluating their interface and features. This analysis aided in the implementation of the interface prototype for the Toursplan Web application, and in the creation of the Toursplan Ontology, which describes the application domain of data. The last state of the art analysis focused in SNA methods and algorithms, which led to the creation of the SocioNet library and the SocioNet Ontology.

The analysis and characterization of the algorithms presented in chapter 3 is of paramount importance to the correct selection and application of SNA algorithms to concrete SNA scenarios. The SocioNet ontology not only provides a classification of multiple SNA algorithms according to their input and output, but also describes a series of important SNA concepts that allow the description of multiple SNA algorithm executions with different execution contexts.

On top of the SocioNet ontology, the SocioNet library not only introduces the concept of Virtual Graph, allowing the extraction and normalization of social data from different data models according to different modeling patterns, but also features multiple SNA algorithms that run over a Virtual Graph, storing all their input and output data in a SocioNet ontology data model. This may not only be relevant to perform an analysis of the network evolution, but also to share and compare the results given by multiple SNA algorithms.

Backed up by semantically-rich domain models and triple graph data structures, Toursplan now runs over a highly flexible DAL with rule inference capabilities.

To avoid the manual implementation of the Java classes representing both the TPO and SNO model classes, a Jena Perspective Generator (JePerGen) was implemented. Although JePerGen correctly generates the TPO and SNO model classes, results highly depend on how the ontology is modeled. Currently, it only works for DL ontologies.

During the analysis and implementation phases, multiple ideas emerged. While some are considered future work, others might slightly fall out of the scope of this thesis. Future work tasks include:

- A detailed analysis of multiple graph persistence layers, evaluating multiple dimensions such as performance and scalability in a Web application environment;
- Use the resulting data from the SocioNet library SNA algorithm execution to improve the recommendation system results, especially through clustered collaborative filtering;
- Implement new SNA algorithms in SocioNet, including resume mining of communities and positional analysis algorithms;
- Improve and complete the SocioNet Web application interface and provide an entire framework for Semantic Web SNA;
- Complete the Toursplan Web application interface and features;
- Adapt both the recommendation and planning systems so they can feed on data from the triple store;
- Complete the analysis of the computational complexity of the SNA algorithms presented in chapter 3, and thereafter evolve the SocioNet ontology to express that new dimension of the classification;
- Test the SocioNet ontology in multiple practical scenarios, to further determine its usability as part of the Linked Data Web. An example would be a scenario where algorithms are automatically chosen according to the social network context and characteristics;
- Optimize and evolve JePerGen, so it can effectively generate Jena ontology perspectives for multiple ontologies modeled in different ways and serialized in different formats;
- Perform a detailed analysis of how the triple store and inference mechanisms can contribute to new or improved features in the Toursplan application.

Also, since Toursplan lacked the necessary data to provide the results in chapter 5, multiple instance generators had to be implemented. Since the ontology model already contains most of the required information for generating new instances and property values, an ontology-based Jena instance generator tool could be quite useful in similar situations. For information not contained in the ontology model such as a set of names for the firstName data type property, simple restriction rules could be used.

As both the SocioNet and Toursplan ontologies can be considered as ever evolving models, their maintenance and evolution is envisaged.

References

Bansal, N., Blum, A. & Chawla, S., 2004. Correlation Clustering. *Machine Learning*, 56(1), 89-113.

Brandes, U., 2001. A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology*, 25, 163-177.

Cai, D. et al., 2005. Community Mining from Multi-relational Networks. In *Knowledge Discovery in Databases: PKDD 2005*. pp. 445-452. Available at: http://dx.doi.org/10.1007/11564126_44 [Accessed March 19, 2010].

Chen, J., Zaıane, O.R. & Goebel, R., 2009. Detecting Communities in Social Networks using Max-Min Modularity.

Clauset, A., Newman, M. & Moore, C., 2004. Finding community structure in very large networks. Available at: http://dx.doi.org/10.1103/PhysRevE.70.066111 [Accessed March 19, 2010].

Coelho, B., Figueiredo, A. & Martins, C., 2009. Tours Planning Decision Support. In *ISCIES09*. Porto, Portugal.

