Journal of Information Processing Vol.22 No.2 243–252 (Apr. 2014)

[DOI: 10.2197/ipsjjip.22.243]

Regular Paper

Community Discovery for Knowledge Collaborations in Collective intelligence Systems

Jing Jiang[1,a)] Quan Bai[1,b)] Minjie Zhang[2,c)] Shaojie Yuan[2]

Received: July 7, 2013, Accepted: January 8, 2014

Abstract: Knowledge collaborative communities play an important role in collective intelligence systems. To discover a knowledge collaborative community, we need to consider not only the structure of a network but also the performance of knowledge collaboration among members within the community. Traditional community discovery approaches are not suitable to discover knowledge collaborative communities since most of them focus too much on the network topologies, and ignore some other important factors. In this paper, we propose two community discovery approaches, which can be used in different sizes of networks, and take more knowledge collaboration factors into account. Compared with some other existing approaches, the proposed approach can perform better in forming knowledge collaborative communities for multi-domain problem solving.

Keywords: community discovery, knowledge collaborative community, multi-domain problem solving, collective intelligence

1. Introduction

Collective intelligence is aggregate intelligence that emerges from the collaboration and competition of many individuals [16]. For decades, knowledge collaboration and collective intelligence have been considered as effective ways for enhancing problem solving and decision making capabilities of human and computer systems [5], [8], [12]. Especially in nowadays, with the rapid evolution of computer and Internet technologies, people more rely on collaborations to solve complex problems. Meanwhile, more and more information and knowledge are generated by groups of experts rather than individuals [1], [18]. Hence, how to facilitate the generation of collective intelligence has become an important research question.

In a collective intelligence system, a Knowledge Collaborative Community (KCC) can be considered as a group of experts combining and sharing their knowledge to solve a problem. Such hidden knowledge is required for many recommendation systems for complex problem solving or cross-domain problem solving. However, in large scale and fast changing collective intelligence systems, how to select suitable experts from candidates with overlapping or even identical knowledge and skills, and ensure successful collaboration among them is still a challenging problem.

In this paper, we introduce a community discovery approach for recommending suitable expert groups for multi-domain problems. The experts may be from different domains, and connected together through some interactions. The proposed approach can analyse and estimate the collaboration quality of expert groups (i.e., how well they can work together), and recommend suitable groups for a cross-domain complex problem.

The rest of this paper is arranged as follow. In Section 2, some related work has been reviewed. The problem description and related definitions are proposed in Section 3. Factors which may affect the knowledge collaboration are introduced in Section 4. The Knowledge Collaborative Community Discovery (KCCD) approach for centralized networks and the 2-Step approach for distributed networks are introduced in Section 5. The experimental results and analysis are given in Section 6. Finally, the paper is concluded in Section 7.

2. Related Work

In recent years, community discovery in networks has attracted the attention of researchers from different areas including collective intelligence, social network analysis, knowledge engineering, etc [2], [7], [11]. In general, community can be considered as a kind of hidden knowledge in network or social transactional data [9]. A community can be discovered by using data-driven approaches or domain expertise. Community detection focuses on understanding and supporting knowledge transferring and collaboration. Several popular community detection model will be introduced in this section.

The Dynamic Community (DC) was proposed by Ye et. al [18], [19]. The goal of a DC approach 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 approach to discover experts who might help the candidate. 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 [19]. The DC method includes all experts who have related expertise and have connections with

[1] Auckland University of Technology, Auckland, New Zealand
[2] Wollongong University, Wollongong, Australia
[a)] jing.jiang@aut.ac.nz
[b)] quan.bai@aut.ac.nz
[c)] minjie@uow.edu.au

© 2014 Information Processing Society of Japan

the candidate, while the proposed approach in this paper includes only the experts with required expertise. Wikipedia [6] 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 [6]. The mechanism of Wikipedia could support many-to-many communication and editing. Our approach is different from Wikipedia in two aspects. Firstly, Wikipedia has many participants, but the KCCD/2-Step approach proposed in this paper only contains necessary experts. Secondly, Wikipedia allows all users, no matter what their specialised areas are, to contribute to collaborated knowledge. However, in order to guarantee the quality of KCC for mutli-domain problems, our community discovery approaches only include experts who have high knowledge levels in the required domains to a KCC.

