2010

Discovery of knowledge collaborative communities for multi-domain problem solving

Shaojie Yuan
University of Wollongong

Recommended Citation
Yuan, Shaojie, Discovery of knowledge collaborative communities for multi-domain problem solving, Master by Research thesis, School of Computer Science and Software Engineering, Faculty of Informatics, University of Wollongong, 2010.
NOTE

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
Discovery of Knowledge Collaborative Communities for Multi-Domain Problem Solving

A thesis submitted in fulfillment of the requirements for the award of the degree

Master by Research

from

UNIVERSITY OF WOLLONGONG

by

Shaojie Yuan

School of Computer Science and Software Engineering

October 2010
Dedicated to

Xin Liu, Huiying Yuan and Linhao Shao
Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

__________________________
Shaojie Yuan
October 13, 2010
Abstract

The rapid technological advancement in the modern world has brought about a surge in the quantity of available data. There are vast potential knowledge and predictive pattern that exists within the abundance of data. However the technique to locate desired information and to identify useful patterns still poses a problem in today’s information age [38]. With the endowment of knowledge, many multi-domain problems arise, and can only be solved with diverse expertise. Hence the ability to obtain and utilise the knowledge within the myriad data to help solve multi-domain problems remains an important and challenging research issue.

This thesis aims to find an approach to discover a knowledge collaborative community to solve a multi-domain problem by study transaction data. In this thesis, firstly, a core-based node ranking approach is proposed, which could be used for an expert finding task; secondly, a Knowledge Collaborative Community (KCC) approach to discover a group of experts to efficiently solve a multi-domain problem in a small-size or medium-size network is proposed; thirdly, a two-step KCC approach used in large-scale networks to discover KCCs is presented.

The major contributions of this thesis are as follows.

1. A core-based node ranking approach used in event-based social network is proposed. The approach can rank nodes based on the importance and the activeness of the nodes in a network. The ranking result could vary based on different unit core (which demonstrates different demands). This approach could be deployed to rank experts.

2. A Knowledge Collaborative Community (KCC) approach is proposed, which could be used to discover KCCs in small or medium networks. Although some existing work proposed similar approaches, most of them considered only part of factors which might influence knowledge collaboration. In KCC approach, more factors, such as knowledge coverage, size of community, personal desires,
are considered. Furthermore, the knowledge level of an expert is introduced as an important factor which may impact the efficiency of knowledge collaboration.

3. A two-step KCC approach is proposed in this thesis to discover KCCs in large-scale networks. Compared with previous research, this approach has better performance in poorly connected and/or decentralized large-scale networks.
Studying abroad is an arduous and long journey. This thesis could not be completed without the support of a number of people. I would like to thank the following people:

I am indebted to appreciate my supervisor, Associate Professor Minjie Zhang. Her insightful comments and guidance was important for me to complete this thesis, and to make it an exciting experience. I am grateful to Minjie Zhang for her kind encouragement and help. I would also like to thank the School of Computer Science and Software Engineering, at the University of Wollongong for the financial support of my conference attendance.

I am delightful for Dr. Quan Bai’s enthusiasm for my research. Dr. Quan Bai often gives me useful comments and good papers related to my research. I thank Mr. Dayong Ye, who constantly had discussed with me in the lab, gave me comments on my papers to enrich my knowledge. My thanks are extended to Yikang Yang and Yuan Zhong, for their effort into correcting my English writing errors.

I would like to express my deepest gratitude to my parents, Linhao Shao and Huiying Yuan, and my wife, Xin Liu, who always show me their love, understanding, encouragement and financial support. This thesis could not have been completed without them.

My thanks are also extended to all the anonymous viewers of my papers, and all my other relatives and dear friends.
The following is a research paper which has been published during my master study.

Abstract v
Acknowledgements vii
Publications viii

1 Introduction 1
  1.1 General Process of Knowledge Discovery and Data Mining .......... 2
  1.2 Background of Knowledge Collaborative Community ................. 3
  1.3 Research Issues and the Solutions ........................................ 5
    1.3.1 Research Issues .................................................. 5
    1.3.2 The Solutions ................................................... 6
  1.4 Contributions and Organization of Thesis .............................. 7
    1.4.1 Contributions .................................................... 7
    1.4.2 Organization of Thesis ........................................... 8

2 Literature Review and Related Work 9
  2.1 Expert Finding .......................................................... 9
  2.2 Knowledge Collaboration ............................................... 14
  2.3 Community Discovery ................................................... 17
    2.3.1 General Approaches for Community Discovery ................. 17
    2.3.2 Approaches for Knowledge Collaborative Community Discovery 20
  2.4 Summary ........................................................................... 22

3 Core-Based Node Ranking in Event-Based Social Networks 23
  3.1 Event-Based Social Network Structure .................................. 24
    3.1.1 Definitions ......................................................... 25
    3.1.2 Rules to Construct an EBSN ...................................... 27
  3.2 Formulas ........................................................................... 29
3.2.1 Knowledge Discovery Based on Actor-Nodes ................. 29
3.2.2 Knowledge Discovery Based on Event-Nodes .................. 30
3.3 Case Study .................................................. 31
3.3.1 Conceiving the EBSN .................................. 31
3.3.2 Experimental Setup .................................. 32
3.3.3 Size of the EBSN .................................. 33
3.3.4 Actor-Node Ranking .................................. 33
3.4 Discussion ................................................. 36
3.5 Summary .................................................. 38

4 Discovery of Knowledge Collaborative Communities .......... 40
4.1 The Model Structure .................................. 41
  4.1.1 The Basic Layer .................................. 44
  4.1.2 The Actor Layer .................................. 46
  4.1.3 The Knowledge Layer .................................. 48
4.2 Algorithm ................................................. 49
  4.2.1 Evaluation of Knowledge Collaborative Community Based on
         Eight Factors .................................. 50
  4.2.2 An Approach for Discovery of Knowledge Collaborative Commu-
         nities .................................. 53
4.3 Discussion ................................................. 53
  4.3.1 Cost of Community Attention .......................... 53
  4.3.2 Fatigued Candidates .................................. 55
  4.3.3 The Probability of Catastrophe Phenomenon ................. 55
  4.3.4 The Knowledge Level of Candidates ......................... 55
4.4 Experiment ................................................. 56
  4.4.1 Data in Experiment .................................. 56
  4.4.2 Experimental Setup .................................. 57
  4.4.3 Experimental Results and Comparison ......................... 58
4.5 Summary .................................................. 61

5 Multi-Domain Problem Solving in Large-Scale Networks .... 62
5.1 Problem Description .................................. 63
5.2 A Two-step Approach .................................. 66
  5.2.1 Principle .................................. 66
  5.2.2 Step 1: Discovery of Potential Community in LSN ........... 67
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.3</td>
<td>Step 2: Discovery of KCC in PC</td>
<td>68</td>
</tr>
<tr>
<td>5.2.4</td>
<td>The Structure of the Approach</td>
<td>70</td>
</tr>
<tr>
<td>5.3</td>
<td>Experiment</td>
<td>71</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Experimental Setup</td>
<td>72</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Experimental Process and Results</td>
<td>72</td>
</tr>
<tr>
<td>5.4</td>
<td>Related Work and Comparison</td>
<td>75</td>
</tr>
<tr>
<td>5.5</td>
<td>Summary</td>
<td>76</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
<td>78</td>
</tr>
<tr>
<td>6.1</td>
<td>Major Contribution</td>
<td>78</td>
</tr>
<tr>
<td>6.1.1</td>
<td>A Core-Based Node Ranking Approach</td>
<td>78</td>
</tr>
<tr>
<td>6.1.2</td>
<td>An Approach to Discover A KCC</td>
<td>79</td>
</tr>
<tr>
<td>6.1.3</td>
<td>An Approach to Discover A KCC in Large-scale Networks</td>
<td>80</td>
</tr>
<tr>
<td>6.2</td>
<td>Limitations and Future Work</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>82</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Top Three Authors Based on Their Activeness .......................... 34
3.2 Top Three Authors Based on Their Importance ........................ 35
3.3 Top Three Authors Based on Their Core ............................... 35

4.1 Data of Actors ............................................................... 56
4.2 Data of Events ............................................................... 57
4.3 Average Size of Communities ........................................... 59
4.4 Average Knowledge Level ................................................ 59
4.5 Average Connective Level ................................................. 60
4.6 Average Number of Cultural Backgrounds ............................. 60
4.7 Average Number of Multi-Task Candidates ............................. 61
4.8 Average Collective Attention Cost ..................................... 61
## List of Figures

<table>
<thead>
<tr>
<th>Number</th>
<th>Figure Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>The Process of Knowledge Discovery</td>
<td>3</td>
</tr>
<tr>
<td>3.1</td>
<td>The Core</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Event-Based Social Network</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>The Numbers of Authors in DBLP in Different Periods</td>
<td>33</td>
</tr>
<tr>
<td>3.4</td>
<td>The Numbers of Publications in DBLP in Different Periods</td>
<td>34</td>
</tr>
<tr>
<td>3.5</td>
<td>Top Active Authors in Different Years</td>
<td>36</td>
</tr>
<tr>
<td>3.6</td>
<td>Top Important Authors in Different Years</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>The Model Structure</td>
<td>43</td>
</tr>
<tr>
<td>4.2</td>
<td>The Actor Layer</td>
<td>46</td>
</tr>
<tr>
<td>4.3</td>
<td>The Knowledge Layer</td>
<td>48</td>
</tr>
<tr>
<td>4.4</td>
<td>The Approach Used to Find $g_p$</td>
<td>54</td>
</tr>
<tr>
<td>4.5</td>
<td>The Network Structure in the Basic Layer</td>
<td>57</td>
</tr>
<tr>
<td>4.6</td>
<td>The Network Structure in the Actor Layer</td>
<td>58</td>
</tr>
<tr>
<td>5.1</td>
<td>The Two-step Approach</td>
<td>66</td>
</tr>
<tr>
<td>5.2</td>
<td>Initial PC</td>
<td>68</td>
</tr>
<tr>
<td>5.3</td>
<td>Neighbors of PC</td>
<td>68</td>
</tr>
<tr>
<td>5.4</td>
<td>Discovery of KCC in a Large-scale Network</td>
<td>71</td>
</tr>
<tr>
<td>5.5</td>
<td>The Performance in Different Size Networks</td>
<td>73</td>
</tr>
<tr>
<td>5.6</td>
<td>The Performance in Different Connective Networks</td>
<td>74</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In last decades, rapid developments in information industry has brought huge amount of data in both research and applications in many domains. Useful information and knowledge are hidden in such tremendous data sets. Knowledge discovery and data mining have attracted great attention in both information industry and information research area due to the need to discover useful information and knowledge from such huge amounts of data [27]. Many fields have the consensus that knowledge discovery and data mining are important. The various application areas of knowledge discovery and data mining include software engineering, marketing, finance, health care, geology, etc. [47] [27].

Transaction data describes events, and it is omnipresent in various domains such as science, engineering, business, logistics, etc [45]. Transaction data, for example, emails, articles, market order records and project records, contain uncovered patterns and knowledge. To discover knowledge from transaction data, knowledge discovery and data mining techniques are needed.

There are many knowledge fields (knowledge domains) in real life scenario, such as computer science, commerce, engineering and so on. Each knowledge field has its own standout experts. Continual development of knowledge and technology brings more complex problems and issues to the real world. One of new features of the complex problems is that it usually involves knowledge in many knowledge domains. Adler and Heckscher [1] pointed out that “Work is increasingly a matter of knowledgeable experts cooperating on projects in rapidly changing environments.” In order to solve such kind of problems, knowledge collaboration among experts is necessary. Knowledge collaboration is an intelligent activity, and takes place among a group of people. This group of people is considered to be one kind of a knowledge collaborative community. Simply speaking, a knowledge collaborative community is a group of experts combining their knowledge to solve a multi-domain problem. Finding collaborative communities is a key issue to the success of multi-domain problem solving.
1.1. General Process of Knowledge Discovery and Data Mining

The development of modern technologies allows people to store and access data at almost no cost [38]. The difficulty in current information-centric world is not how to store or access huge amounts of data but how to extract useful information, interesting trends, correlations and other knowledge from data [20]. To discover the hidden knowledge from data, data mining technologies are deployed. Data mining is the main phase of knowledge discovery process [47]. Data mining techniques are also employed in this thesis to give support for the discovery of a knowledge collaborative community. Figure 1.1 presents the process of knowledge discovery. There are mainly four steps in knowledge discovery process which are data preprocessing, data mining, pattern evaluation and knowledge presentation.

1. Data preprocessing: The tasks in this step include: data selection, data combination, data cleaning, data transformation, choosing data mining goal and data mining algorithm.

2. Data Mining: In this step, the chosen data mining algorithms are used to extract nuggets of knowledge from data (to achieve the data mining goal) [19] [31], for example, discovering the hidden relations among nodes in event-based social network, and finding the knowledge level of an expert.
1.2 Background of Knowledge Collaborative Community

Figure 1.1: The Process of Knowledge Discovery

3. Pattern evaluation: In this step, suitable measurements are deployed to evaluate the discovered knowledge.

4. Knowledge presentation: The discovered knowledge is presented to users in this step. If the knowledge is presented to a human user, then the techniques of visualization might also be adopted [11] [35].

This thesis involves the study of the first two steps of the process of knowledge discovery, which are data preprocessing and data mining. The algorithms and tools of these two steps are used in this thesis to discover a knowledge collaborative community. For example, node ranking in a social network [28] is deployed to discover the knowledge level of experts, and link mining techniques are used to analyse the relations among experts.

1.2 Background of Knowledge Collaborative Community

In this subsection, the discovery of knowledge collaborative community is introduced, including both the background and related work.

In early nineties, there was some research for enhancing user’s knowledge by knowledge collaboration and sharing [54]. From that time, the knowledge collaboration has attracted more and more attention. Currently, in rapidly changing environments, since knowledge is complicated and unlikely to be held by any expert individually, more and more projects and work need knowledgeable experts to cooperate together [1] [75]. In this situation, a new theory, called Dynamic Community, was proposed by Ye et al. [75]. The Dynamic Community was a knowledge collaborative community to initially
support knowledge collaboration among a group of experts in software development industries. The concept of Dynamic Community has been widely used in many domains for different purposes. A knowledge collaborative community is a group of experts combining and sharing their knowledge to solve a problem. The study of knowledge collaborative communities is necessary for discovery of knowledge collaborative community. The main research issues related to knowledge collaborative community include: the factors which may affect the efficiency and/or the performance of knowledge collaboration, and the cost of knowledge collaboration.

There are two forms of knowledge collaboration [72], which are knowledge exchange and knowledge combination. Knowledge exchange is the process that one person’s knowledge is transferred to others. Knowledge combination only takes place in social interaction and co-activity. This thesis focuses on the latter one.

Many factors could affect the efficiency of knowledge collaboration. For example, knowledge collaboration across culture could reduce the efficiency of knowledge collaboration, personal wills are essential to the efficiency of knowledge collaboration, and the number of experts participating in a knowledge collaboration could also affect the efficiency of knowledge collaboration [17].

Many approaches are developed to support knowledge collaboration with the consideration of some factors which may affect knowledge collaboration. For example, Kuriyama et al. [39] proposed a wearable system, called Social Context-Aware Communication System (SCACS), to enhance face-to-face communications or to build social networks to improve the efficiency of knowledge collaboration by using the preserved information in a virtual world. The SCACS could seek information related to experts who are communicating with the user of SCACS, and could present the information to the user.

Another system to support knowledge collaboration by visualization was developed by Ohira et al. [51]. In this system, a visualization tool, named Graphmania, to visualize the links among experts and projects was developed by using collaborative filter and social networks. This system could help the user to find suitable experts for knowledge collaboration.

The cost of knowledge collaboration should be counted in the discovery of knowledge collaborative community. In the process of knowledge collaboration, all participants consume attention in the communication with each other, and the consumed attention is called “Collective Attention Cost”. It was pointed out that the collective attention cost should be concerned to discover a knowledge collaborative community by Ye et
The catastrophe phenomenon may occur in a knowledge collaboration complexity network. There are three factors that may arouse the catastrophe phenomenon [77]:

1. Supplying too much knowledge;
2. Costing too much time and resource in knowledge collaboration path; and
3. Lacking of encouraging mechanism.

All three factors introduced above may cause a catastrophe phenomenon. These factors should be considered to avoid the discovery of knowledge collaborative community.

Most previous research only considered parts of the factors which might affect the efficiency and/or performance of knowledge collaboration. To discover a knowledge collaborative community which has good performance and could efficiently solve a multi-domain problem, considering all above factors is necessary.

1.3 Research Issues and the Solutions

1.3.1 Research Issues

The research issues regarding discovery of knowledge collaborative community for multi-domain problem solving include:

1. **Expert Finding.**

   The expert finding aims to find experts through huge amounts of data. To discover a knowledge collaborative community for solving a multi-domain problem, the suitable members (experts) of the community are discovered by mining transaction data. The basic requirement of a knowledge collaborative community is that the expertise held by members of this community should cover the required knowledge domains of the problem. To find experts with required expertise, the expert finding techniques should be deployed. The issue of expert finding can be separated into two smaller pieces:

   (a) **Expertise Discovery:** Some previous research for discovering the expertise of experts relied on self-assessment documents of the experts [7]. However in most cases, using self-assessment documents could not objectively discover the knowledge of an expert. A method is needed to discover the expertise of experts objectively.
(b) Experts Ranking: To discover a knowledge collaborative community, the discovered experts should be ranked based on different criteria. Most previous research ranked experts based on their importance or influence in the social network. However, these methods normally neglected the activeness of experts, which is an important factor for choosing members of a knowledge collaborative community. How to rank experts based on both their activeness and their importance is a challenging issue.