Ding, C.H.Q. et al., 2001. A Min-max Cut Algorithm for Graph Partitioning and Data Clustering. In *Proceedings of the 2001 IEEE International Conference on Data Mining*. IEEE Computer Society, pp. 107-114.

Duan, D. et al., 2009. Community Mining on Dynamic Weighted Directed Graphs. In *Proceeding of the 1st ACM international workshop on Complex networks meet information & knowledge management*. Hong Kong, China: ACM, pp. 11-18.

Eifrem, E., 2009. Neo4j - The Benefits of Graph Databases. Available at: http://www.slideshare.net/emileifrem/neo4j-the-benefits-of-graph-databases-oscon-2009 [Accessed August 9, 2010].

Erétéo, G. et al., 2009. Semantic Social Network Analysis.

Fan, T., Liau, C. & Lin, T., 2007. Positional Analysis in Fuzzy Social Networks. In *Proceedings of the 2007 IEEE International Conference on Granular Computing*. IEEE Computer Society, p. 423.

Flake, G.W., Lawrence, S. & Giles, C.L., 2000. Efficient identification of Web communities. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. Boston, Massachusetts, United States: ACM, pp. 150-160.

Fowler, R.F. & Greenough, C., 1998. *Ralpar-RAL Mesh Partitioning Program: version 2.0*, Citeseer.

Freeman, L., 1977. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1), 41, 35.

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

Handcock, M., Raftery, A. & Tantrum, J., 2007. Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2), 354, 301.

Hanneman, R.A. & Riddle, M., 2005. Introduction to social network methods. *Riverside, CA: University of California, Riverside*.

Jackson, M.O., 2008. Average distance, diameter, and clustering in social networks with homophily. In *Workshop on Internet and Network Economics (WINE)*. pp. 4–11.

Jain, A.K., Murty, M.N. & Flynn, P.J., 1999. Data clustering: a review. *ACM Comput. Surv.*, 31(3), 264-323.

Kadushin, C., 2004. Introduction to Social Network Theory. *Unpublished manuscript*.

Karypis, G. & Kumar, V., 1998. Multilevel k-way partitioning scheme for irregular graphs. *J. Parallel Distrib. Comput.*, 48(1), 96-129.

Kleinberg, J. & Tardos, É., 2005. *Algorithm Design* United States ed., Addison Wesley.

Kozinets, R.V., 1999. E-tribalized marketing?: The strategic implications of virtual communities of consumption. *European Management Journal*, 17(3), 252–264.

Levine, S.S. & Kurzban, R., 2006. Explaining Clustering in Social Networks: Towards an Evolutionary Theory of Cascading Benefits. *SSRN eLibrary*, 27, 173-187.

Mishra, N. et al., 2007. Clustering Social Networks. In *Algorithms and Models for the Web-Graph*. pp. 56-67. Available at: http://dx.doi.org/10.1007/978-3-540-77004-6_5 [Accessed March 19, 2010].

Newman, M.E.J. & Girvan, M., 2004. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 026113.

Porter, J., 2008. *Designing for the Social Web*, Peachpit Press.

Segaran, T., 2007. *Programming Collective Intelligence: Building Smart Web 2.0 Applications* 1st ed., O'Reilly Media.

Shi, J. & Malik, J., 2000. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 888-905.

Spivack, N., 2004. The Future of the Net. *Minding the Planet*. Available at: http://www.novaspivack.com/science/new-version-of-my-metaweb-graph-the-future-of-the-net [Accessed July 13, 2010].

Stoer, M. & Wagner, F., 1997. A simple min-cut algorithm. *J. ACM*, 44(4), 585-591.

Vieira, A. & Lavos, R., 2010. Sairmais.com, a new web 2.0 portal for tourism information with a Recommender System. *Revista Turismo & Desenvolvimento*, 13, 621-630.

Von Luxburg, U., 2007. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17(4), 395-416.

Wasserman, S. & Faust, K., 1994. *Social Network Analysis: Methods and Applications* 1st ed., Cambridge University Press.