A Dynamic Social Networking System (D-SNS) was introduced by Ohira et al. [10]. The D-SNS can be used to support knowledge collaboration in distributed networks. The D-SNS built a well connected network via "hub" [15] candidates. In their system, if a questioner (requester) 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 could be solved by these candidates. The D-SNS can be used to develop a well connected network. However, the Collective Attention Cost [17] 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 much 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 of a requested problem. The differences between D-SNS and our community discovery approaches 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 our approaches have high probability to discover a KCC in any network without spending resources to rebuild a network; (2) D-SNS does not control the number of experts when building a network, while our approaches assemble a KCC with only necessary experts; and (3) D-SNS assumes that every expert has a same knowledge level, while our approaches consider different knowledge levels of different experts, and choose experts with higher knowledge levels.

3. Problem Description and Definitions

A collective intelligence system can be modeled as a network containing a number of interactive experts (nodes). The network can also be represented as a graph, where the nodes in the graph stand for the experts in the network, and the graph edges denote the interaction relationships among experts [14]. There can be different kinds of interaction relationships among experts, e.g., co-authorship, membership of the same department, participant of a conference, or citation. In this paper, we focus on the analysis of co-authorship as the data can be easily obtained.

This research is based on the following assumptions. Suppose there is an expert node q in the network who wants to solve a multi-domain problem p. As p requires multi-domain expertise, which q does not have, we need to find a knowledge collaborative community for solving the p. In addition, we also assume that each expert can only have at most one specialised domain. Namely, most experts usually have substantial knowledge in their own specialised domains, while in other domains, their knowledge might not be sufficient. The networks could be classified into two types based on their topologies, i.e., centralized networks and distributed networks. In centralized networks, we suppose there is a central control system and all nodes of network could be explored. When a multi-domain request is proposed, the central control system can go through all available nodes in the system and compose a community for the request. However, for distributed networks, there is no global view for all nodes in a network, and the node information is stored in distributed repositories without central control system. Obviously, approaches for discovering KCCs in the two types of networks should be different. We also assume that the factors which may impact the efficiency of knowledge collaboration for centralized networks and distributed networks are same.

In this paper, two approaches are proposed for KCC discovery, i.e., the KCCD approach and the 2-Step approach. The KCCD is developed for community discovery for centralized networks, and the 2-Step approach is designed for distributed networks.

Definition 1: A network N is an undirected graph, which is defined as a two-tuple, i.e., $N = (A, E)$. $A = \{a_1, \ldots, a_n\}$ is a finite node set which contains all nodes in N. $E = \{e_1, \ldots, e_r\}$ is a finite arc set which contains all arcs in N.

Definition 2: A node (a_i) in N represents an expert in the system. It is defined by a five-tuple, i.e., $a_i = (nID, domain, rank, coAuthor, kcc)$. nID is the unique id of a_i. $domain$ is the tag which indicates the specialised domain of a_i. $rank$ is the number which indicates the knowledge level of a_i (higher $rank$ value means the expert has higher knowledge level in $domain$). kcc is a set of Knowledge Collaborative Communities that a_i belongs to. $coAuthor$ is a matrix which represents the co-authorship relationships of a_i and other expert nodes in the system (see Eq. (1)). a_k in $coAuthor$ is an expert node with co-publications with a_i. n_{ik} in $coAuthor$ is the numbers of co-publications between a_i and a_k.

$$coAuthor = \begin{pmatrix} \ldots, & a_k, & \ldots \\ \ldots, & n_{ik}, & \ldots \end{pmatrix} \quad (1)$$

Definition 3: An arc e_{ij} is an undirected weighted edge in N. It defined by a three-tuple $e_{ij} = (a_i, a_j, w_{ij})$ where a_i and a_j are the connected expert nodes of e_{ij}, and w_{ij} is the weight of e_{ij}.

e_{ij} is established after a_i and a_j have some interactions. Meanwhile, the establishment of e_{ij} means that there is an effective knowledge collaboration path between a_i and a_j. In addition, w_{ij} (the weight of e_{ij}) denotes the knowledge collaboration degree between the two expert nodes, namely, how effective collaborations a_i and a_j can have. As we mainly focus on the co-authorship relationships here, an interaction means that the two expert nodes have a co-publication.