2. Discover Knowledge Collaborative Communities in Small-Size or Medium-Size Networks.

To discover a knowledge collaborative community in a small-size or medium-size network, more factors need to be considered, which may influence the success of knowledge collaboration or the efficiency of knowledge collaboration. Most previous research only considered part of the factors. Neglecting any factor could not discover the best knowledge collaborative community.

3. Discover Knowledge Collaborative Communities in Large-Scale Networks.

There are some differences between discovering a knowledge collaborative community in a large-scale network and discovering a knowledge collaborative community in a small-size or medium-size network. The first difference is that in a large-scale network it is not suitable to explore all nodes in the network to find experts. That is because that a large-scale network contains uncountable nodes, and exploring all nodes would consume huge computation resources. The second difference is that most large-scale networks are decentralized. In other words, the information of all nodes and their relations are not stored in one place (for instance, a server or a database). Choosing experts for a knowledge collaborative community through getting information of all nodes is not realizable. There are similar factors which need to be considered when experts are chosen for a knowledge collaborative community in a large-scale network and in a small-size or a medium-size network.

1.3.2 The Solutions

The solutions to solve the above issues are provided in the following chapters of this thesis.
1. The solution of Expert Finding:

(a) For Expertise Discovery: This thesis discovers the expertise of experts based on the published documents, which could avoid the drawback of using self-assessment documents. The expertise of experts is discovered based on the knowledge domains of events in which the experts participate.

(b) For Experts Ranking: In the thesis, a core-based node ranking approach is proposed in event-based social network which considers both the importance and the activeness of a node.

2. The Solution of Discovery of a Knowledge Collaborative Community in a Small-Size or a Medium-Size Network:

An approach considering all factors that may affect the knowledge collaboration is developed. Furthermore, the knowledge levels of experts are also considered to choose experts for knowledge collaborative community, which have never been considered in previous research.

3. The Solution of Discovery of a Knowledge Collaborative Community in a Large-Scale Network:

An approach to discover a knowledge collaborative community by considering all factors is introduced without exploring all nodes in a large-scale network. In this approach, relations among the questioners and their neighbors are analysed for the discovery of knowledge collaborative community.

1.4 Contributions and Organization of Thesis

1.4.1 Contributions

The major contributions of this thesis are as follows.

1. A core-based node ranking approach used in event-based social network is proposed. In this node ranking approach, the relationships among nodes are considered, and both the importance and activeness of nodes are taken into account in ranking results. Definitions of event-based social networks and rules to construct an event-based social network are proposed.
2. An approach to discover a knowledge collaborative community for multi-domain problem solving in a small-size or a medium-size network is proposed. All factors that could affect the knowledge collaboration are considered in this approach. The knowledge level of experts is also considered in the approach, which has not been mentioned in current literature.

3. An approach to discover a knowledge collaborative community to support for solving multi-domain problem in a large-scale network is developed. This approach can discover a knowledge collaborative community without exploring all nodes of a large-scale network. The experiments in Chapter 5 shows that the approach has good performance.

1.4.2 Organization of Thesis

The rest chapters are arranged as follows.

Chapter 2 reviews current literature related to this thesis, which includes expert finding, knowledge collaboration and community discovery.

Chapter 3 presents a core-based node ranking approach used in event-based social network, which could be deployed in expert finding tasks. This approach counts in both the importance and the activeness of nodes. An event-based social network model is introduced in this chapter.

Chapter 4 introduces an approach to discover a knowledge collaborative community in a small-size or a medium-size network. This approach considers all factors that may affect the knowledge collaboration to choose members of knowledge collaborative community, and introduces the knowledge level of experts which is a factor to be considered to choose experts for knowledge collaboration.

Chapter 5 introduces a two-step approach to discover a knowledge collaborative community in a large-scale network. This approach could discover a knowledge collaborative community without exploring all nodes in the network, which is essential in large-scale networks for discovery of knowledge collaborative community.

Chapter 6 concludes the thesis, and outlines the further research direction.
Chapter 2

Literature Review and Related Work

There are useful knowledge hidden in transaction data. Different knowledge may be discovered based on different demands. This thesis focuses on discovering knowledge collaborative communities which include experts in different knowledge domains for knowledge collaboration to efficiently solve multi-domain problems. The discovery of knowledge collaborative communities consists of three main parts, expert finding, knowledge collaboration and community discovery. This chapter explores in detail of the literature reviews in these three parts in three sections, respectively. Section 2.1 introduces the literature in expert finding. The related work of knowledge collaboration is reviewed in Section 2.2. In Section 2.3, the relevant community discovery approaches are briefly reviewed. Finally, this chapter is summarized in Section 2.4.

2.1 Expert Finding

The study of expert finding has attracted much attention. The goal of expert finding is to discover the expertise of experts and/or rank experts through technologies of social network analysis or other types of networks. The common method to discover the knowledge of experts is to analyse supporting documents, which are a type of sociable transaction data. To rank experts, the relations among experts and/or supporting documents could be analysed. In the social networks or other networks, node (expert) ranking algorithms might be deployed to rank experts.

Most previous research of expert finding relied on the self-assessment documents of a person to decide the knowledge domain of this person. However, self-assessment documents do not always objectively reflect the fact. Becerra-Fernandez introduced an expert finding system, called People-Finder [7] [8], to find experts who possessed skills in a specific knowledge domain. In People-Finder, the published documents had heavier weight than self-assessment documents to discover the knowledge domain of an expert. The People-Finder could also rank experts based on the published documents.
2.1. Expert Finding

To locate expertise of an expert, the Term Frequency Inverse Document Frequency (TFIDE) algorithm was used in the system [8]. The advantage of this system is that it mainly used the published documents to discover the expertise of an expert and to rank experts instead of self-assessment documents. However, the relations among experts and questioners were overlooked. The time of publications was neglected as well. This drawback may cause a problem. For example, an expert published many documents some years ago, but he did not publish any document in recent few years. The system could still recognize the expert described above as the one who has a high rank score even he is not currently active at all.

A novel hybrid approach to find experts in question-answering websites was proposed by Kao et al. [34]. The approach used a query-based method, and the discovered expert could answer a specific question. The subject relevance, user reputation and the authority of experts were all considered in the approach. The influence of experts were counted in the criteria for choosing experts, but the relations between experts were not taken into account. Since this approach focused on question-answering websites, the relations between questioners and answerers were essential. The relations between experts could be analysed to discover the suitable experts.

An important application domain of expert finding is Software Engineering. Renuka Sindhgatta presented an approach to identify domain expertise of software developers through programming codes [62]. In her approach, the programming codes were used as supporting documents, and the K-Mean cluster method was used to cluster documents into different key concepts. Each concept represented a knowledge domain. The expertise of each developer could be calculated based on the occurrence of the key concept in his/her documents. The advantages of the approach include: 1) the consideration of the time period for each document and the activeness of a developer could discover active experts; and 2) the keywords are clustered into key concepts. There are also two drawbacks of the approach. (1) Since the approach employed the K-Mean method, different values of \( K \) could significantly impact the result. (2) Each document was treated equally without consideration of the ranking of importance.

Mattox et al. introduced MITRE’s ExpertFinder system through ranking methods for expert finding [46]. In MITRE’s ExpertFinder, experts could be located by end user through entering keywords. The experts were then ranked based on the number of keywords mentioned in their related documents (i.e. resumes, publications and newsletters). The rank value of an expert could increase if experts add more keywords into their documents; and the rank result could easily be affected by the ranked experts.
through changing the number of occurrences of keywords in their documents.

There are many models deployed in expert finding. The language model is one of the most popular models. Balog et al. proposed a language model to discover the expertise of experts in a specific knowledge domain through supporting documents by a given topic [4][5]. Petkova and Croft proposed another language model which focused on the relationship between an expert name and the content of the supporting documents [55]. All above language models neglected the activeness of experts. These methods might choose experts who were active some years ago, but are not active currently.

The keyword matching methods are basic methods to discover the expertise of experts. Many approaches have been developed based on keyword matching. An approach to automatically find experts presented by Ru et al. [58] is an example of approaches deploying keyword matching. The goal of their approach was to find experts who had requested expertise. In their approach, a probabilistic cascading framework was proposed, a set of experience measures were used to discover the expertise of an expert, and the expertise of each expert was calculated in advance. After a query was given, the keywords in the query were used to match the expertise of each expert, and experts were ranked based on the keywords matching result. The advantage of their approach is that the expertise of each expert was pre-calculated, and after a query was given, the system could quickly produce a result. The disadvantages of the approach include that: 1) keywords were used as the only criterion for ranking experts; 2) the publishing time of supporting documents and the topics were neglected; and 3) the relationship between a questioner and an expert were not considered.

Some researchers developed models to improve keyword matching methods. For instance, a model was proposed by Wu et al. [71], called EM2, to exploit semantic information of metadata corpus to discover experts of a query. EM2 was built based on the Word-Topic-Document association. Experts were ranked depending on not only keywords but also topics. EM2 used the concept of topic, and improved the previous keyword-document association to the word-topic-document association. There are two shortcomings of their model. (1) The model did not consider the activeness of experts, and depends only on the number of occurrences of keywords, topics and experts, without the consideration of the publishing time of documents. (2) The relations between users and experts were neglected.

Another system, called ArnetMiner, which used the author-conference-topic model to find an expert in a specific topic area, was introduced by Tang et al. [63][79].
2.1. Expert Finding

The system could find experts, extract a researcher’s profile from webs, search for an author, search for publications and find out people’s associations. In other words, this system could discover the expertise of experts. In ArnetMiner, for each publication, an author was chosen from “a uniform distribution”. A topic $z$ was chosen from “a mixture weight of the chosen author and a distribution from a symmetric Dirichlet prior”. Then a word and a conference stamp were generated from the chosen topic $z$. The above process could model the dependencies between different kinds of nodes (authors and topics). A random walk model was deployed in the network, and a combined score for each node was returned to the query in the system. The advantage of the system is that ArnetMiner could automatically get information from web by using ontology techniques [32] to build a knowledge base. The disadvantages include that: 1) the number of topics needed to be specific in a query; and 2) the activeness was overlooked in ArnetMiner.

Some approaches were developed to rank experts based on the relations among experts. For example, Fiaidhi and Mohammed proposed an expert ranking approach to rank experts in Digital Bibliography & Library Project (DBLP) [22]. In their approach, the published articles were ranked first, and then the experts were ranked based on the publication ranking results. The expert collaboration score was the rank score for each expert. Actually, this approach ranks experts based on their links. In their system, the attributes of authors (such as cultural background) had been neglected and the activeness of experts were not considered.

There is another expert finding algorithm based on the connection among experts. A social-network-based expertise propagation algorithm was introduced by Fu et al. [26]. In their algorithm, firstly a social network using emails or web pages was built. Secondly some top ranked experts were chosen as seeds to discover potential experts. Thirdly, the probability of a person to be an expert was calculated based on the number of seeds this person connected with. If this person had many connections with seeds, then this person had large probability to be an expert. Their algorithm works like the PageRank algorithm [13] [60]. The number of connections determined the rank value of an expert.

Madadhain and Smyth defined a model, named EventRank for node ranking in event-based social network [52] [53]. This approach ranked nodes based on their links. A matrix in their model was used to represent a social network. In EventRank, the rank value of a node in time $t_{i+1}$ was calculated based on the rank value of time $t$. The rank value (where the rank value of each node was a vector) in time $t_{i+1}$ equaled to that
2.1. Expert Finding

the rank value in time $t_i$ multiplied the network matrix $M_i$ where $M_i$ represented the events effect at time $t_i$. Their model could catch the dynamic features of the network, and the evolution of the node ranking in the network could also be captured. Since they used a matrix to represent a network, if the number of nodes in the network was very large, the computation cost could be very high. The EventRank ranked nodes only based on the connections between nodes, but the attributes of nodes were not counted in.

A system called Expertise Oriented Search (EOS) was proposed by Li et al. [42]. EOS ranked experts not only based on the relationship among experts, but also the relationship among experts and knowledge. In their system, co-authorship was employed as the relationship among researchers to build a social network. They developed an aggregate social network to find experts and important nodes in specific topic areas. Their system was a relevance propagation-based system, and contained two steps. At the first step, the relevancy of a candidate to a topic was calculated. In the second step, the topic relevancy of a candidate to other related candidates was propagated by a propagation-based algorithm [3]. The advantage of their system is that both the attributes and links were considered. The shortcoming of their system is that their model could not catch the dynamic features of social networks.

There is a novel expert finding approach which uses strength measurement to rank experts in a network. The name of the strength measurement was integrated cohesion, which was proposed by Bar-Yossef et al. [6]. This measurement was suitable to be used in arbitrary weighted networks, i.e. email networks. The limitations of this approach are that: 1) the approach was suitable for explicitly linked networks; and 2) this approach neglected the activeness of nodes.

According to the survey of current related work, most of existing approaches considered only connections or attributes to rank experts. A few of them considered both of the connections and the attributes of experts for experts ranking. Most previous research also neglected activeness of an expert in the knowledge domain and ranked experts based on the aggregated data, which could not catch the dynamic feature of experts. In this thesis, experts (nodes) are ranked based on links, attributes, activeness and the importance of the nodes in the network. The details of the solutions will be introduced in Chapter 3.
2.2 Knowledge Collaboration

To find knowledge collaborative communities for solving multi-domain problems through knowledge collaboration among experts, investigation and study in knowledge collaboration are necessary. Many researchers have been working on knowledge collaboration in recent years based on different perspectives. Some works focused on the platforms of knowledge collaboration, e.g. Wikipedia [44], some approaches studied about the cost of knowledge collaboration, for example, cost of collective attention [74] in knowledge collaboration, some methods analysed factors which could affect the efficiency and/or performance of knowledge collaboration, such as cultural background [17], and some methods presented the application domains of knowledge collaboration. In this section, a number of approaches related to knowledge collaboration for different purposes are reviewed.

Blogosphere is widely used in the Internet to post news of events and personal online diaries. People could leave comments in a blogosphere. Blogosphere could be a typical platform to support knowledge collaboration [21]. The Blogosphere represents an “intellectual cyberspace”. All candidates could publish their thoughts and ideas on the platform. The blog owner could control the contents of the blog. The advantage of Blogosphere is that it can support many to many collaboration. However Blogosphere can bring some problems, e.g. the owner of a blog may delete a right answer to a posted question since he/she may not be the expert in the knowledge domain of the question. In this platform, it is hard to know whether the content was written by an expert or by a person who is lack of the knowledge in this field.

Many approaches were developed based on the application domains of knowledge collaboration. Horaguchi proposed an interesting view that the marketing is a kind of knowledge in economics [30]. Horaguchi believed that the knowledge was created with cost but could be transferred without additional cost. Knowledge collaboration is important in economics to reduce the marketing cost. Collaboration in knowledge was defined as “a process of connecting two markets to make a single market” in economics by Horaguchi [30]. The models of knowledge collaboration, e.g. how to connect markets in economics area, were studied in his research. Based on his research, different patterns of knowledge collaboration had different affections of the production level. Choosing a suitable pattern of knowledge collaboration could increase the production level. The research of Horaguchi brought the knowledge collaboration into a new application domain, marketing.

Software development is another application domain of knowledge collaboration,
since software development is a knowledge-intensive process [76]. Ye et al. pointed out that the knowledge collaboration played an important role in the software industry, and defined the knowledge collaboration was defined as “utilising knowledge repository systems as well as enlisting the help of peer developers to acquire the needed knowledge that the developer does not have yet” [73]. The process of knowledge collaboration was defined by Ye et al. as an one-to-many interactivity. If a developer does not have the knowledge elements to accomplish a task but could get through knowledge collaboration with other developers, the idea of knowledge collaboration could be really helpful. In practical view of knowledge collaboration in a software development industry, the process of knowledge collaboration should be a many-to-many interactivity, but not an one-to-many interactivity, since all developers involved in a task should communicate with others to accomplish the task.

The collective attention cost is one kind of costs brought by knowledge collaboration. A notation of “collective attention cost” was proposed by Ye et al.. It denotes all the attention consumed by all experts involved in the process of knowledge collaboration [74]. It was assumed that there were two parties in the knowledge collaboration. One party was a questioner who needed help from others, and the other party included recipients who may give their help to the asker. The Cost of Collective Attention (CoCA) included (1) the attention consumed by an asker to find who has the expertise that the asker required, to formally describe and present the question, and to evaluate the quality of all answers provided by recipients; (2) the attention consumed by recipients who are interrupted by the asker to attend the event of interruption and the resumption of their current work, and to make a decision of whether responds the question or not; and (3) the attention consumed by recipients who have decided to answer the question to compose an answer to the question. It can be seen from the study of CoCA that the CoCA could be significantly reduced if the total number of recipients could be reduced.

A number of factors may affect the knowledge collaboration, but most research only considers one or part of them. Some approaches only consider the connection and expertise in knowledge collaboration. For instance, the Dynamic Mailing List approach (DML), proposed by Ye et al. [74], is one of approaches to help an asker to find the suitable expert for knowledge collaboration, using social relations between experts, and expertise of experts as the criterion for choosing recipients. DML could reduce the total number of recipients so as to reduce the CoCA, but other factors which may affect the efficiency of knowledge collaboration had been neglected, such as the cultural
2.2. Knowledge Collaboration

Another approach which only considered the connection and expertise of experts is Graphmania. Graphmania is a tool which was proposed by Ohira et al. [50] to help knowledge collaboration in a software industry. The relationship among developers and projects could be visualized using the tool, and a user could find answers of questions such as “Who could answer my question?” and “What question could this person answer?”. The advantage of Graphmania is that all users could explore all relationships among the developers and the projects. The disadvantage of Graphmania is that all users had the same visualization of networks. Graphmania could be improved by only showing useful information required by the current user, which means different users could get different visualizations.