Wu, B. et al., 2007. Resume Mining of Communities in Social Network. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*. 2007 Seventh IEEE International Conference on Data Mining - Workshops (ICDM Workshops). Omaha, NE, USA, pp. 435-440.

Xu, X. et al., 2007. SCAN: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. San Jose, California, USA: ACM, pp. 824-833.

Yang, B., Cheung, W. & Liu, J., 2007. Community Mining from Signed Social Networks. *IEEE Trans. on Knowl. and Data Eng.*, 19(10), 1333-1348.

Yang, B. et al., 2008. On Modularity of Social Network Communities: The Spectral Characterization. In *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*. Los Alamitos, CA, USA: IEEE Computer Society, pp. 127-133.

Yun, S. & Kim, H., 2009. Modeling User Interactions in Online Social Networks. Available at: http://www.slideshare.net/Channy/modeling-user-interactions-in-online-social-networks [Accessed March 19, 2010].

Zhou, W.J. et al., 2002. A concentric-circle model for community mining in graph structures. *Microsoft Research, Seattle, Technical. Report MSR-TR-2002*, 123.

Annex 1
The Jena Perspective Generator (JePerGen)

JePerGen generates Jena base perspectives (model and node classes) extending the Jena Enhanced Node API, from ontologies modeled in DL. The base perspective consists in a model factory class, a model class, and in a Java base class and interface for each ontology class, including methods to easily access instance property values. Refer to figure 40 for some of the generated classes of the TPO perspective.

The top level classes and interfaces contain no code and are only generated if the file doesn't exist. They exist so that manual code implementation apart from the automatically generated code is possible. Generated code resides inside the base package classes and interfaces, which are overwritten every time JePerGen runs. Since Java does not allow class poly-hierarchy, class extensions are applied to Java classes using the first direct super class retrieved through the Jena Ontology API.

An ontology can have multiple restrictions defining multiple classes. To properly generate the Java class methods, all the direct restrictions of the ontology class are evaluated. When evaluating these restrictions, multiple issues arise:

- There are multiple types of restrictions (e.g. maximum and minimum cardinality, existential and universal quantification);
- There might be multiple restrictions over the same property, defining the same class;
- Negation, and the existence of intersection and union classes with or within restrictions;

The first issue is solved by properly merging all the restrictions over a single property. However, the TPO and SNO do not contain such restrictions. Thus, restriction merging is currently not implemented in JePerGen. The same applies to the third issue.

Also, currently, no distinction between existential and universal quantification restrictions is made.

Methods are named according to both the property name and its range (if specified). This avoids problems when restrictions over the same property apply to both a class and its super class, with different specified ranges, thus tackling the second issue.

Unless an exact or maximum cardinality restriction equal to 1 is applied, a list method is generated. Table 16 contains the JePerGen generated Java interface for the Person TPO class (see table 7 for its Manchester syntax serialization).

Table 16 - JePerGen generated Java interface for the Person TPO class.

```
public interface PersonBase extends SocialThing {

        public Double getPerfectionism();
        public void setPerfectionism(Double obj);
        public void removePerfectionism();

        public String getAddress(String lang);
        public boolean addAddress(String obj, String lang);
        public boolean removeAddress(String obj, String lang);
        public void removeAllAddresss();
        public ExtendedIterator<String> listAddresss();

        public Account getAccountAccount();
        public boolean addAccountAccount(Account obj);
        public boolean removeAccountAccount(Account obj);
        public void removeAllAccountAccounts();
        public ExtendedIterator<Account> listAccountAccounts();

        public Friendship getFriendshipFriendship();
        public boolean addFriendshipFriendship(Friendship obj);
        public boolean removeFriendshipFriendship(Friendship obj);
        public void removeAllFriendshipFriendships();
        public ExtendedIterator<Friendship> listFriendshipFriendships();

        public Double getMobilityHandicap();
        public void setMobilityHandicap(Double obj);
        public void removeMobilityHandicap();

        public Place getLocationPlace();
        public boolean addLocationPlace(Place obj);
        public boolean removeLocationPlace(Place obj);
        public void removeAllLocationPlaces();
        public ExtendedIterator<Place> listLocationPlaces();


        public SocialAction getPerformedSocialAction();
        public boolean addPerformedSocialAction(SocialAction obj);
        public boolean removePerformedSocialAction(SocialAction obj);
        public void removeAllPerformedSocialActions();
        public ExtendedIterator<SocialAction> listPerformedSocialActions();

        ...
}
```

Annex 2

Front and Back Office Prototypes

A2.1. The Toursplan Web Application

The online social network state of the art described in chapter 2 served as a basis for the creation of a new Social Web user interface. The following interface screenshots come from a prototype version of the Toursplan interface that currently lacks some of the Web applications' intended features.