Definition 4: A multi-domain problem p_i is a problem, which needs to be solved by using knowledge from more than one do-

© 2014 Information Processing Society of Japan

mains. p_i can be defined by a two-tuple, $p_i = (pID, TPC)$, where pID is a unique id of the problem, and TPC is a matrix denotes required knowledge for solving p_i (see Eq. (2)). In Eq. (2), tpc_j is a required knowledge domain for solving p_i, and α_i means how important the knowledge in tpc_i in solving p_i and the sum of values of α_i is 1. (Here, we assume knowledge from different knowledge domains has different contributions for solving p_i).

$$TPC = \begin{pmatrix} \cdots, & tpc_j, & \cdots \\ \cdots, & \alpha_j, & \cdots \end{pmatrix} \tag{2}$$

Definition 5: A requester q is a node in N with a multi-domain problem to be solved.

Any node in N can become a requester if it has a multi-domain problem, and then, will call for a KCC to solve the problem. After solving the problem (or failing to solve the problem), a requester will turn back to a normal node.

Definition 6: A Knowledge Collaborative Community kcc_i is a subgraph in N, which is defined by a six-tuple, $kcc_i = (p_i, q, KCM, KCE, KD)$, where p_i represents the multi-domain problem that kcc_i corresponding to, q_i is the requester who proposed p_i, KCM is a finite set of nodes which contains all expert members in kcc_i, KCE is a set of arcs which contains all arcs in the kcc_i, and KD is a set of knowledge domains that all expert nodes in kcc_i are specilised in. For each knowledge domain in KD, there exists at least one expert node $a_k \in kcc_i$ who is specialised on it.

kcc_i is formed for solving p_i, and will be disassembled after p_i is solved.

Definition 7: A Potential Community (PC) is a subgraph of N. It is defined by a six-tuple $PC = (PCA, q, PCE, PCD, PCN, PCNE)$, where PCA is a node set containing all nodes in PC, q is a requester with a multi-domain problem to be solved, PCE is an arc set containing all arcs of the PC, PCD is a set of knowledge domains containing all knowledge domains member of PC specialised on, PCN is a node set which contains all neighbour nodes of PC and $PCNE$ is an arc set which contains all arcs connecting PC with its neighbour nodes.

A Potential Community is formed only when we want to discover KCCs in a distributed network. Namely, when a network is too large to be explored, a Potential Community (PC) is identified for community discovery. A Potential Community is dispersed after a KCC is discovered.

4. Evaluation of Knowledge Collaborative Community Quality

The success of solving a multi-domain problem relies on both the expertise of participated members and the collaboration quality among expert members. In this section, we will introduce the factors of effective team configuration, which can yield the best trade-off among knowledge coverage for multi-domain problem, and the rank function based on members' competence for the required knowledge and cohesion of community [4]. Furthermore, we will introduce how to use the factors to evaluate the quality of a KCC.

In this research, we consider five major factors, which may impact on the efficiency of knowledge collaboration. The five factors are (1) the knowledge domain coverage, (2) the average knowledge level, (3) the average connective level, (4) the community connectivity, and (5) the catastrophe level. In our approach, these five factors are used to evaluate the quality of KCCs.

(1) **Knowledge Domain Coverage**

Suppose there is a multi-domain problem p_i. The knowledge domains that are required to solve p_i could be represented as $T(p_i)$. kcc_i is a potential community for solving p_i. The knowledge domains covered by kcc_i could be represented as $TPCC(kcc_i) = \{a_i.domain | a_i \in kcc_i\}$. Ideally, the knowledge domains required by p_i should be covered by kcc_i. Eq. (3) is the function for checking whether kcc_i has sufficient domains of p_i.

$$com(p_i, kcc_i) = \begin{cases} 1; \text{ if } p_i.TPC \subseteq TPCC(kcc_i) \\ 0; \text{ otherwise} \end{cases} \tag{3}$$

(2) **Average Knowledge Level**

Each member in kcc_i has a *rank* value in their individual knowledge domain. The *rank* 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. The average knowledge level of kcc_i represents the overall knowledge level of the whole community and can be calculated by using Eq. (4).

$$rank(kcc_i) = \frac{\alpha_l \cdot a_i.rank + \alpha_m \cdot a_j.rank + \alpha_n \cdot a_k.rank}{\|kcc_i.KCM\|}, \tag{4}$$

where $\|kcc_i.KCM\|$ represent the cardinality of $kcc_i.KCM$, and $kcc_i.KCM$ represents all members of kcc_i.