The method of communication is an important factor in knowledge collaboration. A macro-level framework, named knowledge communities, was presented by Upham et al. to organize a large-scale innovation network [65]. In their work, the importance of communication in the knowledge collaboration procedure was addressed. They argued that different knowledge patterns could also affect the performance of knowledge collaboration. The knowledge communities indicate that methods of communication and the presentation of knowledge are important.

Another important factor of knowledge collaboration is the size of knowledge collaborative community. The size of knowledge collaborative community could affect the frequency of knowledge collaboration. Kakimoto et al. did detail investigation to analyse the relation between the size of community and the frequency of knowledge collaboration, and used social network analysis tools to analyse four communities of the open source software communities [33]. Their research result showed that the knowledge collaboration occurred more often in a small-size community than a big-size community.

Ding and Huang used the Stackelberg leader-follower framework to analyse the benefits and drawbacks of organization’s knowledge collaboration [18]. Through their study, the knowledge sharing in knowledge collaboration could lead to a dilemma, knowledge spillover; but knowledge collaboration could also bring benefit to the participants. They believed that to control the ratio of influence of current knowledge creation to prior knowledge created could help organizations to gain benefits from knowledge collaboration and to limit the dilemma of knowledge spillover.

From the above depiction, it is easily to be summarized that many factors could affect knowledge collaboration. Most research only considers one or limited number of
factors to support knowledge collaboration. In this thesis, more factors to find suitable experts for knowledge collaboration are considered.

2.3 Community Discovery

In this thesis, community discovery is an essential step to discover knowledge collaborative communities. Several algorithms and approaches related to community discovery are reviewed in this section. General approaches for community discovery are reviewed in Subsection 2.3.1. Approaches for communities discovered to support knowledge collaboration are introduced in Subsection 2.3.2.

2.3.1 General Approaches for Community Discovery

The Kernighan-Lin algorithm was proposed by Kernighan and Lin to discover communities based on the network structure [36]. In Kernighan-Lin algorithm, a benefit function $Q$ was defined as the number of edges within the communities minus the number of edges between the two communities. The Kernighan-Lin algorithm assumed that there were only two communities in a network, and all nodes in a network could only belong to one of the two communities. A pair of nodes (where each node in the pair was from one of the two communities) were chosen to swap. If the swap could increase the value of benefit function $Q$, then performed the swap, otherwise another pair of nodes would be chosen to swap. After swapping all possible pairs of nodes in the network, the two communities could be discovered. The disadvantages of Kernighan-Lin algorithm are that: 1) the sizes of the two communities should be specified for discovering the communities; and 2) the algorithm could only be used in networks which only contain two communities.

Scott proposed a hierarchical clustering approach by the analysis of network structure [61]. The goal of this approach was to discover communities in social networks. This approach has been widely used in sociological science. In this approach, a measure of similarity between pairs of nodes was deployed based on the structure of a network. Edges were added in the empty network, which only contained nodes but no edges initially, from nodes with highest similarity to nodes with lowest similarity. Note that the similarity measure was calculated based on the original network; the added edges were independent of the original edges. The advantage of the approach is that the sizes of communities do not need to be specified before starting to discover communities. The disadvantage of this approach is that the investigator needs to specify the stop
2.3. Community Discovery

There are many measurement methods which could be used in hierarchical clustering approaches. Following are three examples:

1. Euclidean distance proposed by Ronald S. Burt [14],
2. The pearson correlation measurement introduced by Wasserman and Faust [66], and
3. Count of paths between nodes presented by Ahuja et al. [2].

There is another type of measurement method applied in community discovery, named betweenness measurement. There are three examples of this type of measurement method:

1. the shortest-path betweenness, proposed by Freeman [25],
2. the random walk betweenness, proposed by Newman and Girvan [49], and
3. the current-flow betweenness, proposed by Newman and Girvan [49].

These measurement methods were used in the remove-recalculate algorithm presented by Newman and Girvan to discover communities in social networks based on the structure of the networks [49]. A divisive technique was used in the algorithm, which could associate with a betweenness measurement method (the shortest-path, the random walk or the current-flow) to remove edges and split the network into communities. The main steps of the remove-recalculate algorithm included:

1. removing the edge with the highest betweenness score in the network,
2. recalculating the betweenness score of all remaining edges, and
3. going back to Step 1 until the communities were discovered.

This method had good performance in discovering communities based on the structure of network. The disadvantages of this method are that: 1) it had a huge computation cost, and 2) the stop point needed to be specified.

Newman proposed modularity measurement method for community discoveries [48]. The modularity was used as a good indicator for community discovery in a network. In this method, the positive modularity meant a good division of network. By contrast, a bad division of network could cause a negative modularity. Newman’s method
has become one of the most popular methods for community discovery. The links among nodes were used for community discovery in this method. The advantage of this method is that it did not need to know the number and the size of communities. The disadvantage of this method is that it is not sensitive for small community discovery [24].

The closeness is another measurement method applied to community discovery. An approach was proposed by Zhdanova et al. to model a network of web community through considering the closeness among nodes in general networks [81]. Their approach could represent and calculate the closeness based on the individual online information in a particular community. If two nodes had connected by many links, they could get a high closeness score, and they had high probability belonging to the same community. In their approach, the closeness of two nodes depended on only the links between two nodes excluding attributes (for example, personal interest). In their approach, the data needed to be updated by the candidate himself/herself. If the updated information is inaccuracy, it might lead to a wrong closeness score. Since it would lead to a wrong closeness value, the communities discovered might be inaccurate.

Wennerberg developed an interesting system deploying visualization method to discover communities. It is an ontology-based system that included information about people, organizations, locations and events to represent a social network [68]. The goal of their system was to help discovering links among the entities that might not be obvious at the first glance, and to discover hidden communities. The knowledge base of the system was built by ontologies which could be reused and machine-readable. The system could track security issues on a semantic web. Nevertheless, Wennerberg’s system could not catch dynamic characteristics of a network because they aggregated all information into a static social network. Braha and Bar-yam indicated that aggregating all information into a static network could not capture the dynamic features of the social network [12].

Wu and Huberman developed a resistor network algorithm [70]. The algorithm treated a network as an electrical circuit to discover communities. At the beginning, a unit resistor was placed into each edge of the network. Then, two nodes were randomly chosen from different communities to apply a unit of potential difference. Finally, other nodes could be classified into these two communities based on the voltage of the nodes. The advantage of this method is that this algorithm could discover a community which includes a specified node without identifying all communities. The disadvantage of this algorithm is that it could only split the network into two communities.
### 2.3.2 Approaches for Knowledge Collaborative Community Discovery

This thesis focuses on the discovery of knowledge collaboration communities. This subsection gives reviews of several major approaches to discover knowledge collaborative communities.

A framework of “dynamic community” was proposed by Ye et al. initially to find “dynamic community” in software development industries for knowledge collaboration [75]. In their research, a dynamic community was a dynamic group formed to support knowledge collaboration. The framework chose members of a dynamic community based on the links of experts and knowledge relations. All experts who fit both the following two criteria were gathered to be a dynamic community:

- The experts had connections among them.
- The experts had required expertise or related knowledge.

The advantage of their framework is that the relationships among collaborators, which could significantly affect the efficiency of knowledge collaboration, become an important criterion for choosing collaborators. The framework might gather a large dynamic community since all experts with required expertise and connections with the questioner would all become members of a dynamic community. Because the size of community was not considered in this framework, the efficiency of knowledge collaboration might reduce if the size of a community was large. Knowledge collaboration would cost much more time in a large-scale community than that in a small-size one.

Ohira et al. introduced a “Dynamic Social Networking System” [50]. Their system could be used to support knowledge collaboration in software development industries. In this system, a scale-free network was constructed by connecting developers who participate projects together and by connecting related projects. This system could help developers who had few connections to build more connections, and made the network with more links. The advantage of this system is that it could address the poor performance of knowledge collaboration in poorly connected networks. The shortcoming of this system is that the system might gather a large-size community which would cost more resource for knowledge collaboration.

A novel approach was proposed by Zhang et al. [80] in 2009. In their approach, an interactive network is constructed by using log files. The composite web services communities could be discovered by using spectrum clustering for knowledge collaboration. Visualization of log files through the network could get a deep insight of the
2.3. Community Discovery

log files. Members in a same community discovered by the approach might provide similar services, i.e. they might have similar knowledge domains. If the community had a multi-domain problem to solve, it had highly probability that could not solve the multi-domain problem because members of the community might provide the similar services from the similar knowledge domain but a multi-domain problem required expertise in different knowledge domains.

An approach was proposed by Lappas et al. to discover a team of knowledgeable individuals to solve a specific problem [40]. Their approach, called TEAM FORMATION, aimed at discovering a team, which had minimum cost of communication among the team members. This approach used graph theory to discover a team for solving a specific problem. Each node in graph indicated an expert, and the edges were weighted and undirected. The weight of each edge presented the cost of communication between the two nodes connected by the edge. Each node in their approach was assigned a skill. The TEAM FORMATION discovered a team with the skills required by a specific problem, and the communication cost among members of the team was minimum. There are a number of factors which may influent the knowledge collaboration. The TEAM FORMATION considered the expertise (skill) of experts and the communication cost, but neglected other important factors, such as knowledge level, personal desire, multi-tasked experts, etc.. The approach could only be used in a centralized network, where the information could be accessed through a server or database. This approach did not work in a large-scale network or a decentralized network.

An approach was proposed by Dos Santos et al. to discover potential collaborations among nodes in a social network [59]. In their approach, the information was grouped into three sets: the individual set, the document set and the term set. There were three kinds of networks in their approach which were defined from the three object sets. The first network was the collaboration network, where each node represented a candidate and two nodes would have a link if they were connected by the same document. The second network was the document network, where each node represented a document, links among documents represented the high similarities among the documents. The similarity of two documents was determined by the number of same terms appearing in these two documents. The third network was generated by the combination of the first two networks, where each node represented a candidate and the links among nodes indicated that the candidates had the same interest. This approach built connections among individuals through their interests, and could discover communities which contain the candidates who had the similar interest. The advantage of this approach is
that both the attributes and relations among candidates are used for community discovery. The disadvantage is that the approach might have high computation cost in a large-scale network since it deployed a recursive algorithm.

A framework to form a team in an organization for solving a multi-objective project was proposed by Wi et al. [69]. In this framework, experts were chosen for a specific project based on their knowledge competence score. The knowledge competence score was impacted by two parts, knowledge part and social network part. The knowledge part was based on whether the expertise was required by the project, and the social network part was evaluated by how close the individual was connected with other team members. Like most previous approaches, this framework only considered part of factors which might affect the knowledge collaboration, e.g. knowledge coverage and connective level, but overlooked some important factors, such as multi-task experts and personal desires.

In this thesis, communities are discovered for knowledge collaboration based on both the network structure and attributes of experts. Compared with the traditional community discovery methods [70] [23] [57] [36] [61] [48], the approach presented in this thesis discovers communities for the specific purpose, knowledge collaboration. Unlike [75] [50], the approach could get a smaller community which could cost less resource for knowledge collaboration. Against the approaches presented by Zhang et al. [80] and Santos et al. [59], the approach discovers communities which contain experts in different knowledge domains to solve multi-domain problems. In addition, the approach could discover experts with required expertise who are not directly connected to the questioner.

2.4 Summary

This chapter reviewed research related to the research issues described in Chapter 1. Firstly, approaches of expert finding was reviewed in Section 2.1. Secondly, knowledge collaboration was discussed in Section 2.2, which included platforms of knowledge collaboration, application domains of knowledge collaboration, the patterns of knowledge collaboration, the cost of knowledge collaboration and so on. Thirdly, various approaches of community discovery were elaborated in Section 2.3, which included general approaches for community discovery reviewed in Subsection 2.3.1 and major community discovery approaches for knowledge collaboration introduced in Subsection 2.3.2.
Core-Based Node Ranking in Event-Based Social Networks

Most previous expert ranking algorithms for event-based social networks only considered the number of events an expert (node) participates in. However, in event-based social networks, the influences of events should also be considered when experts are ranked. That is because different events may have different influences. Event-based social networks are a category of dynamic networks. All event-based social networks have a common feature that the events in a network are immanently temporal. In other words, all events have a time stamp to indicate when the events have happened [52]. Furthermore, there might be more than one expert which participate in one event, and one expert might participate in more than one event.

Different from approaches introduced in Chapter 2 Section 2.1, this chapter introduces a generic approach to rank nodes (experts) in event-based social networks. This approach can rank not only experts, but also general nodes (i.e. an email address, a telephone number, and a bank account.) in event-based social networks. Firstly, a social network is built using events (i.e. emails, publications and other supporting documents). Different events have different influence values. Events which could affect other events have large influence value. Each event has a time tag to indicate when the event has happened. For nodes in the event-based social network, two important attributes are generated for them: activeness and importance. The activeness of a node shows how active this node is in the network. The activeness of a node could be large if this node participates many events. The importance of a node presents how important this node is in a network. The importance of a node is related to the influence of events this node participates. Secondly, the concept of core is introduced. The core is a vector which tightly depends on activeness and importance. The nodes of a network are ranked based on queries. Each query contains a unit of core vector indicating that whether more important nodes could have higher rank value or more active nodes have higher rank value. The core of each node is normalized based on
the unit core provided by a query. It is noticed that query relevant ranking algorithms are needed in many application domains. For instance, in a disease control system, the more active nodes should be located to control the spread of disease. However, to locate the source of a disease, nodes which have larger importance value should be discovered. In short, this approach is query relevant (compared with the PageRank algorithm [13]); and the rank result depends on two attributes of nodes, which are derived from both the attribute of event and the relations between nodes and events, and can vary in different application domains (compared with the HITS algorithm [37]); the rank result could be hardly impacted by nodes but easily impacted by enquirers (Compared with the MITRE’s ExpertFinder system [46]); both links and influence of events could affect the rank result (compared with the expertise propagation algorithm [26]).

This chapter is arranged as follows. Formal definitions and network construction rules for event-based social networks are proposed in Section 3.1. In Section 3.2, formulas for evaluating the importance, activeness and core of actor-node, and influence of event-node are introduced. A case study is presented in Section 3.3. In the case study, the DBLP data set is used as an example to demonstrate the use of the proposed approach. In Section 3.4, a discussion is presented. Finally, the chapter is summarized in Section 3.5.

3.1 Event-Based Social Network Structure

In this section, an assumption is put forward that all network changes are caused by events in an Event-Based Social Network (EBSN). An EBSN is a network structure which contains actors (nodes), events, relationships among events, and relationships between actors and events. An EBSN can represent the real world scenarios. Each actor in an EBSN may have a new relationship when an event happens. For example, Mark is Marry’s husband after their marriage. The relationship between the two actors (i.e. Mark and Mary) is established after the event (i.e. get married) happens. In an EBSN, an event could create a (set of) new relationship(s) or change a (set of) relationship(s). Formal definitions of EBSN, actor-nodes, event-nodes, link between Actor-node and Event-node (lbAEs), link between Event-node and Event-node (lbEEs) and core are given in Subsection 3.1.1.
3.1.1 Definitions

Definition 3.1: An Event-Based Social Network (EBSN) is a bipartite directed graph. It is defined by a two tuple \( EBSN = (N, C) \), where

- \( N = \{n_1, n_2, ..., n_s\} \) is a finite node set which contains all vertices in the network.
- \( C = \{c_1, c_2, ..., c_r\} \) is a finite arc set which contains all arcs in the network, where \( c_i = (n_j, n_k) \) with \( n_j, n_k \in N \).

An EBSN is a bipartite graph as there are two types of nodes in the graph, i.e. actor-nodes and event-nodes.

Definition 3.2: An actor-node \( a_m \) is a node which represents an actor in EBSN. It can be defined by a three-tuple, \( a_m = (ID, activeness, importance) \), where

- \( ID \) is a unique string to identify different actor-nodes.
- \( activeness \) shows how active an actor-node is in a particular domain. It is defined by a four-tuple, \( activeness = (value, domain, time_0, time_1) \), where
  - \( value \) is a non-negative number which indicates the value of activeness.
  - \( domain \) is a string that presents a specified field in which the activeness is.
  - \( time_0 \) and \( time_1 \) are time tags which show the activeness of an actor-node in the period from \( time_0 \) to \( time_1 \).
- \( importance \) indicates an actor’s influence in an EBSN. It is defined by a four-tuple, \( importance = (value, domain, time_0, time_1) \), where
  - \( value \) is a non-negative number which expresses the value of importance.
  - \( domain \) is a string that presents a specified field in which the importance is.
  - \( time_0 \) and \( time_1 \) are time tags which show the importance of an actor-node in the period between \( time_0 \) and \( time_1 \).

Definition 3.3: An event-node \( e_q \) is a node which represents an event in EBSN. It can be defined by a four-tuple, \( e_q = (ID, influence, domain, time) \), where

- \( ID \) is a unique string to identify event-nodes.
- \( influence \) is a non-negative number, which expresses the value of influence;
• domain is a string that presents to which knowledge field the event belongs, and
• time is a time tag indicates when this event has happened.

Event-nodes in EBSN represent interactions among actor − nodes, and usually these interactions could create new relations; and the activeness or/and importance of their participants could always be impacted by events.

Definition 3.4: A lbAE (link between Actor-Node and Event-Node) is a directed arc which can be defined by a two-tuple, lbAE = (ai, ej), where

• ai ∈ A (0 < i) where A is a finite node set which contains all actor-nodes of EBSN, and
• ej ∈ E (0 < j) where E is a finite node set which contains all event-nodes of EBSN.

A lbAE (ai, ej) represents an arc which connects an actor-node ai and an event-node ej in EBSN. It indicates that actor ai participates in event ej. LBAE represents a set of lbAEs which contains all lbAEs of EBSN.

Definition 3.5: A lbEE (link between Event-Node and Event-Node) is another type of directed arc which can be defined by a two-tuple, lbEE = (ei, ej) where

• ei, ej ∈ E; ei.time < ej.time, and 0 < i, 0 < j.