The Toursplan home page has a brief description (or links to a description) of what is Toursplan, who can use it, and how it works (see figure 52 for a prototype). With further evolution and use of the online social network, other important information can be added (e.g. top points of interest or places).

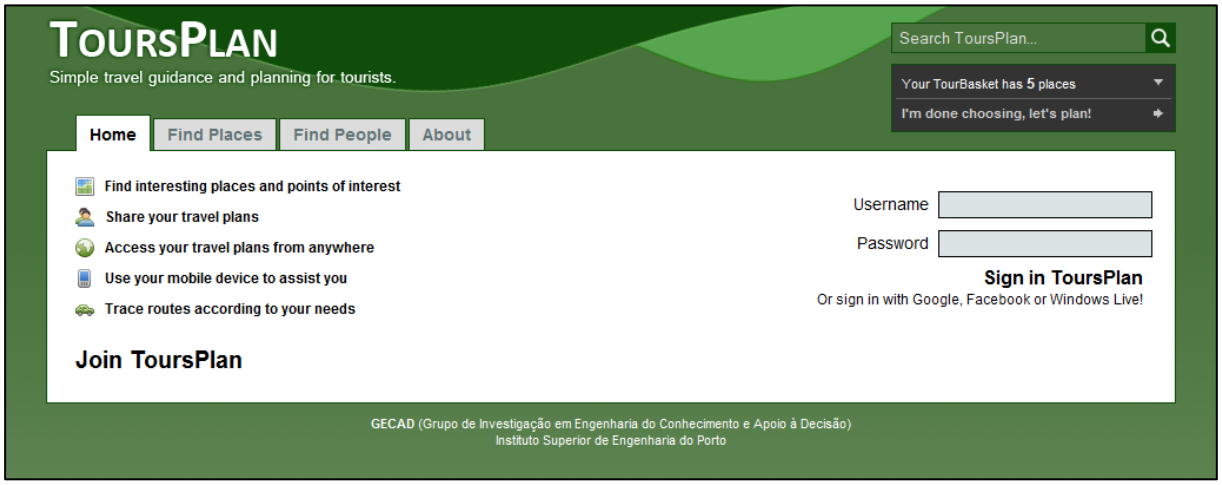

Figure 52 - The Toursplan online social network prototype user interface for an anonymous user.

Using the TPO, the new Toursplan online social network is structured in a way so that information can be filtered by places, and recommendations of multiple types of POIs (e.g. hotels, restaurants and sights) can be given by place (see figure 53).