(3) **Average Connective Level**

The connections among members of a KCC also have great impact on the quality of knowledge collaboration [10], [18]. The average connective level of a KCC is a factor for evaluating how well a KCC (kcc_i) is connected. It can be calculated by using Eq. (5).

$$s(kcc_i) = \frac{\left\| \left\{ (a_i, a_j) | a_i, a_j \in kcc_i.KCE \wedge (a_i, a_j) \in E \right\} \right\|}{EDG}, \tag{5}$$

where $\left\| \left\{ (a_i, a_j) \right\} \right\|$ represents the total number of edges in kcc_i, E represents all edges in the network, and EDG is the number of edges to make kcc_i as a fully-connected network. EDG can be calculated by using Eq. (6). If $\|kcc_i.KCM\| = 1$, we define EDG as 1.

$$EDG = \frac{\|kcc_i.KCM\|^2 - \|kcc_i.KCM\|}{2} \tag{6}$$

(4) **Community Connectivity**

The frequency of previous interactions is an important indicator of successful community collaboration. Co-Author of two researchers implies that the function of expert a_i and the function of expert a_j interact with each other [13]. We calculate the experts connection degree ECD based on publicly available publication records, given by:

© 2014 Information Processing Society of Japan

$$ECD(a_i, a_j) = 1/2 \times \left[\frac{f(a_i, a_j)}{(a_i)} + \frac{f(a_i, a_j)}{(a_j)} \right] \quad (7)$$

where, $f(a_i, a_j)$ is the total number of papers which experts a_i and a_j co-authored; and $f(a_i)$ denotes the total number of papers which experts a_i publish; $f(a_j)$ denotes the total number of papers which experts a_j publish.

In order to measure how well an individual expert belongs to the rest of the community, we also define a local clustering coefficient for a node in the community ($a_j \in kcc_i$) as follows [3]:

$$C_{a_j} = \frac{2 \times \sum_{a_k, k \neq j} ECD(a_j, a_k)}{||kcc_i.KCM|| \times (||kcc_i.KCM|| - 1)} \quad (8)$$

The global clustering coefficient for the community is then the average of local coefficients of all nodes, and the community connectivity $Con(kdd_i)$ can be calculated by using Eq. (9).

$$Con(kcc_i) = \frac{\sum_{a_i \in kcc_i.KCM} C_{a_i}}{||kcc_i.KCM||} \quad (9)$$

(5) **Catastrophe Level**

The number of experts in a knowledge collaborative community should not exceed the number of knowledge domains required by a problem p_j. More experts in a KCC could provide high probability to cause a catastrophe [20]. 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 be also increased. Equation (10) shows the catastrophe level of a knowledge collaborative community kcc_i where $p_j.TPC$ represents the knowledge domains that p_j requires.

$$num\left(p_j, kcc_i\right) = \frac{||kcc_i.KCM|| - \left||p_j.TPC\right||}{||kcc_i.KCM||} \quad (10)$$

As introduced in previous paragraphs, the above five factors will impact on the quality of knowledge collaboration, and should be taken into account in the evaluation of KCC performance. Hence we propose the following equation which can give an overall evaluation about the performance of kcc_i.

$$\begin{aligned} pc(kcc_i) =& com(p_i, kcc_i) \cdot [k_1 \cdot rank(kcc_i) + k_2 \cdot s(kcc_i) \\ &+ k_3 \cdot Con(kcc_i) - num(p_i, kcc_i)] \end{aligned} \quad (11)$$