A lbEE connects two event-nodes in an EBSN. It represents that an event ej is affected by another event ei. It directs from ei to ej. LBEE represents a set of lbEEs which is composed of all lbEEs of EBSN.

Definition 3.6: Core is a vector to describe an actor-node which can be defined by a tuple in two dimensions, core = (activeness, importance).

The activeness in this definition represents activeness.value, and importance represents importance.value. The definition of activeness.value and importance.value could be found in Definition 3.2. There are activeness.domain = importance.domain, activeness.time0 = importance.time0 and activeness.time1 = importance.time1. The direction of core can be calculated as θ = arctan(importance/activeness); and the magnitude of core can be calculated as ∥core∥ = √importance² + activeness².

Different actor-nodes may have different values of activeness and importance. Some actor-nodes show more importance compared with their activeness; but some
of them may show more activeness compared with their importance. The core can be used to present the comparison of importance and activeness of an actor-node. Figure 3.1 shows an example of the vector core. The direction of core shows whether the actor-node shows more activeness or importance in EBSN. If the direction of core is closer to activeness then it means that the activeness of this actor – node has more weight, otherwise the importance has more weight or activeness and importance have equal weight in core. Both the magnitude and the angle of a core present the combination of activeness and importance of an actor-node. Comparing two cores which have the same angle (direction), the longer magnitude of core demonstrates a larger value of importance and/or activeness. Comparing two cores which have different angles (directions), the two cores are needed to be normalized first. The details of how to normalize a core are introduced in Section 3.2.1.

3.1.2 Rules to Construct an EBSN

Most social network visualization approaches [78] [43] only treated actors as nodes in social network. Davitz et al. [16] used two types of nodes, e.g. node and supernode, to indicate a network. In EBSN, both actors and events are treated as two types of nodes in the network to make the model carry more information. Figure 3.2 shows a simple EBSN. It can be seen in Figure 3.2 that there are two types of nodes, actor-nodes and event-nodes (refer to Definition 3.2 and Definition 3.3), and two types of arcs, lbAEs and lbEEs (refer to Definition 3.4 and Definition 3.5).

In this subsection, a set of rules have been defined to construct an EBSN.
3.1. Event-Based Social Network Structure

Rule 1: An actor-node could only directly connect to an event-node or event-nodes.

In an EBSN, an actor-node could connect to an event-node or various event-nodes via IbAE. In an event-based social network, people might have a number of types of relations with others, but not all relations among them exist at the beginning. The relations are created by interactions. Rule 1 only allows an actor-node to connect to an event-node or event-nodes, and the relations among people could be traced by analyzing event-nodes. For example, Bob does not know Winly until an event happened, e.g. both of them attended same conference.

Rule 2: The relation between an actor-node and an event-node could only direct from an actor-node to an event-node. In other words, there are only unidirectional relations in an EBSN.

The reason behind the rule is that actor-nodes could choose whether or not to participate in an event-node.

Rule 3: If an event-node $e_m$ affects another event-node $e_n$, then $(e_m, e_n) \in LBEE$, and $e_m.time < e_n.time$.

The reason behind this rule is that if an event is influenced by another event, then this rule can be used to represent the relation between the two event-nodes.

Rule 4: An event-node has to be connected by at least one actor-node.

This can be explained as each event happens because of involvement of at least one actor. In an EBSN, there is no event existing without any actor. For instance, an email must have at least one sender and a receiver(s), an article must have at least one author.
3.2 Formulas

To rank actor-nodes in an EBSN, actor-nodes are needed to be valued based on their activeness and importance. The next subsection introduces formulas to discover the values of the activeness, the importance, and the core of an actor-node.

3.2.1 Knowledge Discovery Based on Actor-Nodes

In this subsection, formulas are proposed to discover activeness, importance and core of an actor-node based on previous definitions.

1. A formula to discover the activeness of an actor-node.

In an EBSN, the value of activeness in domain_0 shows the capable of an actor-node’s activeness in that domain. Intuitively, the value of activeness should be proportionally the number of event-nodes that the actor-node connecting to via lbAE between t_0 and t_1. Equation 3.1 shows the formula to calculate activeness.value of actor-node a\_m under the constraints that a\_m.activeness.domain = domain_0, a\_m.activeness.time_0 = t_0 and a\_m.activeness.time_1 = t_1.

\[a\_m.activeness.value = ||E_n||\] (3.1)

In Equation 3.1, ||E_n|| represents the number of elements in set E_n. E_n is a set of event-nodes, which could be obtained according to Equation 3.2.

\[E_n = \{\forall e_i \in E_n | e_i.time > t_0 \land e_i.time < t_1 \land (a\_m, e_i) \in LBAE \land e_i.domain = domain_0\}\] (3.2)

2. A formula to discover the importance of an actor-node.

The importance of an actor-node indicates the influence of the actor-node in a domain. The importance value relates to the influence of every event-node that the actor-node connects with. Equation 3.3 introduces how to calculate the importance of actor-node a\_m under the constraint that a\_m.importance.domain = domain_0.

\[a\_m.importance.value = \sum_{e_i \in E_m} \alpha_i \cdot e_i.influence\] (3.3)
In Equation 3.3, $\alpha_i$ is the coefficient of $e_i$.influence, $\alpha_i \in [0, 1]$, different events may have different influences to an actor. An event which could have strong impact on the actor has a high coefficient. It is up to the user to determine appropriate values of the coefficients. $E_m$ is a set of events. The importance of an actor-node only depends on the influence of events that the actor participating in. The details of the set $E_m$ are demonstrated in Equation 3.4.

$$E_m = \{\forall e_i \in E_m | (a_m, e_i) \in LBAE \land e_i \text{.domain} = \text{domain}_0\} \quad (3.4)$$

3. **A formula to discover core of an actor-node.**

To discover core of an actor-node, the value of importance and the value of activeness of the actor-node are needed to be discovered firstly. The direction and the magnitude could be calculated using Definition 3.6 in Section 3.1.1.

If actor-nodes would be ranked based on their core, then rules which could normalize cores should be made. Assuming there is a core $aCore = (a, i)$ and a unit vector $\overrightarrow{w}$ whose direction is $\angle \alpha$. The direction of $aCore$ could be calculated as $\angle \theta = \arctan \left(\frac{1}{a}\right)$. Then the angle of normalized $aCore$, $NaCore$, should equal to $\angle \alpha$. Formula 3.5 shows how to calculate the length of $NaCore$, say $||NaCore||$.

$$||NaCore|| = \begin{cases} a \cos \alpha, & \text{when } \alpha < \theta; \\ i \sin \alpha, & \text{otherwise}. \end{cases} \quad (3.5)$$

### 3.2.2 Knowledge Discovery Based on Event-Nodes

As introduced before, an event-node $e_q$ is a node in EBSN which could be represented as $e_q = (ID, influence, domain, time)$. The capability of influence for event-node is represented by influence. Equation 3.6 exhibits how to calculate the value of influence under the constraint that $e_q$.domain = domain$_0$.

$$e_q$.influence = \beta_0 \cdot ||E_m|| + \beta_1 \cdot ||A_n|| + initialValue \quad (3.6)$$

In Equation 3.6, $initialValue$ is a constant, $\beta_0$ is a coefficient of $||E_m||$ and $\beta_1$ is a coefficient of $||A_n||$. The value of $initialValue$ could vary in different domain areas, and
3.3. Case Study

$\beta_0, \beta_1 \in [0, 1]$. The influence of event-node could be affected by participants and other events. The default value of $\beta_0$ and $\beta_1$ are set to 0.5. If events have great influence of the event, then the value of $\beta_0$ is high, and if participants have great influence of the event, then $\beta_1$ is high. The initial value could vary in different circumstances. $E_m$ is a set of event-nodes introduced in Equation 3.7 and $A_n$ is a set of actor-nodes introduced in Equation 3.8, respectively.

$$E_m = \{\forall e_i \in E_m | (e_i, e_q) \in LBEE \land e_i.domain = e_q.domain\}$$  (3.7)

$$A_n = \{\forall a_i \in A_n | (a_i, e_q) \in LBAE\}$$  (3.8)

After event $e_q$ has happened, there may be some other events which occur after $e_q$ and are affected by event $e_q$. In other words, if an event $e_r$ happens after $e_q$ and is affected by $e_q$, then the influence of $e_q$ has increased. The value of $e_q.(influence, domain_0)$ should be updated. The $e_q.(influence', domain_0)$ is used to represent the updated value. Equation 3.9 shows how to calculate the updated value under the constraint that $e_q.domain = e_r.domain$.

$$e_q.influence' = e_q.influence + \beta_3 \cdot e_r.influence$$  (3.9)

In Equation 3.9, $\beta_3 \in [0, 1]$ and $(e_q, e_r) \in LBEE$. If $e_r.influence$ has high impact on $e_q.influence$, the value of $\beta_3$ is set high. In some cases, the value of $\beta_3$ is set low or even 0. For instance, treat publication as events, then for self-referenced publications, the $\beta_3$ could be set to 0.

3.3 Case Study

In this section, the DBLP (Digital Bibliography & Library Project) [41] data set is analysed. The most active actors and most important actors are discovered; and actors in DBLP are ranked.

3.3.1 Conceiving the EBSN

The DBLP data set contains information about authors, their publications (including publication title, author/authors, editor/editors, year, publisher), and citations. Above
information could be used to construct an EBSN. Authors and publications could be represented by nodes in EBSN; and arcs in EBSN could describe relations among author(s) of publications and citations between publications. Let an actor-node represent an author where \( a_m.ID \) = author’s name. A publication could be represented by an event-node where \( e_q.time \) = publish year and \( e_q.ID \) = title of publication. If an author \( a_m \) is the author or editor of the publication \( e_q \) then \( lbAE = (a_m, e_q) \in LBAE \). If a publication \( e_i \) is cited by another publication \( e_j \) then \( lbEE = (e_i, e_j) \in LBEE \). The EBSN could be analyzed after constructing it.

### 3.3.2 Experimental Setup

In this experiment, a DBLP data set which contains 658132 authors and 1087683 publications was downloaded from http://dblp.uni-trier.de/xml/. The data set contains publications and related information from year 1936 to year 2009. Most of the publication topics are related to ‘Database’. In the experiment, domain of all event-nodes and domain of all actor-nodes are set as domain = database. As introduced before, authors are represented by actor-nodes where \( actor - node.ID \) = author’s name, \( actor - node.activeness.value \) equals to the number of the actor-node’s publications. The importance of an actor-node could be calculated using Equation 3.3, where \( \alpha_i = 1 \) which means the influence of events have equal weight.

As explained before, the event-nodes represent publications. The influence of event-node could be calculated using Equation 3.6. In this case, coefficients are set as \( \beta_0 = 1 \), \( \beta_1 = 0 \) and initialvalue = 1. In other words, it is assumed that the influence of publication is impacted by the number of publications which have cited the publication, but not the number of co-authors in this case. In other cases, above values may be adapted to suit each case.

The earliest publication in the DBLP data set was published in 1936 and the latest publication in DBLP data set was published in 2009. In order to simplify the process, the publication time is separated into seven periods. The first period starts from year 1936 and ends at year 1945. After that, every ten years is a separate period until year 2005. Since there are only four years from year 2006 to year 2009 which less than a period, the information from year 2006 to year 2009 is not analysed.
3.3.3 Size of the EBSN

Figure 3.3 shows the numbers of *actor-nodes* in different periods. From this figure it can be seen that the number of *actor-nodes* in the EBSN grows fast, especially from period 1986-1995 to period 1996-2005. The number of *actor-nodes* increases from 130623 to 403571 which means that the number of researchers studying on ‘database’ in the period 1996-2005 increases more than three times than that of in the period 1985-1996.

Figure 3.4 depicts the numbers of *event-nodes* in different periods. From this figure it can be seen that the number of *event-nodes* in the EBSN grows drastically, especially from the period 1986-1995 to the period 1996-2005, just like *actor-nodes*. The number of *event-nodes* rose from 178243 to 570266 which means that the number of publications on topic ‘database’ in the period 1996-2005 grew up more than three times than that of in the period 1985-1996.

3.3.4 Actor-Node Ranking

An *actor-node* in the EBSN represents an author in the DBLP data set. There are two numerical attributes of *actor-node*, *activeness.value* and *importance.value*. Since in period 1996-2005 the EBSN had more actors and events than in other periods, authors in that period are ranked in this subsection.
3.3. Case Study

Figure 3.4: The Numbers of Publications in DBLP in Different Periods

Table 3.1 describes the actor-node ranking based on activeness from year 1996 to year 2005. The activeness is discovered based on Equation 3.1.

Table 3.2 shows the actor-node ranking based on importance from year 1996 to year 2005. The importance is discovered based on Equation 3.3.

Comparing Table 3.1 with Table 3.2, it is found that only in year 2000 the ranking result of top three actor-nodes based on activeness are same as top three actor-nodes based on importance. Actually, in year 2000 the fourth active actor-node is different from the fourth important actor-node. Hence, it can be concluded that in most years the

<table>
<thead>
<tr>
<th>Year</th>
<th>First</th>
<th>Second</th>
<th>Third</th>
</tr>
</thead>
<tbody>
<tr>
<td>1996</td>
<td>Alberto L. Sangiovanni-Vincentelli</td>
<td>Kang G. Shin</td>
<td>Roberto Tamassia</td>
</tr>
<tr>
<td>1997</td>
<td>Kang G. Shin</td>
<td>Sushil Jajodia</td>
<td>Edwin R. Hancock</td>
</tr>
<tr>
<td>1998</td>
<td>Miodrag Potkonjak</td>
<td>Hector Garcia-Molina</td>
<td>Irith Pomeranz</td>
</tr>
<tr>
<td>1999</td>
<td>Edwin R. Hancock</td>
<td>Alok N. Choudhary</td>
<td>Miodrag Potkonjak</td>
</tr>
<tr>
<td>2000</td>
<td>Bill Hancock</td>
<td>Thomas S. Huang</td>
<td>Edwin R. Hancock</td>
</tr>
<tr>
<td>2001</td>
<td>HongJiang Zhang</td>
<td>Thomas S. Huang</td>
<td>Hong Yan</td>
</tr>
<tr>
<td>2002</td>
<td>Mahmut T. Kandemir</td>
<td>Edwin R. Hancock</td>
<td>Sajal K. Das</td>
</tr>
<tr>
<td>2003</td>
<td>HongJiang Zhang</td>
<td>Wei Li</td>
<td>Hans-Peter Seidel</td>
</tr>
<tr>
<td>2004</td>
<td>Wen Gao</td>
<td>HongJiang Zhang</td>
<td>Minglu Li</td>
</tr>
<tr>
<td>2005</td>
<td>Mahmut T. Kandemir</td>
<td>Wen Gao</td>
<td>Licheng Jiao</td>
</tr>
</tbody>
</table>

Table 3.1: Top Three Authors Based on Their Activeness
3.3. Case Study

<table>
<thead>
<tr>
<th>Year</th>
<th>First</th>
<th>Second</th>
<th>Third</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997</td>
<td>Kang G. Shin</td>
<td>Edwin R. Hancock</td>
<td>Elisa Bertino</td>
</tr>
<tr>
<td>1998</td>
<td>Philip S. Yu</td>
<td>Thomas S. Huang</td>
<td>Hector Garcia-Molina</td>
</tr>
<tr>
<td>1999</td>
<td>Edwin R. Hancock</td>
<td>Thomas S. Huang</td>
<td>Elisa Bertino</td>
</tr>
<tr>
<td>2000</td>
<td>Bill Hancock</td>
<td>Thomas S. Huang</td>
<td>Edwin R. Hancock</td>
</tr>
<tr>
<td>2001</td>
<td>Thomas S. Huang</td>
<td>HongJiang Zhang</td>
<td>Edwin R. Hancock</td>
</tr>
<tr>
<td>2002</td>
<td>Mahmut T. Kandemir</td>
<td>Sudhakar M. Reddy</td>
<td>Edwin R. Hancock</td>
</tr>
<tr>
<td>2003</td>
<td>Wei Li</td>
<td>Mahmut T. Kandemir</td>
<td>Hans-Peter Seidel</td>
</tr>
<tr>
<td>2004</td>
<td>Wei Wang</td>
<td>Wen Gao</td>
<td>Chin-Chen Chang</td>
</tr>
<tr>
<td>2005</td>
<td>Chin-Chen Chang</td>
<td>Mahmut T. Kandemir</td>
<td>Philip S. Yu</td>
</tr>
</tbody>
</table>

Table 3.2: Top Three Authors Based on Their Importance

<table>
<thead>
<tr>
<th>Year</th>
<th>First</th>
<th>Second</th>
<th>Third</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997</td>
<td>Kang G. Shin</td>
<td>Edwin R. Hancock</td>
<td>Sushil Jajodia</td>
</tr>
<tr>
<td>1998</td>
<td>Hector Garcia-Molina</td>
<td>Sudhakar M. Reddy</td>
<td>Philip S. Yu</td>
</tr>
<tr>
<td>1999</td>
<td>Edwin R. Hancock</td>
<td>Thomas S. Huang</td>
<td>Elisa Bertino</td>
</tr>
<tr>
<td>2000</td>
<td>Bill Hancock</td>
<td>Thomas S. Huang</td>
<td>Edwin R. Hancock</td>
</tr>
<tr>
<td>2001</td>
<td>Thomas S. Huang</td>
<td>HongJiang Zhang</td>
<td>Edwin R. Hancock</td>
</tr>
<tr>
<td>2002</td>
<td>Mahmut T. Kandemir</td>
<td>Edwin R. Hancock</td>
<td>Sudhakar M. Reddy</td>
</tr>
<tr>
<td>2003</td>
<td>Wei Li</td>
<td>Mahmut T. Kandemir</td>
<td>Hans-Peter Seidel</td>
</tr>
<tr>
<td>2004</td>
<td>Wen Gao</td>
<td>Wei Wang</td>
<td>HongJiang Zhang</td>
</tr>
<tr>
<td>2005</td>
<td>C Mahmut T. Kandemir</td>
<td>Chin-Chen Chang</td>
<td>Wen Gao</td>
</tr>
</tbody>
</table>

Table 3.3: Top Three Authors Based on Their Core

The activeness and importance could catch different attributes of actor-nodes. Also, using EBSN to represent the network could present dynamic changes of the network.