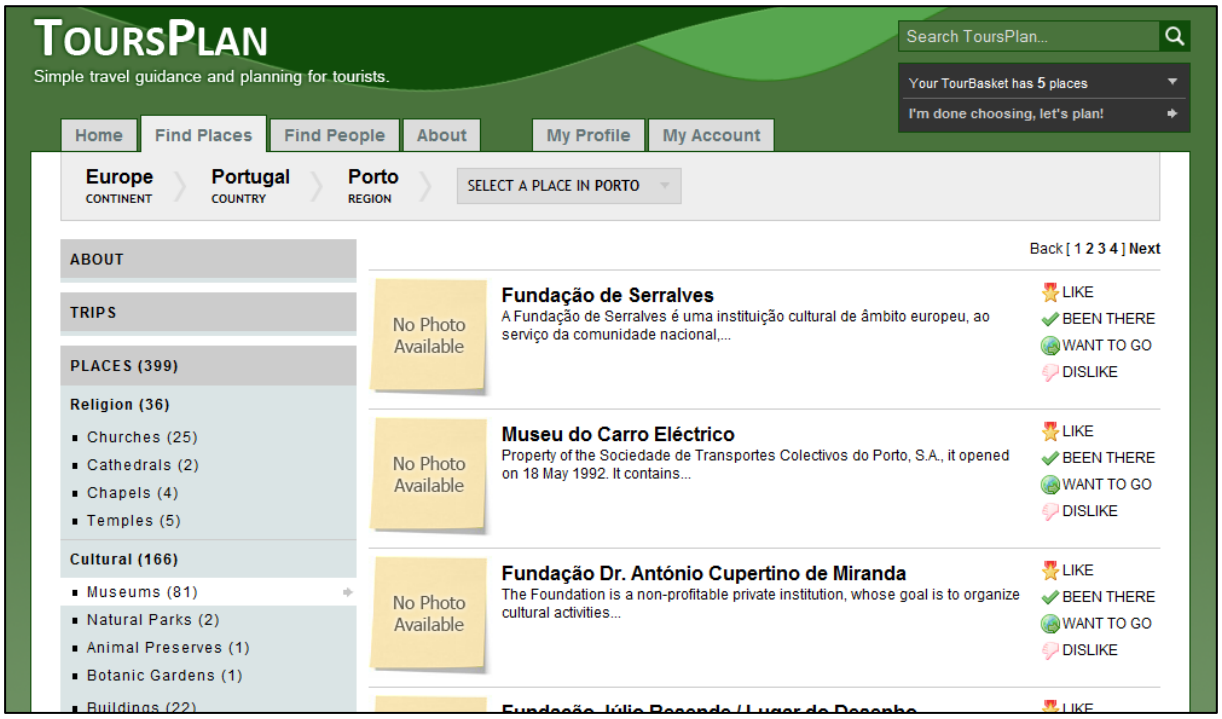

Figure 53 - Toursplan prototype user interface for an authenticated user: finding places and POIs.

A2.2. The SocioNet Web Application

To easily use the SocioNet library, a simple Java Web application was devised: the SocioNet Web application. Currently, it represents a back office application for Toursplan, allowing the dynamic creation of Virtual Graphs and execution of SNA algorithm sequences (see figures 54 and 55).