In Eq. (11), k_1, k_2 and k_3 ($k_1, k_2, k_3 \in [0, 1]$), are coefficients of the average knowledge level, average connective level and community connectivity, respectively. The values of k_1, k_2 and k_3 can be varied in different application domains. The value of $pc(kcc_i)$ indicates the performance of kcc_i. The higher value $pc(kcc_i)$ is, the better performance kcc_i can achieve in solving p_i. If $pc(kcc_i) = 0$, it can be concluded that kcc_i could not solve p_i. For example, if the knowledge domains of kcc_i cannot cover the required knowledge domains of p_i, the value of $com(p_i, kcc_i)$ is 0, and the value of $pc(kcc_i)$ is also 0. In such a situation, p_i cannot be solved by kcc_i. 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 very necessary.

5. The Knowledge Collaborative Community Discovery Approach

Two approaches are introduced in this paper. The first approach is the KCCD approach, it is utilized to discover a KCC in a centralized network, where a global view of all expert nodes of a network is available. However, such a condition cannot be satisfied in many large scale networks, so the second approach is the 2-Step approach for discovering KCCs in distributed networks.

5.1 The KCCD Approach

In the KCCD approach, all nodes of a network can be explored, and potential expert nodes for a problem can be gathered to form a KCC. The details of the KCC approach is presented by Algorithm 1 in **Fig. 1**. Line 1 in Algorithm 1 sets requester q as the only member of kcc_i. Line 2 to Line 6 browse all nodes in N, and select candidates with required expertise in each knowledge domain of p as a member of kcc_i. Line 7 to Line 9 check whether the N contains a KCC of p or not. For each member in particular domain $domain_k$ of kcc_i, Line 10 to Line 19 search the whole network to find whether there is any node can replace the member and improve the performance of kcc_i. If such a node exists, then the corresponding member will be replaced. Line 20 returns the final knowledge collaborative community kcc_i.

5.2 The 2-Step Approach

Figure 2 shows the major procedures in the 2-Step approach. When a requester q proposes a problem p_i to be solved in N, the 2-Step approach will firstly generate a potential community PC from q (i.e., Step 1). Then, a KCC will be discovered from all members of the PC (i.e., Step 2).

A PC is discovered by using the following criteria:

(1) The PC is a small-size subgraph of N.

(2) PC members should have expertise in $q.p_i.TPC$.

(3) To complete PC, the $PC.PCD$ should cover $p_i.TPC$.

The first criterion is grounded in transferring the problem from a distributed network to a centralized network. The reason behind this criterion is that it is not realistic to explore all nodes in a distributed network, concerning the cost of time and computation. The second criterion is grounded in that every member in a PC should contribute some expertise in solving p_i, in which case all nodes are potential members of a KCC. The third criterion is grounded in that all required knowledge domains of p_i are important for finding a KCC. A KCC could solve a multi-domain problem only if the KCC can cover all required knowledge domains of p_i.

In Step 1 of the 2-Step approach (see Fig. 2), initially, only q is included as the only member in PC. Then, the PC will be reorganised by following the the steps below.

(1) At the beginning, an uncompleted PC has only one member q, which could be presented as $PC.PCA = q$ (see **Fig. 3**).

© 2014 Information Processing Society of Japan

Algorithm 1: The KCCD Approach

input:
N, q
output:
A Knowledge Collaborative Community kcc_i
begin
1 Set $kcc_i.KCM = kcc_i.KCM \cup q.a$
2 **for** $\forall a_j \in A$
3 **if** $a_j.domain \in q.p_i.TPC \wedge \|kcc_i.KD\| < \left\|\left(kcc_i.KD \bigcup a_j.domain\right)\right\|$ **then**
4 $kcc_i.KCM \leftarrow kcc_i.KCM \cup \{a_j\}$;
5 **end if**
6 **end for**
7 **if** $pc(kcc_i) = 0$ **then**
8 *EXIT* (Form KCC Fail.)
9 **end if**
10 **for** $\forall domain_k : domain_k \in q.p_i.TPC$
11 **for** $\forall a_i : a_i \in kcc_i \wedge a_i \in domain_k$
12 **for** $\forall a_j \in A \wedge a_j \in domain_k$
13 $kcc_i' \leftarrow kcc_i \cup \{a_j\}$; $kcc_i' \leftarrow kcc_i' \setminus \{a_i\}$;
14 **if** $pc(kcc_i') > pc(kcc_i)$**then**
15 $kcc_i \leftarrow kcc_i'$
16 **end if**
17 **end for**
18 **end for**
19 **end for**
20 **return** kcc_i
end

Fig. 1 The Community discovery algorithm of the KCCD approach.