Table 3.3 exhibits the top three authors based on their core in different years. The core-based ranking is based on both the importance and the activeness. It is believed that the core-based rank is an important indicator because it combines both the activeness and importance information in a domain; and the result of core-based ranking could be varied by using different units of core.

Figure 3.5 records the activeness values of top active actor-nodes from year 1996 to year 2005. In the graph, the stacked area indicates the author’s activeness value, and the x-axis represents years. The change of stacked area reveals the author’s research “trend”. If the stacked area increases, then it can be concluded that the author is
in developing period, otherwise it can be concluded that the author is in developed period. There are only eight actor-nodes in Figure 3.5. Refer to Table 3.1, it can be seen two of them, i.e. HongJiang Zhang and Mahmut T. Kandemir, are top active actor-nodes in two years.

Figure 3.6 demonstrates the importance values of the top important actor-nodes from year 1996 to year 2005. In the figure, the stacked area presents author’s importance value, which shows an author’s influence in a specific research area. If the stacked area increases, it can be concluded that the author’s influence is in a developing period. This will provide some hints if researchers are chosen for a project. If more than one researchers have the same importance value, then the one whose influence is in a developing period can be chosen.

3.4 Discussion

Generally, there are three ways to analyse a dynamic social network [64]. The first method is to analyse a snapshot of a time point. This method only analyses very
3.4. Discussion

Figure 3.6: Top Important Authors in Different Years
limited information of the network. It could hardly get high accurate results. The second method is to analyse an aggregated network which aggregates all events into a single network. This method treats the event-based social network as a static network, and achieves a static summary for the network. It could not catch the sequence of the events which are very important for analysing event-based social networks [53]. The third method is to analyse series of aggregated networks. This method is suitable for event-based social networks because it analyses all of the information that the event-based social network contains, and it respects the temporal elements in networks. In the area of social network analysis, most current studies use the first two methods [64] [10]. In this chapter, the third method is used to construct an event-based social network. Actor-nodes are used to represent authors and event-nodes to represent publications. The authors are ranked based on their publications and the citations of their publications. In other words, an actor-node is ranked based on influence of the event-nodes which the actor-node participates in. Furthermore, the network is analyzed as a dynamic network, and authors are ranked in different years based on their activeness value and importance value through the proposed approach. It can be concluded that the same author in different years might have different rank values, and they have different research trends. The core is defined by considering both importance and activeness of an author. Table 3.3 shows the top three authors ranked based on their core. The “developing period” and “developed period” are exploited to describe an author’s research trends. Such information can be used to evaluate core researchers in different fields, and the trends of their impact. This work could also be employed in other event-based social networks. The case study only provides an example of how to use the EBSN.

3.5 Summary

A core-based node ranking approach was introduced in this chapter. This approach could be deployed to rank experts. The formal definitions of an actor-node, an event-node, a lbAE, a lbEE and core of an actor-node were given. The rules to construct an EBSN were introduced. Algorithms were proposed to discover the activeness, importance and core of an actor-node, and the influence of event-node. Algorithms for ranking actor-nodes based on their core were proposed. An EBSN was constructed on the DBLP data. The authors in DBLP were ranked based on their activeness, importance and core in different years, respectively. The “developing period” and the
“developed period” were used to describe an author’s impact trend on the research field. The *EBSN* could also be borrowed to analyse other event-based social networks, such as email networks and disease control fields.
Chapter 4

Discovery of Knowledge Collaborative Communities

A multi-domain problem is a kind of problem that requires knowledge and expertise from multiple domains to find a solution of the problem. How to find a group of suitable experts to solve a multi-domain problem has become an important issue in many fields, such as Engineering, and Social Science. In this chapter, a concept “Knowledge Collaborative Community (KCC)” is introduced. A knowledge collaborative community is a dynamic community which is assembled by experts to solve a specific multi-domain problem and would be disbanded after completing the problem. In this research, knowledge collaboration is defined as an intellectual activity that two or more experts with different domains knowledge work together to achieve their common goal, e.g. solve a multi-domain problem. Research on supporting knowledge collaboration for single-domain problem solving has obtained significant achievements in recent years. A single-domain problem is a problem which only needs expertise from one domain to find a solution of the problem. A Dynamic Community algorithm was proposed by Ye et al. to support knowledge collaboration in software engineering industries [75]. Their model used knowledge relations and social connections to find experts for solving a single-domain problem. A system called D-SNS (Dynamic Social Network System) was proposed by Ohira et al. [50], which used social connections among nodes and “hub” nodes (which are active and linked with many nodes) to build a well connected network to support knowledge collaboration. Poorly connected nodes in their system could be linked via some “hub” nodes. In a scale-free network, their system had bigger probability to discover collaborators than that of DC (Dynamic Community) algorithm. The D-SNS system focused on single-domain problem solving. P@NOPTIC was proposed by Craswell et al. [15]. The goal of P@NOPTIC was to generate an expert profile for knowledge collaboration. All above models and approaches for knowledge collaboration were based on single domain problem solving.
However, in practice, many multi-domain problems require diverse expertise from different domains. Approaches based on a single domain collaboration are difficult to meet requirements for solving most multi-domain problems.

In this chapter, a model for abstracting useful data to build a network is proposed first; then an approach to discover knowledge collaborative communities for solving multi-domain problems in general situations is introduced. There are three main issues that need to be considered to form a knowledge collaborative community for solving a multi-domain problem. The first issue is that all knowledge domains the knowledge collaborative community grasped should cover the knowledge areas which the multi-domain problem solving is required. The second issue is to consider each collaborator’s personal desires, e.g. a collaborator may like working with someone but not every one. The third issue is the performance of knowledge collaboration.

This chapter is arranged as follows. Section 4.1 presents a three-layer model for data abstraction. In Section 4.2, an approach to discover knowledge collaborative communities is introduced in detail based on the three-layer model. In Section 4.3, the attention cost, fatigued candidates, catastrophe phenomenon and knowledge level of a knowledge collaborative community are discussed. Section 4.4 demonstrates the experimental results. The chapter is concluded in Section 4.5.

4.1 The Model Structure

In this section, a three-layer data processing model for building a network is proposed. Before starting to introduce the details of this model, the problem that to be solved is introduced.

**Informal description**: A person wants to solve a multi-domain problem, but he/she does not have sufficient expertise to solve the problem. A group of suitable experts may be discovered to help the person solving the multi-domain problem. It is assumed that the real world information could be collected. The real world information includes events and participants. For instance, a piece of real world information for a simple scenario is that Dr. Ye and Dr. Wan are coauthors of a publication. In this scenario, Dr. Ye and Dr. Wan are two participants and the publication is an event. Another example is that Mr. Zhao and Mr. Sun participate in same project. In this example, the participants are Mr. Zhao and Mr. Sun, and the event is the project that Mr. Zhao and Mr. Sun participate in. An event represents a phenomenon which could affect knowledge domains of people and/or relations of people, and each
4.1. The Model Structure

event has a specific influence to different knowledge domains. Assuming there is a multi-domain problem \( p \) related to knowledge domains \( \{\text{topic}_1, \ldots, \text{topic}_i, \ldots, \text{topic}_n\} = T(p) \) where \( \text{topic}_i \) represents a knowledge domain; and \( T(p) \) is a knowledge domain set which includes all knowledge domains required to solve \( p \). There is a candidate (person) who wants to solve the multi-domain problem \( p \), and the candidate is called as questioner \( q \). Questioner \( q \) will use his/her domain knowledge \( TPC(q) \) to solve multi-domain problem \( p \) where \( TPC(q) \) represents the main knowledge domain of \( q \), and \( q \) represents either a questioner or an expert. Other candidates who master other required domains knowledge could be gathered to solve \( p \). Although a candidate could have knowledge in more than one domains, but he/she has only one main knowledge domain with the strongest knowledge in that domain. It is reasonable to consider the main knowledge domain of a candidate during the selection of suitable members of a knowledge collaborative community.

**Formal Description:** Thus the problem is now transferred to the following description: There is a questioner \( q \) whose main knowledge is \( TPC(q) \) and \( q \) wants to solve a multi-domain problem \( p \). The expertise required in knowledge domains for solving \( p \) could be presented as \( T(p) \). Other candidates whose main knowledge domains are in \( T(p) \) are needed to assemble a \( KCC \) to solve \( p \).

The factors below should be considered when choosing a candidate, say “\( a_x \)”, to join for solving \( p \):

- The candidate \( a_x \)’s main knowledge domain \( TPC(a_x) \) should be a member of \( T(p) \), i.e. \( TPC(a_x) \in T(p) \).

- It is desirable that \( a_x \) has a high knowledge level.

- It is desirable that \( a_x \) has cooperated with questioner \( q \) before.

- It is important that both \( a_x \) and \( q \) are willing to work together.

Figure 4.1 shows an abstract chart of this model. The main idea of this model is to use the real world information to build a network. This model has three layers. The bottom layer is the Basic Layer. This layer abstracts information from the real world and builds an event-based social network, which contains the candidates, events, and their relations. The upper layer is the Actor Layer which contains candidates and their relations, and extracts information from the Basic Layer, as well as the real world. The Actor Layer focuses relations between participants. The top layer is the Knowledge
4.1. The Model Structure

Figure 4.1: The Model Structure
Layer, which contains knowledge extracted from both the basic layer and the actor layer.

Three layers are introduced in detail from Subsection 4.1.1 to 4.1.3, respectively.

### 4.1.1 The Basic Layer

The Basic Layer is in the bottom of the model, and it reflects the information of real world. This layer represents an Event-Based Social Network (EBSN). Events happened in the real world are abstracted by the Basic Layer. Figure 3.2 in Chapter 3 shows the structure of the Basic Layer. It is constructed by actor-nodes (which refer to candidates), event-nodes (which describe affairs related to knowledge domains), the relations among event-nodes and relations among actor-nodes and event-nodes. The formal definitions of above concepts are introduced below.

**Definition 4.1:** A **Basic Layer** (BL) is a bipartite directed graph. It is defined by a two tuple \( BL = (N, C) \), where

- \( N = \{n_1, n_2, ..., n_s\} \) is a finite set which contains all vertices in the graph,
- \( C = \{c_1, c_2, ..., c_r\} \) is a finite set which contains all arcs in the graph, where \( c_i = (n_j, n_k) \) with \( n_j, n_k \in N \).

A Basic Layer is a bipartite graph as there are two types of nodes in the graph, e.g. actor-node and event-node. There are also two kinds of relations, e.g. the relations between actor-nodes and event-nodes, and the relations between event-nodes.

**Definition 4.2:** An **actor-node** \( a_m \) is a node which represents a person in the Basic Layer. It can be defined by a three-tuple, \( a_m = (ID, activeness, importance) \) where

- \( ID \) is a unique string to identify different candidates in Basic Layer.
- \( activeness \) shows how active an actor-node is in different knowledge domains. It is defined by a three-tuple, \( activeness = (domain - vector, time_0, time_1) \) where
  - \( domain-vector \) has \( n \) components, corresponding to the activeness values in \( n \) domain areas.
  - \( time_0 \) and \( time_1 \) are time tags which show the activeness of an actor-node in the period from \( time_0 \) to \( time_1 \).
• \textit{importance} indicates the influence of an actor-node in different knowledge domains. It is defined by a three-tuple, \( \text{importance} = (\text{domain-vector}, \text{time}_0, \text{time}_1) \), where

- \text{domain-vector} has \( n \) components, which corresponds to the importance values in \( n \) domain areas.
- \( \text{time}_0 \) and \( \text{time}_1 \) are time tags which show the importance of an actor-node in the period between \( \text{time}_0 \) and \( \text{time}_1 \).

An \textit{actor-node} in the Basic Layer represents a person in the real world. The \textit{activeness} and \textit{importance} are two attributes to describe a person.

\textbf{Definition 4.3:} An \textit{event-node} \( e_q \) is a node which represents an event in the Basic Layer. It can be defined by a three-tuple, \( e_q = (\text{ID}, \text{influence-vector}, \text{time}) \), where

- \text{ID} is a unique string to identify event-nodes in the Basic Layer.
- \text{influence-vector} has \( n \) components, corresponding to the activeness values in \( n \) domain areas.
- \text{time} is a time tag indicating when this event happens.

An \textit{event-node} in the Basic Layer represents an affair in the real world. The \textit{influence-vector} is used to indicate the influence of this event in different knowledge domains.

\textbf{Definition 4.4:} A \textit{lbAE (link between Actor-Node and Event-Node)} is a directed arc which can be defined by a two-tuple, \( \text{lbAE} = (a_i, e_j) \) where

- \( a_i \in A (0 < i) \) where \( A \) is a finite node set which contains all actor-nodes in the Basic Layer,
- \( e_j \in E (0 < j) \) where \( E \) is a finite node set which contains all event-nodes in the Basic Layer.

A lbAE represents an arc which connects an actor-node and an event-node in the Basic Layer. It indicates that actor-node \( a_i \) participates in event-node \( e_j \). LBAE is used to represent a set of lbAEs which contains all lbAEs of a Basic Layer. If two actor-nodes \( a_i, a_j \) participate in same event \( e_j \), and \( (a_i, e_j), (a_j, e_j) \in \text{LBAE} \), then it could be concluded that the two actor-nodes \( a_i, a_j \) have a relation.
4.1. The Model Structure

Definition 4.5: A \( lbEE \) (link between Event-Node and Event-Node) is another type of directed arcs which can be defined by a two-tuple, \( lbEE = (e_i, e_j) \) where

- \( e_i, e_j \in E; e_i.time < e_j.time \) and \( 0 < i, 0 < j \)

A \( lbEE \) connects two event-nodes in the Basic Layer. It represents that an event \( e_j \) is affected by another event \( e_i \). It directs from \( e_i \) to \( e_j \). For instance, a project \( e_i \) in software industry is used by another project \( e_j \), then \( lbEE = (e_i, e_j) \) represents this relation. The LBEE represents a set of \( lbEE \)s which is composed of all the \( lbEE \)s of a Basic Layer.

The Basic Layer is used to represent useful information from the real world and provide information to the Actor Layer for further knowledge discovery. The actor-node is used to represent a candidate in a knowledge collaborative community and the event-node is used to represent an affair related to knowledge domains. Basically, the Basic Layer is an event-based social network [52]. The event-based social network is a dynamic network, and events in the event-based social network are the power of evolution of this network. An event in a Basic Layer might change the value of activeness and importance of an actor-node, and might change the value of influence of an event-node. The details of how event could impact activeness and importance of actor-node and influence of other events are introduced in Chapter 3. An event could create new \( lbAE \) and \( lbEE \) of the Basic Layer. Since the event-based social network is deployed to update data, the model could capture the latest information of real world. The next subsection introduces the Actor Layer of this model.

4.1.2 The Actor Layer

The actor layer is the second layer of this model. It extracts information from the Basic Layer, as well as the real world. As Figure 4.2 shows, this layer contains AIA's...
4.1. The Model Structure

(Actor In Actor layer) and their connections. Two AIA(s) would have a connection if they participate in same event in the Basic Layer (refers to Definition 4.4 in Subsection 4.1.1) or they have a social relation in the real world, i.e. they might be colleagues, relatives, classmates and so on. Several formal definitions in this layer are introduced below.

**Definition 4.6:** An Actor Layer ($AL$) is a graph. It is defined by a two tuple $AL = (ND, AC)$, where

- $ND = \{nd_1, nd_2, \ldots, nd_s\}$ is a finite set which contains all vertices in the graph,
- $AC = \{ac_1, ac_2, \ldots, ac_r\}$ is a finite set which contains all arcs in the graph, where $ac_i = (nd_j, nd_k)$ with $nd_j, nd_k \in ND$.

The Actor Layer focuses on candidates and their relations.

**Definition 4.7:** An AIA (Actor In Actor layer) is a node which can be defined by a five-tuple, $AIA = (ID, Cul, Pref, Non-Pref, IM)$ where

- $ID$ is a unique string to identify different candidates in Actor Layer.
- $Cul$ is a string that represents the cultural background of this AIA.
- $Pref$ is a set of AIA(s) which includes all AIA(s) this AIA prefers to collaborate with.
- $Non-Pref$ is a set of AIA(s) which includes all AIA(s) that the AIA does not want to collaborate with.
- $IM$ is a vector which has $n$ components corresponding to this AIA’s knowledge level in $n$ knowledge domains.

$$IM = \left[ \frac{noe_{tpc_1}}{noe} \ldots \frac{noe_{tpc_i}}{noe} \ldots \frac{noe_{tpc_n}}{noe} \right] \quad (4.1)$$

Where $ noe_{tpc_i}$ indicates the number of events which includes $tpc_i$ that the AIA participates in; $noe$ is the total number of events the AIA participates in.

The main knowledge domain of an AIA could be derived from its $IM$. If $\frac{noe_{tpc_i}}{noe}$ has the largest value in $IM$, then the main knowledge domain of the AIA is $tpc_i$, which is represented by $TPC(AIA) = tpc_i$. Obviously, the IM could only affected by the event-nodes which AIA participates in.
4.1. The Model Structure

Definition 4.8: A \textit{lbAA (link between Actor-Node and Actor-Node)} is an arc which can be defined as a two-tuple, \(lbAA = (a_i, a_j)\), where

- \(a_i, a_j \in A\); and \(0 < i, 0 < j, i \neq j\)

The lbAA represents that two AIAs, e.g. \(a_i, a_j\), have participated in an event or they have a social relation. In other words, they know each other. LBAA is used to represent all lbAAs in Actor Layer. Generally speaking, if two AIAs have cooperated before or have social connections, they could do a better knowledge collaboration than other pairs who are strangers [50]. The Actor Layer is used to describe candidates and their relations.