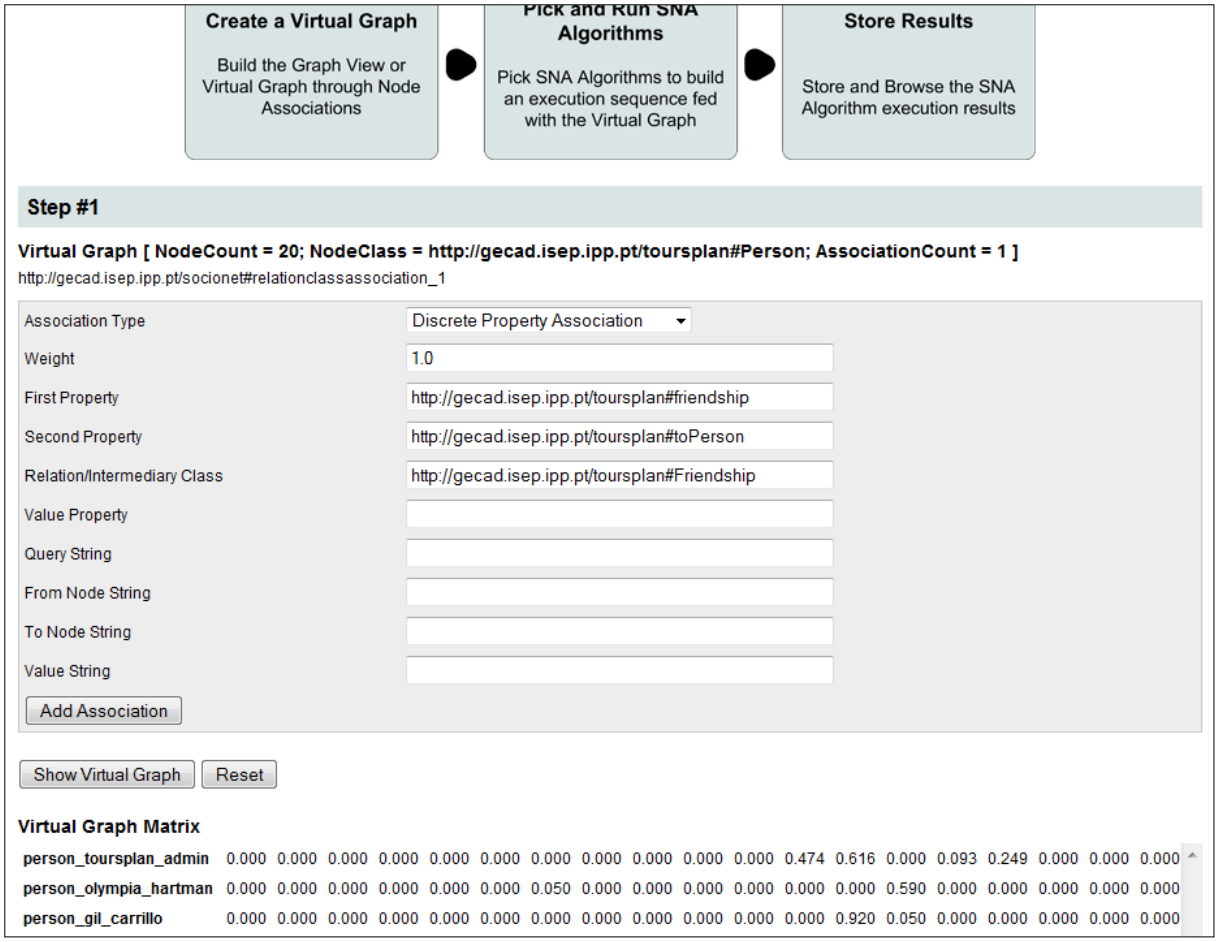

Figure 54 - Virtual Graph generation and SNA algorithm execution user interface (part 1).