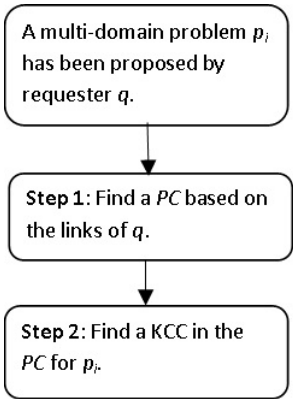


Fig. 2 Major procedures in the 2-Step approach.

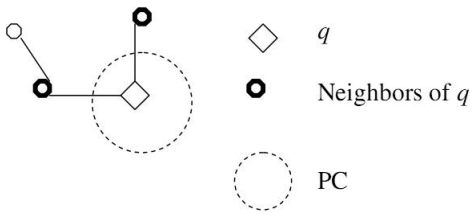


Fig. 3 Initial PC.

(2) All neighbours of q are joined into PC. If the PC could cover all knowledge domains of p_i, then the PC is completely formed.

(3) If the current PC could not cover all required knowledge domains of p, neighbours of all members in PC are added into PC (see **Fig. 4**).

(4) Step 3 is repeated until the PC can cover all knowledge domains of p_i, or there is no more neighbours can be included. If there is no more neighbours to add, and $PC.PCD \cap p.TPC \neq p.TPC$, the task fails.

(5) After a PC is found, then the size of PC is needed to be narrowed. Those nodes whose knowledge domains are not in $p.TPC$ would be removed from PC (except q).

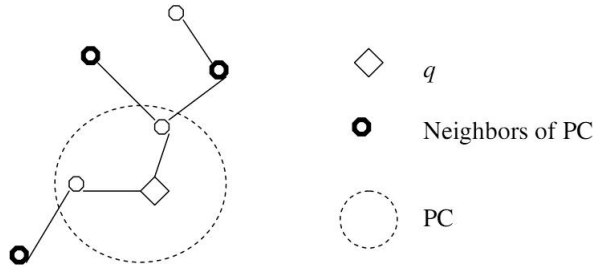


Fig. 4 neighbours of PC.

After the execution of Step 1, if the approach fails to find enough candidates to solve p_i, the whole approach is terminated (and the task is failed). If a PC with sufficient candidates for solving p_i can be discovered, the approach process to Step 2. Similar to the KCCD approach, the forming of the KCC in the 2-Step approach is also based on the evaluation of the performance of KCC. At the beginning of Step 2, only q is included in the KCC. Then, each member of PC temporarily joins into KCC, and the performance of the temporal KCC will be evaluated by using Eq. (11).

The community discovery algorithm of the 2-Step approach is shown in **Fig. 5**. In the algorithm, Line 1 to Line 14 are used to find a PC from N for solving a multi-domain problem p_i (i.e., Step 1). Line 15 to Line 31 are to find a KCC (kcc_i) from the PC, which is discovered in Step 1 (i.e., Step 2). More detailed explanation of the algorithm is given below.

- Line 1–Line 2: The approach is initialised. The requester q is the only member in PC;
- Line 3: A temporal community $tempKCC$ is generated;
- Line 4–Line 6: PC is expanded by including members' neighbours into PC until existing members in PC can solve p_i or no neighbours can be included;