4.1.3 The Knowledge Layer

Figure 4.3 shows the Knowledge Layer of the model. The Knowledge Layer is on the top of this model which contains knowledge collaborative communities and AIKs (Actor In Knowledge layer). The knowledge collaborative communities are dynamically formed to solve a multi-domain problem and dissembled when the problem has been solved. In the Knowledge Layer, every AIK has a vector \textit{core} which represents the knowledge level in its main knowledge domain. The definitions of the Knowledge Layer, the knowledge collaborative community and AIK are given below.

Definition 4.9: A \textit{Knowledge Layer(KL)} is a graph. It is defined by a two tuple \(KL = (NK, KC)\), where

- \(NK = \{nk_1, nk_2, ..., nk_s\}\) is a finite set which contains all vertices in the graph,


- \(KC = \{kc_1, kc_2, ..., kc_r\}\) is a finite set which contains all existing subgraphs of \(KL\), where \(kc_i = \{nk_j, ..., nk_k\}\) with \(nk_j, ..., nk_k \in NK\).

**Definition 4.10:** An \(AIK(\text{actor in knowledge layer})\) is a node in knowledge layer which is defined by a three-tuple, \(AIK = (ID, MKnlgD, core)\) where

- \(ID\) is a unique string to identify different candidates in the Knowledge Layer.
- \(MKnlgD\) is the main knowledge domain of \(AIK\).
- \(core\) is a vector of \(AIK\). It represents how active and important of this AIK in its main knowledge domain. The length and angle of \(core\) is depended on activeness and importance of this actor in the Basic Layer. The direction of \(core\) could be calculated as \(\theta = \arctan(\frac{\text{importance}}{\text{activeness}})\); and the magnitude of \(core\) could be calculated as \(\|core\| = \sqrt{\text{importance}^2 + \text{activeness}^2}\).

**Definition 4.11:** A \(KCC(\text{Knowledge Collaborative Community})\) is a subgraph in knowledge layer which is defined by a three-tuple, \(kcc = (p, q, KCM)\) where

- \(p\) represents a multi-domain problem.
- \(q\) is a questioner who needs help from the network to solve \(p\).
- \(KCM\) is a finite set of \(AIKs\) which contains all members of this knowledge collaborative community.

Any AIK could become a questioner if he/she wants to form a KCC to solve a multi-domain problem. The KCC which contains \(q\) is formed to solve the multi-domain problem \(p\), and would disassemble after solving the multi-domain problem \(p\).

Next subsection presents an approach to discover knowledge collaborative communities based on this three-layer model.

### 4.2 Algorithm

In this section, eight factors are introduced which could impact the performance of knowledge collaboration, and an approach is proposed to discover a knowledge collaborative community. It is assumed that there is a multi-domain problem \(p\) and a questioner \(q\). The knowledge domains that are required to solve the \(p\) could be
4.2. Algorithm

represented as $T(p)$. Then a KCC $g_p$ is needed to solve the multi-domain problem $p$. The knowledge domains covered by $g_p$ could be represented as $TPCC(g_p)$ where $TPCC(g_p) = \{TPC(a_1), ..., TPC(q), ..., TPC(a_n)\}$ and $a_1, ..., a_n$ are members of $g_p$.

4.2.1 Evaluation of Knowledge Collaborative Community Based on Eight Factors

There are eight factors to be considered for evaluating the performance of a knowledge collaborative community. They are: knowledge domain coverage, average knowledge level, average connective level[50][75], number of cultural backgrounds[17], number of multi-task candidates, preferred connection, non-preferred connection, and size of the community.

1. Knowledge Domain Coverage: the knowledge domains required by $p$ should be covered by the knowledge collaborative community $g_p$, which could be represented as Function 4.2. By applying Function 4.2, it can be guaranteed that the knowledge collaborative community has enough capacity to solve multi-domain problem $p$.

\[
com(p, g_p) = \begin{cases} 
1; & \text{if } T(p) \subseteq TPCC(g_p) \\
0; & \text{otherwise}
\end{cases} \tag{4.2}
\]

2. Average Knowledge Level: each member in the knowledge collaborative community $g_p$ has a core value in their individual main knowledge domain. The core represents the knowledge level of each candidate. In most cases, a candidate with a higher knowledge level could solve same problems more efficiently than the one with a lower knowledge level. Equation 4.3 represents the average knowledge level of $g_p$.

\[
rank(g_p) = \frac{\alpha_1 \cdot a_1 \cdot \|core'\| + ... + \alpha_i \cdot a_i \cdot \|core'\| + ... + \alpha_n \cdot a_n \cdot \|core'\|}{\|g_p\|} \tag{4.3}
\]

where $\|core'\|$ is the length of normalized core (based on a unit vector $i$ where the default angle of $i$, $\angle i = \frac{\pi}{4}$), $\|g_p\|$ represents the cardinality of $g_p$, $\alpha_i \in [0, 1]$ is the coefficient of $a_i$, $\|core'\|$ and $a_i.MKnlgD \in T(p)$, which could be found in the knowledge layer. The unit core could vary in different application domains, the knowledge level of each expert could be affected by changing the unit core. The unit core is changed to indicate that whether a candidate with higher activeness is preferred or a candidate with higher importance is preferred.
3. **Average Connective Level**: the connections among members of the KCC could improve the efficiency of knowledge collaboration [50] [75]. Equation 4.4 indicates the connective level of a KCC.

\[ s(g_p) = \frac{\|\{(a_i, a_j)|a_i, a_j \in g_p \land (a_i, a_j) \in LBAA\}\|}{EDG} \]  

where \(\|\{(a_i, a_j)\}\|\) represents the cardinality of this set, \(LBAA\) represents all lbAAs in the actor layer, and \(EDG = \frac{\|g_p\|^2 - \|g_p\|}{2}\) is the number of edges to make \(g_p\) as a fully-connected network.

4. **Number of Cultural Backgrounds**: It was pointed out by Diamant et al. [17] that the community members with same cultural background could more efficiently finish their task than the culturally diverse community. Equation 4.5 represents the number of cultural backgrounds in a KCC.

\[ cul(g_p) = \|\{c_1, ..., c_i, ..., c_n\}\| \]  

where \(\{c_1, ..., c_i, ..., c_n\}\) represents a set including all cultural backgrounds of \(g_p\), and \(\|\{c_1, ..., c_i, ..., c_n\}\|\) represents the cardinality of \(\{c_1, ..., c_i, ..., c_n\}\).

5. **Number of Multi-Task Candidates**: a candidate may be capable to finish his/her task in a knowledge collaborative community, but if he/she is involved in many KCCs, then he/she would reduce efficiency of the KCCs due to his/her limited energy capacity. Equation 4.6 presents the number of multi-task candidates in a knowledge collaborative community.

\[ tsk(g_p) = \|\{a_1, ..., a_i, ..., a_n\}\| \]  

where \(\{a_1, ..., a_i, ..., a_n\}\) is a set including all members of \(g_p\) who have multiple tasks, and \(\|\{a_1, ..., a_i, ..., a_n\}\|\) represents the cardinality of \(\{a_1, ..., a_i, ..., a_n\}\). The performance of KCC has an inversed relation with this function, which means if the value of Function 4.6 increases then the performance of a KCC will decrease.

6. **Preferred Connection**: if two candidates like to work together then there is a preferred connection between the two candidates. The two candidates could have an efficient knowledge collaboration. Equation 4.7 represents the average number of preferred connections in a community.

\[ pre(g_p) = \frac{c}{EDG} \]  

(4.7)
where $c = \|\{a_x|a_x \in g_p \land a_x \in a_y.pre \land x \neq y \land a_y \in g_p\}\|$.

7. **Non-Preferred connection**: If two candidates do not like to work together then there is a non-preferred connection between the two candidates. These two candidates would reduce the efficiency of knowledge collaboration of the community if choosing them as members of $g_p$. Equation 4.8 indicates the average number of non-preferred connection in a community.

$$npr(g_p) = \frac{d}{EDG} \quad (4.8)$$

where $d = \|\{a_x|a_x \in g_p \land a_x \in a_y.non - pre \land x \neq y \land a_y \in g_p\}\|$.

8. **Catastrophe Level**: the number of experts in a knowledge collaborative community should not exceed the number of knowledge domains involved by $p$ too much. More experts in a KCC could provide high probability to cause a catastrophe [73]. If there are too many experts in a knowledge collaborative community, the community needs more collaboration paths to support the knowledge collaboration and the cost of communication will also increase. Equation 4.9 shows the catastrophe level of a knowledge collaborative community where $T(p)$ represents the knowledge domains that the solution of multi-domain problem $p$ required.

$$num(p, g_p) = \frac{\|g_p\| - \|T(p)\|}{\|g_p\|} \quad (4.9)$$

After introduced the eight factors which should be concerned when discovering a knowledge collaborative community, the following function is proposed to evaluate the performance of a knowledge collaborative community:

$$pc(g_p) = com(p, g_p) \cdot FT \cdot \frac{1}{num(p, g_p)} \quad (4.10)$$

where

$$FT = \frac{k_1 \cdot rank(g_p) + k_2 \cdot s(g_p) + k_3 \cdot pre(g_p)}{k_4 \cdot cul(g_p) + k_5 \cdot tsk(g_p) + k_6 \cdot npr(g_p)} \quad (4.11)$$

where $k_x$, $x = 1,...,6$, are coefficients of Equations 4.3, 4.4, 4.7, 4.5, 4.6, 4.8 and $k_x \in [0,1]$. All the coefficients and constants could be varied in different application domains. The range of $pc(g_p)$ is from 0 to $+\infty$, $pc(g_p) \in [0, +\infty)$. If the knowledge domains of KCC could not cover the knowledge domains that required for the solution of multi-domain problem, then $pc(g_p) = 0$. If $pc(g_p) = 0$ then it can be concluded that $g_p$ could not solve $p$. The value of $pc(g_p)$ indicates the performance of KCC, and the higher value of $pc(g_p)$, the better. The higher value of $pc(g_p)$ means that the knowledge collaborative community could find the solution of multi-domain problem $p$ more efficiently and more effectively.
4.2.2 An Approach for Discovery of Knowledge Collaborative Communities

The approach to find a knowledge collaborative community based on the three-layer model is shown in Figure 4.4. In Figure 4.4, \( \max(x, A) \) would return \( a_j \) if \( a_j \in A \land a_j.MKnlgD = x \) and \( a_j \) has the highest normalized \( ||core'|| \) value. Line 1 in Figure 4.4 sets questioner \( q \) as the only member of \( g_p \). Line 2 to Line 8 set all candidates with highest knowledge level in required knowledge domains as members of \( g_p \). For each member of \( g_p \) from Line 9 to Line 16 searches the whole network to find whether there is any node which could replace the member and improve performance of \( g_p \). If such a node exists, then replace the member with the node. Line 17 returns the knowledge collaborative community \( g_p \). This approach can be used in many application domains. For instance, it can be used in software development industries and academia to find collaborators.

There is an example to illustrate how the algorithm works. Assume there are five nodes in the network, \( a, b, c, d \) and \( e \). Questioner \( a \) want a collaborator with expertise \( t_p \) for solving a multi-domain problem. Suppose that both \( b \) and \( d \) have expertise \( t_p \), where \( b \) has higher normalized core value than \( d \). After using Line 1 to Line 8 of the algorithm in Figure 4.4, the \( g_p \) contains \( a \) and \( b \). The \( g'_p \) is set to contain \( a \) and \( d \), (refer Lin9 to Line 11). Comparing the \( pc(g_p) \) and \( pc(g'_p) \), if \( pc(g_p) \) is higher than \( pc(g'_p) \), the KCC will contain \( a \) and \( b \), otherwise the KCC will contain \( a \) and \( d \).

4.3 Discussion

In this section, the advantages of the proposed approach are discussed.

4.3.1 Cost of Community Attention

As Ye et al. introduced, the economy of collective attention should be considered in software development industry [73]. In this research, the cost of attention is also concerned in the knowledge collaborative community. Ye et al. proposed an equation to calculate the “Cost of Collective Attention” [73] which was presented as:

\[
C_{oca} = C_{Find} + C_{Ask} + N \cdot C_{Interrupt} + p \cdot N \cdot C_{Skim} + q \cdot N \cdot C_{Answer} + C_{Evaluate}
\]

(4.12)
Algorithm: Discovery of $g_p$

input:
- BL, AL, KL, $q$
- A multi-domain problem $p$

output:
- A Knowledge Collaborative Community $g_p$

begin
1. $g_p = g_p \cup q$
2. for $\forall topic_i : topic_i \in T(p)$
   3. for $\forall a_j \in A$
      4. if $a_j.MKnlgD = topic_i \land a_j.core = Max(topic_i, A)$ then
         5. $g_p \leftarrow g_p \cup \{a_j\}$;
      6. end if
   7. end for
8. end for
9. for $\forall a_i : a_i \in g_p$
10. for $\forall a_j \in A \land a_j.MKnlgD = a_i.MKnlgD$
11.     $g'_p \leftarrow g_p \cup \{a_j\}; g'_p \leftarrow g'_p \setminus \{a_i\}$;
12.     if $pc(g'_p) > pc(g_p)$ then
13.         $g_p \leftarrow g'_p$
14.     end if
15. end for
16. end for
17. return $g_p$
end

Figure 4.4: The Approach Used to Find $g_p$

where $Coca$ was the cost of collective attention, $C_{Find}$ was the attention cost to locate the required expertise, $C_{Ask}$ was the attention cost to formulating and articulating the question, $N$ was the number of potential candidates who could join the community to solve the multi-domain problem, $C_{Interrupt}$ was the attention cost of all potential candidates who were interrupted and distracted by the presented question, the $C_{Skim}$ was the attention consumed by some of the potential candidates who decided to help the questioner to understand the question, $C_{Answer}$ was the attention cost of candidates who thought and composed a response of the question, and $C_{Evaluate}$ was the attention cost of the questioner to evaluate all responses and $p, q \in [0, 1]$.

Equation 4.12 could also be used to evaluate the KCC approach. It is assumed that there are total $N$ potential candidates in the network. $M$ members are chosen to form a knowledge collaborative community from the $N$ potential candidates where $M \leq N$. In the KCC approach $C_{Find}$ could be reduced to 0 because the KCC approach already finds the best candidates for solving the multi-domain problem. $C_{Ask}$ could be reduced
to $\frac{C_{Ask} \cdot M}{N}$ where $\frac{C_{Ask} \cdot M}{N} \leq C_{Ask} \cdot N \cdot C_{Interrupt}$, $p \cdot N \cdot C_{Skim}$ and $q \cdot N \cdot C_{Answer}$ in the KCC approach would be reduced to $M \cdot C_{Interrupt}$, $p \cdot M \cdot C_{Skim}$ and $q \cdot M \cdot C_{Answer}$, respectively. The $C_{Evaluate}$ also has high probability to be reduced, because in the KCC approach only one proposal is evaluated in each knowledge domain in most cases while in other models [75] [51] there would often be more than one proposal to be evaluated. In the KCC approach, the “attention cost” equation would be reduced to

$$C_{Coca} = \frac{C_{Ask} \cdot M}{N} + M \cdot C_{Interrupt} + p \cdot M \cdot C_{Skim} + q \cdot M \cdot C_{Answer} + C_{Evaluate}$$

(4.13)

### 4.3.2 Fatigued Candidates

It was pointed out by Ohira et al. [50] that in software development industry the activeness of network depended on some active developers. If the active developers had too many tasks at the same time, they might leave the community [9]. Candidates in a knowledge collaborative community could also leave the community if they perform too many tasks at the same time. In the KCC approach, Equation 4.6 protects the active candidates in the network. If a candidate is involved in many knowledge collaborative communities, he/she could reduce the value of $pc(g_p)$ in Equation 4.10, then another candidate could replace the busy one.

### 4.3.3 The Probability of Catastrophe Phenomenon

It was pointed out by Yu and Fan [77] that in the process of knowledge collaboration, there might be a catastrophe caused by “excessive knowledge supply”. In the KCC approach, Equation 4.9 and Equation 4.10 could be used to reduce the number of redundant experts, and that could decrease the probability of “excessive knowledge supply”. At the same time, Equation 4.2 could guarantee that the knowledge domains of knowledge collaborative community could cover the knowledge domains required by solving the multi-domain problem.

### 4.3.4 The Knowledge Level of Candidates

The knowledge level is also an important criterion for knowledge collaboration. Candidates with higher knowledge level could solve same problem more efficiently than a candidate with a lower knowledge level. Previous research in this area did not consider
4.4 Experiment

This section uses an experiment to demonstrate how to discover a knowledge collaborative community for solving a specific multi-domain problem. The questioner $q$ and the multi-domain problem $p$ were known. A synthetic data set was used in the experiment. The data set was also used by the dynamic community algorithm of Ye et al. [75], but some additional information was added to simulate the real world information.

### 4.4.1 Data in Experiment

Data of candidates were shown in Table 4.1. In Table 4.1, MKD indicated the Main Knowledge Domain of a candidate, CUL represented the CULTural background of candidate, and TSK displayed the number of current tasks for each candidate. In order to simplify the experiment, the specific values of $Pref$ and $Non−Pre$ of each candidate in the data set did not present. Generally speaking, it is rare that every candidate in a network have the same cultural background. Knowledge globalization makes a knowledge collaborative team as a multi-cultural background team. Then, it was reasonable to set each candidate belonging to a different cultural background.