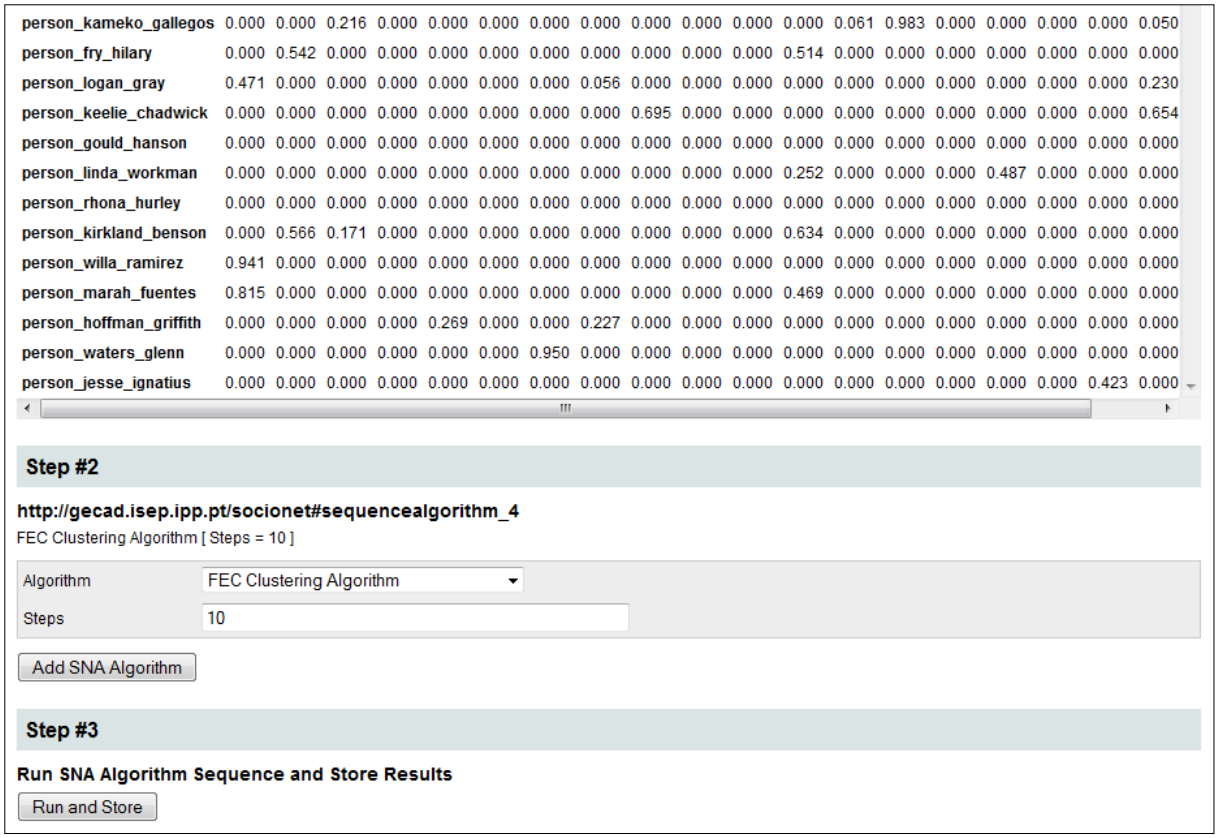

Figure 55 - Virtual Graph generation and SNA algorithm execution user interface (part 2).

Afterwards, the user can easily navigate through the source or sink models (see figures 56 and 57).