© 2014 Information Processing Society of Japan

Algorithm 2: The 2-Step Approach

input:
N, q
output:
A Knowledge Collaborative Community kcc_i
begin
Approach Step 1
Discovery of PC
1 Set $PC.PCA = \{q.a\}$
2 Set $kcc_i.KCM = \{q.a\}$
3 Set $tempKCC = kcc_i$
4 **while** $(q.p_i.TPC \nsubseteq PC.PCD)$ **or** $(\|PC.PCN\| \neq 0)$ **do**
Set PC include its neighbours
5 $PC.PCA = PC.PCN$
6 **end while**
7 **if** $(q.p_i.TPC \nsubseteq PC.PCD)$
8 **EXIT** (Failed to find PC).
9 **end if**
10 **for each** a_i **in** $PC.PCA$
11 **if** $a_i.domain \notin q.p_i.TPC$ **then**
12 $PC.PCA = PC.PCA \setminus \{a_i\}$
13 **end if**
14 **end for**
Approach Step 2
15 **for** $\forall a_j \in PC.PCA$
16 $tempKCC.KCM = tempKCC.KCM \cup \{a_j\}$
17 **if** $\|tempKCC.KD\| > \|kcc_i.KD\|$ **then**
18 $kcc_i = tempKCC$
19 **end if**
20 **end for**
21 **for** $\forall domain_k : domain_k \in q.p_i.TPC$
22 **for** $\forall a_i : a_i \in kcc_i \wedge a_i \in domain_k$
23 **for** $\forall a_j \in PC.PCA \wedge a_j \in domain_k$
24 $kcc'_i \leftarrow kcc_i \cup \{a_j\}; kcc'_i \leftarrow kcc'_i \setminus \{a_i\};$
25 **if** $pc(kcc'_i) > pc(kcc_i)$**then**
26 $kcc_i \leftarrow kcc'_i$
27 **end if**
28 **end for**
29 **end for**
30 **end for**
31 **return** kcc_i
end

Fig. 5 The Community discovery algorithm of the 2-Step approach.

- Line 7–Line 14: Checking whether members in PC can solve p_i, and whether existing members have required knowledge. Members with required expertise could remain in PC, and others are removed;
- Line 15–Line 20: Evaluating whether adding a node into KCC can increase knowledge domains of the KCC, and improve the performance of KCC;
- Line 21–Line 31: Checking whether existing nodes in the KCC can be removed. If removing a node does not decline the performance of the KCC, the node will be removed;

6. Experiment

Three experiments are conducted for demonstrating the performance of the KCCD approach and the 2-Step approach in KCC discovery. In the experiments, the Dynamic Community (DC) approach is selected as the benchmark in the experiments. In Experiment 1, both KCCD approach and DC approach are conducted in a centralized network and compared by four factors mentioned in Section 4, i.e., average catastrophe level, knowledge domain coverage, average knowledge level and average connective level. Experiments 2 and 3 evaluate the performance of the 2-Step approach and DC approach in a distributed network by changing the networks size and connectivities.

6.1 The DC Approach

In the DC approach, when there is a candidate with a multi-domain problem to be solved, the DC approach discovers experts who might help him/her. There were two criteria to find 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. Therefore, the *DC* approach has good performance in a well connected network or when the candidate was well connected, but it might have poor performance (i.e., it cannot find suitable experts) in a poorly connected network or when the candidate was not well connected with other nodes in the network.

A dynamic community (DC) is dynamically formed in a knowledge work space which consists of a group knowledge workers and the knowledge that the workers hold. There are three kinds of relations in such a knowledge work space: the relation between workers; the relation between worker and knowledge; and the relation between knowledge. Suppose a requestor q needs a set of required knowledge domain $p_i.TPC$ for solving multi-domain problem p_i. The DC will include all related workers who satisfy all three kinds of relations. Within the same dynamic community DC for specific multi-domain problem p_i, not only all workers are used to have interaction with the requestor q, but also they hold knowledge in specific domain d_k which belongs to the set of required knowledge domain $p_i.TPC$ or is related to the domains in the $p_i.TPC$.

6.2 Dataset

According to the KDDC approach and the 2-Step approach, the dataset for experiments requires following essential attributes:

- Knowledge Domain: According to TPC, each multi-domain problem requires more than one knowledge domain and each required domain with a particular important degree.
- Expert: A set of experts belong to a particular knowledge domain. Certain experts can have multi-domain knowledge, but they will have different IDs in different domains. Such multi-domain experts mainly contribute to decrease catastrophe level.
- Knowledge Level: In terms of a particular domain, each expert has a specific knowledge level.
- Collaboration History: For each particular expert a_i, there is a matrix which represents the collaboration records of a_i and other experts in the systems.