Table 4.2 showed the data of events. In this table, $Inf−d_1$, $Inf−d_2$, $Inf−d_3$, $Inf−d_4$, $Inf−d_5$, $Inf−d_6$, $Inf−d_7$, $Inf−d_8$, $Inf−d_9$, $Inf−d_{10}$.
4.4. Experiment

<table>
<thead>
<tr>
<th>Event</th>
<th>Inf – d₁</th>
<th>Inf – d₂</th>
<th>Inf – d₃</th>
<th>Inf – d₄</th>
<th>Inf – d₅</th>
<th>Inf – d₆</th>
<th>Inf – d₇</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₁</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>t₁</td>
</tr>
<tr>
<td>e₂</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>t₁</td>
</tr>
<tr>
<td>e₃</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>t₁</td>
</tr>
<tr>
<td>e₄</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>t₁</td>
</tr>
<tr>
<td>e₅</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>t₁</td>
</tr>
<tr>
<td>e₆</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>t₁</td>
</tr>
<tr>
<td>e₇</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>t₁</td>
</tr>
</tbody>
</table>

Table 4.2: Data of Events

The network structure of the Basic Layer is shown in Figure 4.5.

The network structure in Actor Layer is displayed in Figure 4.6. In the figure, connections among AIs presented both social relations and knowledge-related relations.

4.4.2 Experimental Setup

In this experiment, the data introduced in Subsection 4.4.1 was used to examine the KCC approach. The coefficients in Equation 4.11 should be set before the experiment started. To simplify the experiment, it was set that $k_1 = k_2 = k_4 = k_5 = 1$, which meant that the four factors (average knowledge level, number of cultural backgrounds,
number of multi-task candidates, and average connective level) were set in equal weight in the experiment. It was initially set that $k_{3} = k_{6} = 0$ since the data did not contain information about $Pref$ and $Non-Pre$ of a candidate. In Equation 4.3, it was set that $\alpha_{1} = ... = \alpha_{n} = 1$, which meant each knowledge domain in the knowledge collaborative community had the same weight. In Equation 4.6 it was set that $\beta = 1$ which meant that the candidate who had two or more than two tasks at the same time would affect the efficient knowledge collaboration. In different application domains, the coefficients could vary to let different factors have different influence in the knowledge collaborative community.

In the experiment, questioner $q$ and multi-domain problem $p$ were randomly generated ten times. The KCC approach was compared with the Dynamic Community (DC) algorithm, which was proposed by Ye et al. [75]. The experimental results and comparisons of these two algorithms are introduced in next subsection.

### 4.4.3 Experimental Results and Comparison

In the experiment, ten randomly generated multi-domain problems were all successfully solved by the KCC approach. Only 20% of the ten multi-domain problems were solved by the DC algorithm. The above result demonstrates that the KCC approach has a higher probability of solving a multi-domain problem than the DC algorithm in the experimental environment. The performances of Knowledge Collaborative Community (KCC) and Dynamic Community (DC) were compared based on collective attention cost and six factors introduced in the previous subsection. All the comparisons were

![Figure 4.6: The Network Structure in the Actor Layer](image)
1. **Average Size of Communities**: The average number of members in KCC and DC are shown in Table 4.3. From Table 4.3, it can be concluded that the KCC has smaller community size than that of the DC. A smaller community needs less collaboration paths and has less communication cost.

2. **Knowledge Domain Coverage**: In the experiment it could be concluded that the KCC could 100% cover the knowledge domains that the multi-domain problem was required where the DC could only cover 20% knowledge domains that the multi-domain problem was required.

3. **Average Knowledge Level**: The average knowledge level of two communities could be calculated by using Equation 4.3. As introduced before, it was set that $\alpha_1 = \ldots = \alpha_n = 1$, which indicated each knowledge domain in community had the same weight. The average knowledge level of KCC and DC were shown in Table 4.4. From Table 4.4, it could be concluded that the average knowledge level was 1.25 in KCC and 0.86 in DC. The average knowledge level of KCC was higher than that of DC.

4. **Average Connective Level**: The connections among members in a community indicate efficient knowledge collaboration paths. As a common sense, a person collaborating with a familiar person is more efficient than that of with a stranger in same situation. So the connective level is also important for evaluating the efficiency of knowledge collaboration. Equation 4.4 was used to calculate the connective level. The average connective level of each community was shown in Table 4.5. The connective level of KCC was 0.5 and that of DC was 1. The KCC had lower connective level than the DC. The reason for this result was because...
4.4. Experiment

<table>
<thead>
<tr>
<th>Community</th>
<th>Connective Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCC</td>
<td>0.5</td>
</tr>
<tr>
<td>DC</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.5: Average Connective Level

<table>
<thead>
<tr>
<th>Community</th>
<th>Average Number of Cultural Background</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCC</td>
<td>2</td>
</tr>
<tr>
<td>DC</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Table 4.6: Average Number of Cultural Backgrounds

the DC algorithm focuses more on connection, i.e. only two factors, but KCC approach considered eight factors rather than two.

5. **Number of Cultural Backgrounds**: It was pointed out by Diamant et al. [17] that the number of cultural background could affect the collaboration within a community. A person who collaborates with a collaborator in different cultural background would cost more time than collaborating with the same cultural background collaborator.

The larger number of cultural backgrounds within a community means that the knowledge collaboration within the community could cost more resources and might reduce the efficiency of knowledge collaboration [17]. The average number of cultural backgrounds of two communities were shown in Table 4.6. The DC had more cultural backgrounds than the KCC, that meant the KCC could do a more efficient knowledge collaboration than DC in the same situation.

6. **Number of Multi-Task Candidates**: Every candidate has limited energy. To solve a multi-domain problem could cost some energy of a candidate, so a candidate could only participate in a few tasks at a time. If a candidate participates in many tasks, then he/she may leave the community [9]. A candidate is needed to be protected to avoid fatiguing. In other words, a candidate could not participate in many communities at a time. The number of multi-task candidates in a community could be calculated by Equation 4.6. The results of average number of multi-task candidates in each community were shown in Table 4.7. From Table 4.7 it could be concluded that the $KCC$ and the $DC$ had the same result in average number of multi-task candidates.

7. **Average Collective Attention Cost**: Table 4.8 shows the comparison of average attention cost for each community. Equation 4.13 was used to calculate
the value of attention cost. It was set that $p = q = 1$ in Equation 4.13 and $CP = C_{\text{Interrupt}} + C_{\text{Skim}} + C_{\text{Answer}}$. From Table 4.8, it could be concluded that DC had higher average collective attention cost than that of the KCC.

In the experiment, seven comparisons were conducted between KCC and DC. The DC algorithm had better performance only in the average connective level. The DC algorithm and the KCC had the same result on the average number of multi-task candidates. The KCC had better performance than that of the DC algorithm in the rest of comparisons. Please note that in order to make fair comparisons, in this experiment the experimental data did not update. Otherwise, the KCC approach could get a better result. In the KCC approach networks were treated as dynamic networks and each event could affect the network.

### 4.5 Summary

In this chapter, firstly, a three-layer model was introduced to abstract data, discover expertise of experts and to support for discovering knowledge collaborative communities. The real world information was abstracted by the model and transferred into useful data. Secondly, an approach named Knowledge Collaborative Community (KCC) was proposed to discover knowledge collaborative communities for multi-domain problem solving. Thirdly, KCC was compared with a common used method, DC, in terms of knowledge domain coverage, average knowledge level, average connective level, the number of cultural backgrounds, multi-task candidates as well as the collective attention cost. The experimental results demonstrated that the KCC approach had better performance than that of DC method.
Chapter 5

Multi-Domain Problem Solving in Large-Scale Networks

The real world could be modeled as a large-scale network in terms of multi-domain problem solving in many domains, which contains uncountable nodes representing potential problem solvers and relations representing links of problem solvers. Most multi-domain problems happen in such large-scale networks. A large-scale network is a big size interaction graph that two nodes in terms of candidates or experts could get a connection between them if they have participated in same affair or have a collaboration before, such as co-authors of a paper, teammates of a software project, etc. [29]. How to find a group of suitable candidates to solve a multi-domain problem in a large-scale network is an important issue in the area of collaborative community discovery. In Chapter 4, a three-layer model for abstracting information and an approach for discovering a Knowledge Collaborative Community (KCC) were proposed. The approach introduced in Chapter 4 focused on general considerations from small-size to medium-size of networks and was not suitable to apply to large-scale networks, due to the huge computational costs. One simplest way to reduce the computation cost is browsing part of the network instead of browsing the whole network.

This chapter focuses on knowledge collaboration among experts for multi-domain problem solving in large-scale networks. The discussion of how to discover expertise of an expert is not introduced in this chapter. It is assumed that expertise of each expert is known. Each expert is assigned to one knowledge domain only, with the assumption that most experts usually have substantial knowledge in only one domain, while, in other domains, these experts’ knowledge might not be sufficient. A two-step approach to find experts with required expertise in a large-scale network is proposed in this chapter. The proposed approach does not explore the whole network for finding experts. Instead, only a subgraph of the large-scale network is analysed. The subgraph contains potential candidates who could help questioner q to solve a multi-domain problem. After the subgraph is discovered, suitable experts are chosen to form a KCC.
The performance of knowledge collaboration has been considered when experts are chosen for a KCC.

This chapter is arranged as follows. Section 5.1 is the problem description. Section 5.2 introduces a two-step approach to find a Knowledge Collaborative Community in a large-scale network. Experiments which demonstrate the good performance of the approach is presented in Section 5.3. Related work and comparison are elaborated in Section 5.4. The chapter is summarized in Section 5.5.

## 5.1 Problem Description

Some definitions need to be defined before the problem description is given.

**Definition 5.1:** A Large-Scale Network ($LSN$) is an undirected graph. It is defined by a two-tuple, $LSN = (A, E)$, where

- $A = \{a_1, ..., a_n\}$ is a finite set which contains all nodes in the graph.
- $E = \{e_1, ..., e_r\}$ is a finite set which contains all arcs in the graph, where $e_i = (a_j, a_k)$ with $a_j, a_k \in A$.

The $LSN$ contains all experts and their relations. All nodes in $LSN$ are potential candidates to form a Knowledge Collaborative Community.

**Definition 5.2:** A node $a$ representing an expert in $LSN$ is defined by a seven-tuple, $a = (ID, domain, rank, pref, non-pref, cul, kc)$ where

- $ID$ is a unique string to identify different nodes.
- $domain$ is a string which indicates the knowledge domain of node $a$.
- $rank$ is a number which represents the knowledge level of node $a$. A higher value of rank means a higher knowledge level of the node (expert).
- $pref$ is a set of nodes that node $a$ likes to cooperate with.
- $non-pref$ is a set of nodes that node $a$ does not like to work with.
- $cul$ is a string which indicates the cultural background of node $a$.
- $kc$ is a set of Knowledge Collaborative Communities which node $a$ currently participates in.
A node in a LSN represents an expert. Each expert has a unique ID. The domain indicates the expertise field of this expert. Knowledge domain of an expert is an important criterion for choosing KCC members, since the knowledge domain of an expert is a criterion for judging whether this expert is required for the KCC. In Definition 5.2, the rank indicates the knowledge level of an expert. Personal wills of an expert are expressed by \( \text{pref} \) and \( \text{non} - \text{pref} \).

**Definition 5.3:** An arc \( e \) is an undirected edge connecting two nodes, and is defined by a two-tuple \( e = (a_i, a_j) \) where

- \( a_i, a_j \in A \).

The \( a_i, a_j \in A \) indicates that the two node \( a_i, a_j \) are in the same LSN. An arc in a LSN means that the two nodes has a connection between them. In other words, there is an efficient knowledge collaboration path between these two nodes.

**Definition 5.4:** A multi-domain problem \( p \) is a problem which needs knowledge from more than one domains to find a solution. \( p \) can be defined by a two-tuple, \( p = (ID, TPC) \), where

- \( ID \) is a unique string to identify different multi-domain problems.
- \( TPC = \{ tpc_1, \ldots, tpc_i, \ldots, tpc_n \} \) is a finite set of knowledge domains which contains all knowledge domains required to find a solution of the \( p \), where \( tpc_i \) represents a knowledge domain.

**Definition 5.5:** A questioner \( q \) is a special node in a LSN, which has a multi-domain problem \( p \) to be solved. \( q \) is defined by a two-tuple, \( q = (a, p) \) where

- \( a \) is the node, which has a multi-domain problem.
- \( p \) is a multi-domain problem.

A questioner \( q \) is a special node in LSN with a multi-domain problem \( p \); and the questioner \( q \) has no sufficient knowledge and expertise requested by solving \( p \). Any node \( a_i \) could become a questioner if it has a multi-domain problem to solve, and could assemble a KCC (Knowledge Collaborative Community) to solve the multi-domain problem. After solving the multi-domain problem or failing to assemble a KCC, (which means that the questioner fails to solve the multi-domain problem), a questioner could turn back to a normal node.

**Definition 5.6:** A Potential Community (PC) is a subgraph of LSN. It is defined by a six-tuple \( PC = (PCA, q, PCE, PCD, PCN, PCNE) \), where
5.1. Problem Description

- \( PCA \) is a set of nodes indicating all members of the \( PC \).
- \( q \) is a questioner. The \( PC \) is formed to help \( q \) to find a Knowledge Collaborative Community.
- \( PCE \) is a set of arcs which contains all arc members of the \( PC \).
- \( PCD \) is a set of domains. Each member of \( PC \) could find his/her knowledge domain in \( PCD \), and each knowledge domain in \( PCD \) is obtained by at least one \( PC \) member.
- \( PCN \) is a set of nodes which contains all neighbors of this \( PC \).
- \( PCNE \) is a set of arcs which contains all arcs that connect the \( PC \) with its neighbors.

Since a \( LSN \) contains uncountable nodes and arcs, it is impossible to browse all nodes to discover a Knowledge Collaborative Community. A small subgraph is needed in which there are limited nodes and every nodes could be explored. The \( PC \) is the small subgraph in \( LSN \) which is assembled for solving a multi-domain problem \( p \). A \( PC \) is assembled when a KCC is going to be discovered, and it would be dispersed after a KCC is discovered.

**Definition 5.7**: A Knowledge Collaborative Community is defined by a three-tuple \( KCC = (KCCA, KCCD, q) \) where

- \( KCCA \) is a set of nodes which contains all members of the Knowledge Collaborative Community.
- \( KCCD \) is a set of domains. Each member of \( KCC \) could find his/her knowledge domain in \( KCCD \), and each knowledge domain in \( KCCD \) has at least one \( KCC \) member who is an expert in the knowledge domain.
- \( q \) is a questioner. The KCC is assembled for \( q \).

A KCC is formed for questioner \( q \) to efficiently solve a multi-domain problem \( p \).

After introducing the formal definitions, the basic procedure of discovery of a KCC could be described as follows. Suppose that there is a questioner \( q \) with a multi-domain problem \( p \) in a \( LSN \) (Large-Scale Network); and \( q \) could not solve the multi-domain problem since it has no sufficient knowledge and expertise requested by \( p \). The questioner \( q \) needs to find appropriate experts in the \( LSN \) to form a Knowledge
5.2 A Two-step Approach

In this section, a two-step approach for discovering a KCC for multi-domain problem solving is introduced based on the definitions in Section 5.1.

5.2.1 Principle

Figure 5.1 briefly shows the basic principle of the two-step approach. The top block in Figure 5.1 presents the problem to be solved by this approach. In a Large-Scale Network (LSN) a questioner $q$ wants to assemble a KCC to solve a multi-domain problem $p$. The middle block in Figure 5.1 describes the Step 1 as that a $PC$ would be discovered for $q$. The bottom block in Figure 5.1 represents Step 2 as that a KCC would be discovered for $q$ by exploring all members of the PC, and the task is finished.
5.2.2 Step 1: Discovery of Potential Community in LSN

The main purpose of Step 1 is to discover a PC for questioner q to solve multi-domain problem p. A KCC could be discovered in the PC instead of the LSN, which could efficiently reduce the computation cost. The PC is discovered by using the following criteria:

1. The PC is a small-size subgraph of LSN.
2. PC members should have expertise in p.TPC, except q.
3. To complete PC, the PC.PCD should cover p.TPC.

The first criterion is grounded in transferring the problem from a large-scale network to a small-size network. The reason behind this criterion is that it is not realistic to explore all nodes in a large-scale network, concerning the computation cost. However it is possible to explore all members in a small-size network. The second criterion is grounded in that every member except q should have expertise for solving p, in which case all nodes are potential members of KCC. Questioner q needs to stay in PC since the PC is assembled for q. The third criterion is grounded in that all knowledge domains of requested by p are important for finding a KCC. A KCC could solve a multi-domain problem only if the KCC could cover all knowledge domains that p required.

In Step 1, at first, there are only a questioner q and a multi-domain problem p. The goal of Step 1 is to find a PC for questioner q. The detail of this step is shown as:

1. At the beginning, an incompleted PC is set to only contain q, which could be written as PC.PCA = q. Figure 5.2 shows notations of PC, q and q’s neighbors, where a neighbor of q is a node which has an arc connected to q.
2. All neighbors of q are joined into PC. If the PC could cover all knowledge domains of multi-domain problem p, then the PC is completely formed.
3. If the current PC could not cover all knowledge domains of multi-domain problem p required, all neighbors of PC are added into PC, where a neighbor of PC is a node which has an arc connected to any member of PC. It is shown in Figure 5.3.
4. Step 3 is repeated until a PC which could cover all knowledge domains of p is found, or no neighbor of PC exists. If no neighbors of PC can be found and PC.PCD ∩ p.TPC ≠ p.TPC, then the task fails.
5.2. A Two-step Approach

5.2.3 Step 2: Discovery of KCC in PC

To discover a KCC, the efficiency and performance of knowledge collaboration should be considered. There are some factors which could affect the performance and/or the efficiency of KCC. These factors were introduced in Chapter 4.2.1, which were knowledge domain coverage, average knowledge levels of candidates, average connective levels, number of cultural backgrounds, number of multi-task candidates, preferred
connections and non-preferred connections. These factors need to be used in Step 2 of the approach to evaluate the performance of a KCC.