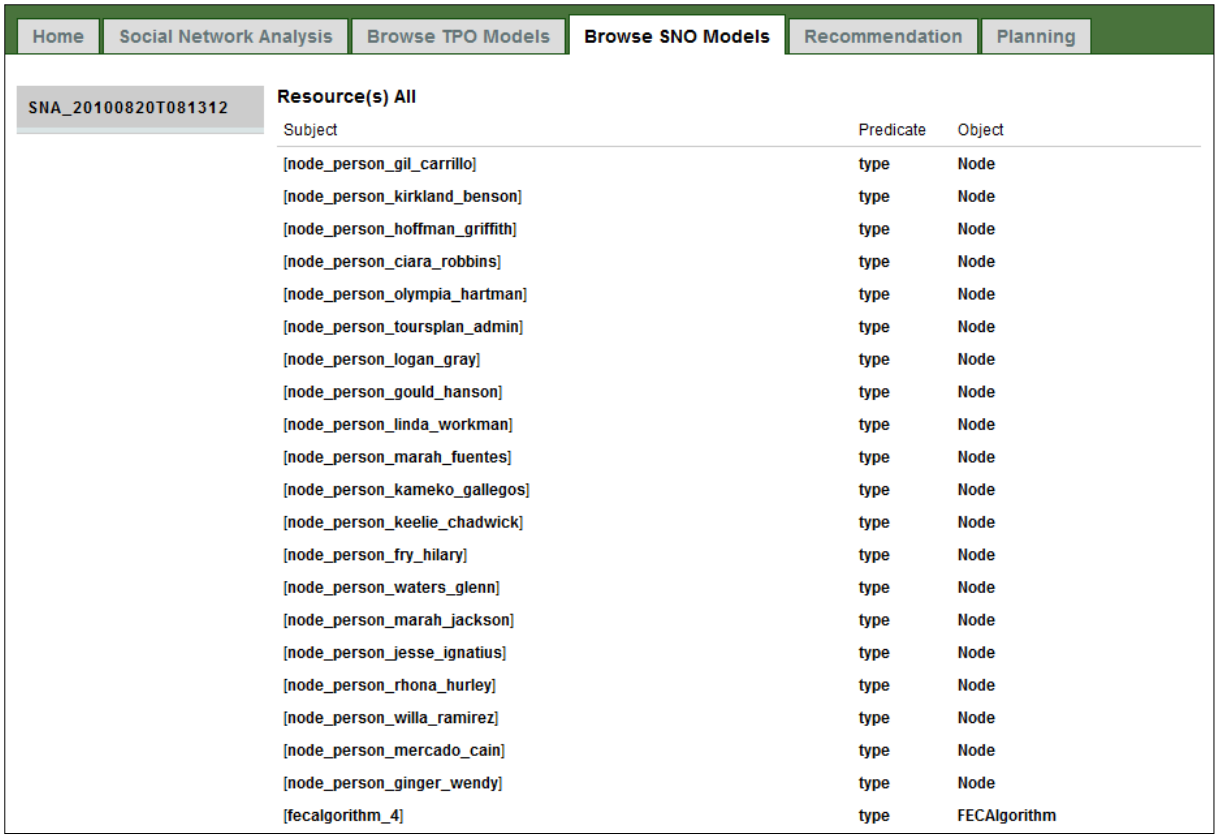

Figure 56 - Browsing a SNO Model after the SNA algorithm execution.

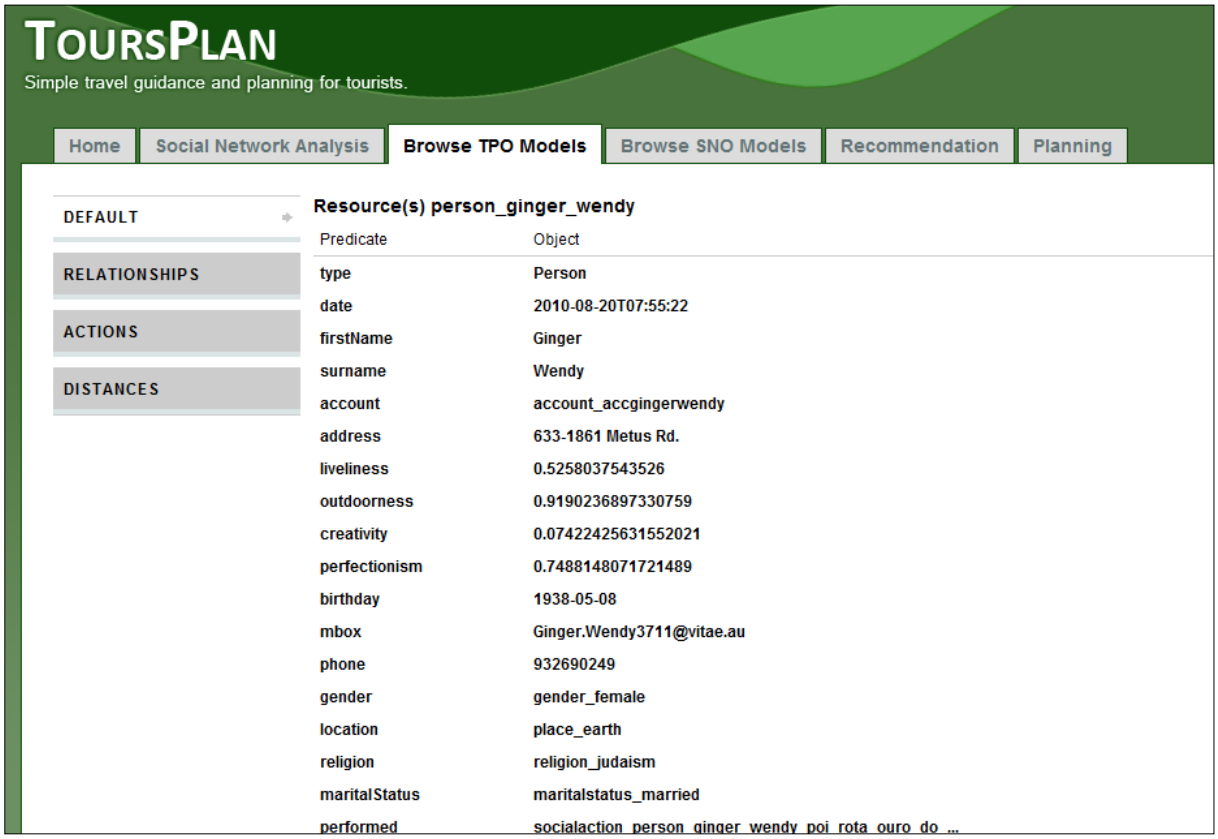

Figure 57 - Browsing the Toursplan main TPO Model.