These factors could be classified as two groups.

1. The first group contains positive factors. Positive factors could improve the performance of knowledge collaboration. In a simple word, the higher value of the factors could lead to a better performance of knowledge collaboration. There are three factors in the first group, which are $\text{rank}(g_p)$, referring to average knowledge level, $s(g_p)$, referring to average connective level and $\text{pre}(g_p)$, referring to preferred connections.

2. The second group contains negative factors. Negative factors have negative affections to knowledge collaboration. There are also three factors in the second group, which are $\text{cul}(g_p)$, referring to number of cultural backgrounds, $\text{tsk}(g_p)$, referring to number of multi-task candidates and $\text{npr}(g_p)$, referring to non-preferred connections.

Equation 5.1 represents the affections of above two group factors. Positive factors in group one are summarized as a numerator of a fraction. Negative factors in group two are summarized as a denominator of a fraction. Equation 5.1 is a simple expression for the affections of the factors. Any expression could replace Equation 5.1 if the expression could correctly present the affections of the factors.

$$pe(g_p) = \frac{k_1 \cdot \text{rank}(g_p) + k_2 \cdot s(g_p) + k_3 \cdot \text{pre}(g_p)}{k_4 \cdot \text{cul}(g_p) + k_5 \cdot \text{tsk}(g_p) + k_6 \cdot \text{npr}(g_p)} \quad (5.1)$$

where $k_1, ..., k_6$ are coefficients. The value of $k_1, ..., k_6$ are in $[0, 1]$, and can vary in different application domains. Different ratios of the coefficients indicate different weights of affection of each factor. For instance, $k_1 = ... = k_6 = 0.5$ presents all the factors have the same weight of affection to a KCC, which means all these six factors have equal importance in the KCC. If one factor is more important than other factors, the value of the coefficient of this factor will be increased. Otherwise, the value of the coefficient of the factor is decreased.

A Potential Community ($PC$) has been discovered in Step 1. The goal of Step 2 is to discover a $KCC$ for questioner $q$ in $PC$. Since the $PC$ is a small-size subgraph, then the computation cost of exploring all nodes of $PC$ is affordable. In Step 2, at the beginning the KCC is set only containing $q$. Then, each member of $PC$ temporarily joins into $KCC$, if the performance of $KCC$ is improved, then let the member join $KCC$, otherwise, move to the next member. If the size of $KCC$ is too big, then the
5.2. A Two-step Approach

The performance of KCC would go down. So it is necessary to control the size of KCC. In Step 2, the size of KCC is maintained as small as possible, only candidates who could improve the performance of KCC (if the node joins KCC could increase the value of Equation 5.1) could join into KCC. If a member of KCC could low the performance of KCC down, then the candidate should leave the KCC. Whenever a new candidate joined the KCC, each member of KCC would evaluate whether remove the member out of KCC or not. If removing the member could improve the performance of KCC, the member will be move out of KCC. The structure of the whole approach is introduced in the next subsection.

5.2.4 The Structure of the Approach

The structure of the approach to discover a KCC from a LSN is shown in Figure 5.4. In Figure 5.4, Line 1 to Line 14 discover a PC from LSN for solving a multi-domain problem. Line 15 to Line 33 find a KCC from the PC which was discovered by Line 1 to Line 14. The more detail explanation of the approach gives below.

Line 1 to Line 2 initialize the approach. The questioner q is the only member of PC (Potential Community) and also the only member of KCC initially. A temporal community tempKCC is defined in Line 3 for representing a modified KCC. Line 4 to Line 6 keep joining neighbors of the PC into the PC until members in the PC could solve multi-problem p or no neighbor of the PC exists. During the process, the size of the PC is increased to find required expertise for solving multi-domain problem. Line 7 to Line 9 decide whether members in the PC could solve p. If they could not solve the problem, a failure message will be returned and the approach is ended. Line 10 to Line 14 narrow the size of the PC. All members with required expertise could remain in the PC, and others are moved out from the PC.

Line 15 to Line 33 discover an optimal KCC in the PC. Line 15 to Line 18 evaluate each node in the PC. If adding a node into KCC could increase the number of Knowledge domains of the KCC, the node would be added in the KCC. Otherwise, Line 19 to Line 21 evaluate whether adding the node into KCC could improve the performance of KCC. If joining the node could improve performance of the KCC, then the node is added into KCC. Line 22 to Line 32 control the size of KCC. Line 22 judges whether a node has been added into the KCC. If a node has been added into the KCC (i.e. the size of KCC has changed), then Line 23 to Line 32 continue to check whether other nodes in the KCC except questioner q could be removed out of KCC. If removing a node out of KCC could improve the performance of KCC, then the node will be removed.
5.3. Experiment

In this section, an experiment is presented to evaluate the two-step KCC approach. The performance is a percentage number used to evaluate the two-step KCC approach and can be obtained by the following formula:

\[
\text{performance} = \frac{\|smp\|}{\|tmp\|} \times 100\%
\]  

(5.2)

Where \(\|smp\|\) represents the number of successfully solved multi-domain problems, and \(\|tmp\|\) expresses the number of total multi-domain problems. The range of performance...
5.3. Experiment

is from 0 to 100%. The high value of performance means that the approach has high probability to solve a multi-domain problem.

Some comparisons between the two-step KCC approach and another algorithm named Dynamic Community are also given. The experiment is designed to evaluate the performance of the two-step KCC approach in discovering a KCC in a large-scale network. The Dynamic Community algorithm is also used in the same large-scale network to discover a dynamic community to help a questioner solving a collaborative problem. In the experiment, firstly, a large-scale network is built. Secondly, a multi-domain problem $p$ is randomly generated and assign to a questioner $q$ to form a KCC/DC for solving $p$.

5.3.1 Experimental Setup

The programming language for this experiment is c++. In this experiment, a random network [56] is generated, which contains $N$ nodes. In the experiment the $N$ is varied to represent different size of networks. The probability for connection of each pair in the random network is set to $(2/N)\%$. The probability is varied to indicate different connective networks from poorly connected networks to well connected networks. The coefficients in Equation 5.1 are set as $k_1 = k_2 = k_3 = k_4 = k_5 = k_6 = 0.5$ to indicate that all six factors have an equal weight. A random knowledge domain is assigned to each node. The knowledge level of each node is also assigned randomly. Then a multi-domain problem $p$ is created and assigned to a random node $q$. The node $q$ becomes a questioner. The questioner in this random network may or may not have the expertise required for solving $p$. Since $q$ is the person who need collaborators to help him/her solve a problem, it is reasonable if he/she does not have required knowledge. Questioner $q$ needs to stay in the KCC even he/she does not have required knowledge since the KCC is formed for helping the questioner $q$.

5.3.2 Experimental Process and Results

The main process of the experiment is:

1. A random network is generated, and multi-domain problem $p$ and questioner $q$ are created.

2. Using the two-step KCC approach, which is introduced in Figure 5.4, tries to find a $KCC$ in the generated random network.
3. A Dynamic Community, which is introduced by Ye et al. [75] [76], is discovered in the same random network using the same multi-domain problem $p$ and the same questioner $q$.

4. The performance of this two methods are compared.

**Experiment 1**: Firstly the number of nodes $N$ in a network was set to 20, 50, 100, 200, 300, 500 and 1000, respectively with the probability $(2/N)\%$ for the connection of each pair in the network. Then $p$ and $q$ were generated ten times in each size of network. At each time for the same $p$ and $q$, the two-step KCC approach and Dynamic Community ($DC$) approach were separately used to discover the community which could help $q$ to solve the multi-domain problem $p$. The goal of this experiment was to compare the performances of these two approaches in different size of networks. The result is shown in Figure 5.5.

![Figure 5.5: The Performance in Different Size Networks](image)

In Figure 5.5, the “KCC” represents “the two-step Knowledge Collaborative Community approach” and the “DC” represents Dynamic Community algorithm. The X axis in Figure 5.5 represents the number of nodes in the network, and the Y axis represents the percentage of successful rate to discover a KCC or a DC. From Figure 5.5 it could be concluded that in all cases, the KCC had better performance than that of the DC. The best result for DC was 30%, appears on 20 nodes in the network. After the size of network was larger than 200 nodes, the best result for DC reduced to
10%. The experimental result obviously shows that the DC has better performance in small-size networks than its performance in big sized networks. In more than 50% cases, the KCC had the best performance, i.e. 90%. The worst performance of KCC appeared on 200 nodes in the network. The best performance for KCC happened when the network contained 50, 100, 500, 1000 nodes. The experimental result indicates that the performance of KCC is fractionally impacted by the size of network, and the two-step approach has good performance in both small networks or large-scale networks. Compared with the DC algorithm, the two-step KCC approach had better performance.

Experiment 2: The following experiment was used to evaluate the performance of the two-step KCC approach from poorly connected networks to well connected networks. In this experiment, the number of nodes in the network was set to 200, and the probability of the link between a pair-node was set in different values, i.e. 0.01, 0.02, ..., 0.2. Different probability of links between a pair-node indicated different connective networks. The higher value of probability meant a denser network. For each probability, $p$ and $q$ were randomly created ten times. At each time, the KCC approach and the DC algorithm were used to discover a community for solving the multi-domain problem $p$, respectively. The results are shown in Figure 5.6. The X axis in Figure 5.6 represents the probability of connection between a pair-node in a network. The Y axis indicates the percentage of successful rate to discover the right community for solving.
5.4 Related Work and Comparison

The Dynamic Community (DC) was introduced by Ye et. al [75][76]. The goal of a DC algorithm was to find experts for knowledge collaboration. It was supposed that there was a candidate who had a problem to be solved. Then the candidate used the DC algorithm to discover experts who might help him/her. The candidate posted the problem to potential experts and waited for solutions. All chosen experts and the candidate together were called as a Dynamic Community [76]. There were two criteria for finding suitable experts to assemble a DC. The first criterion was that an expert must have expertise for a specific problem. The second criterion was that the expert should have social contacts with the candidate. The DC had good performance in a well connected network or when the candidate was well connected, but it had poor performance (i.e. it can not find suitable experts) in a poorly connected network or when the candidate was not well connected. The two major differences between DC and the two-step KCC approach, are that (1) the method assembled a DC includes all experts who have related expertise and have connections with the candidate. This research assembled a KCC includes only a few experts who have required expertise, and (2) their method has high probability that could not find an expert with required expertise if the candidate is not well connected, and the two-step KCC approach has higher probability that could assemble a KCC to find a solution of multi-domain problem in a poorly connected network. The experimental results have demonstrated this fact.

Wikipedia [44] is an internet-based, user contributed knowledge collaboration platform and is a very popular tool to create or change contents (articles). The goal of Wikipedia was to maintain a neutral point of view [44]. The mechanism of Wikipedia
supported many to many communication and editing. The major differences between the Wikipedia and the two-step KCC approach are that (1) their approach has many participants and the KCC only contains necessary experts, and (2) their approach allows all users, no matter they are experts or not, to edit contents in Wikipedia and the two-step KCC approach ranks experts and chooses experts who have high knowledge levels to join a KCC.

A system was introduced by Ohira et al. [50], called Dynamic Social Networking System (D-SNS). The D-SNS can be used to support knowledge collaboration in large-scale networks. The D-SNS built a well connected network via “hub” [67] candidates. In their system, if a questioner wanted to solve a problem, all candidates who had related expertise and had connections with the questioner, would receive the problem description. Then some of chosen candidates replied the questioner, and the questioner could judge whether the multi-domain problem can be solved by these candidates. The D-SNS can be used to develop a well connected network. However, the Collective Attention Cost [73] in D-SNS was high since all candidates connected to the questioner with required expertise would spend their attention on understanding the multi-domain problem, and some of them needed to spend more time on finding a solution of the multi-domain problem. In fact only a few experts in the same field were needed to contribute on the solution. The differences between D-SNS and KCC are that: (1) D-SNS uses “hub” candidates to build a well connected network to overcome the poor performance in a poorly connected network, while KCC has high probability to discover a KCC in any network without the need to spend resources to rebuild a network, (2) D-SNS does not control the number of experts when building a network, while KCC assembles a KCC with only necessary experts, and (3) D-SNS assumes that every expert has same knowledge level, and the two-step KCC approach chooses experts with higher knowledge level.

5.5 Summary

Multi-domain problem solving in a large-scale network is an important issue. In this chapter, a two-step Knowledge Collaborative Community (KCC) approach which could group experts to solve multi-domain problems in a large-scale network is proposed. A KCC is assembled when a questioner wants to solve a particular multi-domain problem. The KCC has all required expertise for solving the particular multi-domain problem and guarantees a solution of the multi-domain problem. The two-step KCC approach
works in two steps. The first step is to find a Potential Community (PC) for KCC
discovery. The second step is to select members of KCC from the PC. The experimental
result indicates that the two-step KCC approach has good performance in a large-scale
network. It also has an outstanding performance in a poorly connected large-scale
network.
Chapter 6

Conclusion

The rapid development of the modern world brings huge amount of data with multi-domain problems. This thesis aimed to discover knowledge collaborative communities for multi-domain problem solving, and proposed the following three approaches.

1. A ranking approach to rank expert for knowledge collaboration;

2. An approach to discover a knowledge collaborative community in small-size or medium-size networks to solve a multi-domain problem;

3. An approach to discover a knowledge collaborative community in a large-scale network to solve a multi-domain problem.

This chapter highlights the major contribution of this thesis through the summary of each proposed approaches, and outlines the direction of future work.

6.1 Major Contribution

A new concept, named core and a core-based node ranking approach were introduced in Chapter 3. Then an approach to discover knowledge collaborative communities in a small-size or medium-size network was proposed in Chapter 4, and knowledge level of experts was also introduced in this chapter. Thereafter, an approach used in a large-scale network to discover a knowledge collaborative community for multi-domain problem solving was presented in Chapter 5. The contributions of each proposed approaches are outlined in the following subsections, respectively.

6.1.1 A Core-Based Node Ranking Approach

A core-based node ranking approach is introduced in Chapter 3. This approach is used in event-based social networks, and it is suitable for expert finding tasks. Rules to
6.1. Major Contribution

build an event-based social network which could be used to analyze actors, events and the relationships among them are also presented in that chapter. The advantages of this approach are as follows.

1. The core-based node ranking approach is query relevant. Compared with the PageRank algorithm [13], which is not query relevant, this approach could let a user have more influence to the ranking result by providing different unit cores.

2. Different from the HITS algorithm [37], the ranking result of the approach is relied on two attributes of node, which are importance and activeness, and it could vary in different application domains (by given different unit core).

3. Compared with MITRE’s ExpertFinder System [46], the ranking result could be easily impacted by enquirers but hardly affected by candidates.

4. Compared with the Expertise Propagation algorithm [26], both the links and influences of events could affect the ranking result.

The core-based node ranking approach is suitable to be used in discovery of knowledge collaborative communities for choosing experts.

6.1.2 An Approach to Discover A KCC

In Chapter 4, a three-layer model to support for discovering KCCs was introduced; and an approach to deploy the three-layer model to discover a KCC for multi-domain problem solving, say KCC approach, was proposed in that chapter. The advantages of KCC are shown as follows.

1. Compared with the DC algorithm [75], the KCC approach could be used in a network to solve not only a multi-domain problem, but also a single-domain problem.

2. Different from most previous research, e.g. the DC algorithm [75] and D-SNS system [50], the KCC approach chooses only necessary experts to assemble a KCC, and this mechanism could increase the probability to avoid a catastrophe phenomenon.

3. Against most previous research, the KCC introduces a new concept, say a knowledge level, to describe experts. Compared with an expert with lower knowledge
level, the expert with higher knowledge level may solve same problem more efficiently. The DC algorithm [75] and the D-SNS system [50] assumed that every expert had same knowledge level, but the KCC approach chooses experts with higher knowledge level.

This approach is suitable to be used in a small-size or medium-size network.

6.1.3 An Approach to Discover A KCC in Large-scale Networks

An approach used in large-scale networks to discover a KCC for multi-domain problem solving was presented in Chapter 5. Compared with the KCC approach proposed in Chapter 4, the two-step KCC approach focuses on discovering a KCC in larger-scale networks, while the KCC approach focuses on small-size or medium-size networks. The advantages of this two-step KCC approach are shown as follows.

1. The two-step KCC approach could discover a KCC for a multi-domain problem without exploring all nodes in the large-scale network.

2. Compared with D-SNS [50] and DC [75], the approach has better performance in a poorly connected network.

3. This approach could be applied in a decentralized network.

4. All factors which may affect the success or efficiency of knowledge collaboration, e.g. knowledge coverage, personal desire, knowledge level, size of community and number of cultural background, are all considered in this approach based on our best knowledge.

6.2 Limitations and Future Work

Although this thesis provided solutions to discover KCCs for solving multi-domain problems in different sizes of social networks, some issues and limitations still remain to be solved. These issues and limitations are considered to be in the future work.

- The core-based node ranking approach proposed in Chapter 3 could work only when data are stored in a database. If data are stored in databases on different formats, the approach would not work. This limitation could be solved by deploying ontology techniques. By doing this, the information about events and
nodes could be searched and obtained automatically. The individuals and events in different databases then could be ranked.

• The KCC approach introduced in Chapter 4 could discover expertise of experts, knowledge level of experts and a KCC for a multi-domain problem. In the process of discovering expertise of experts, each event is only assigned to one knowledge domain. However, in the real world, some events may involve more than one knowledge domain. This limitation will be considered by the future work.

• The two-step KCC approach introduced in Chapter 5 may fail to discover a KCC due to lack of one or more expertise. In the future work, an expert will be assigned to more than one knowledge domains to improve the performance of this approach.


[79] Jing Zhang, Jie Tang, Liu Liu, and Juanzi Li. A mixture model for expert finding. In Advances in Knowledge Discovery and Data Mining, volume 5012 LNCS of