It is hard to find a realistic dataset which can include all previous mentioned attributes. For example, the most popular dataset for community discovery research is DBLP dataset. However, it is hard to get the knowledge domain and knowledge level for a particular author. Therefore, we generates datasets including the co-authorship collaborations and knowledge domain relationships to imitate the co-authorship networks. We define a knowledge domain set consisting of five knowledge domains. A multi-domain problem p_i is proposed by a particular expert node q, and p_i randomly includes not less than two and no more than five different knowledge domains from predefined knowledge domain set. We suppose that an expert node q wants to publish a new paper p_i, which needs several different domain knowledge and each

© 2014 Information Processing Society of Japan

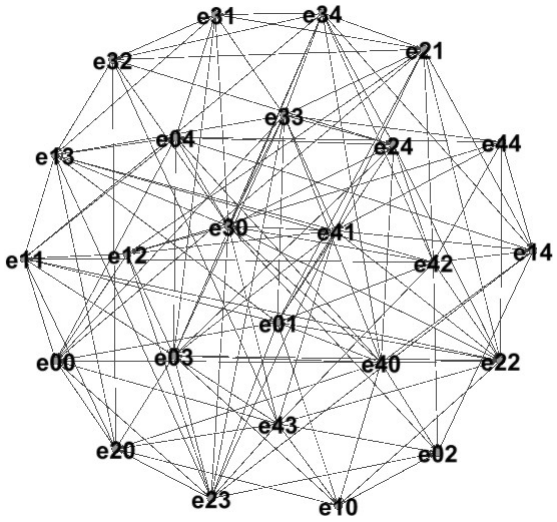


Fig. 6 The structure of the expert network.

domain occupies different important degree in p_i, so the system will compose a knowledge collaborative community including a group of experts, which not only satisfise all knowledge domain required by the expert node q, but also has better performance than the other candidate groups. Detailed datasets generation will be described in each experiments.

6.3 Experiment 1

Experiment 1 is conducted to demonstrate how to discover a knowledge collaborative community by using the KCCD approach in the centralized network with global information.

6.3.1 Experimental Setup

In the experiment, the synthetic dataset that contains a network with five knowledge domains and each domain consists of five expert nodes with specific knowledge level from 1 (lowest) to 5 (highest). The network structure of the dataset is shown in **Fig. 6** and the attributes of each expert node in the network is shown in **Table 1**.

In the experiment, a requester q and a multi-domain problem p_i were randomly generated for 100 times where the number of knowledge domains for p_i ranges from 2 to 5. The probability of the link between a pair-node was set as 30%. In order to imitate the realistic environment, the coefficients in Eq. (11) were also set randomly in each multi-domain problem p_i. In the experiment, we compared the KCCD approach with the Dynamic Community (DC) approach [18] and the comparison results are introduced in the following paragraphs.

6.3.2 Experimental Results

In Experiment 1, we mainly compared following four aspects of the KCCD approach and the DC approach. Due to the low probability of the link between a pair-node, the community connectivity is very small, so we will not use it as an evaluation factor in this experiments.

(1) **Average Catastrophe Level**: The average catastrophe level of a KCC formed by using the KCCD approach and the DC approach are shown in **Table 2**. From Table 2, it can be seen that the average community size in the KCCD approach is smaller than the number of knowledge domains in $p_i.TPC$, which tends to have positive influence of a KCC. However, the average catastrophe level of the DC approach is larger than zero, which decreases the performance of a KCC. Normally, a smaller community needs less communication overheads. Hence, the communities formed by using the KCCD approach can promise more efficient interactions among community members.

(2) **Knowledge Domain Coverage**: In the experiment, we found that the KCCD approach can produce "full coverage" communities for requested multi-domain problems (**Table 3**). However, 6% proposed multi-domain problems cannot be solved by the communities formed in the DC approach.

(3) **Average Knowledge Level**: The average knowledge level of the KCCD approach and the DC approach were shown in **Table 4**. From Table 4, it can be seen that the average knowledge level of communities formed in the KCCD approach is lower than that of the communities formed in the DC approach. This is because that the DC approach only calculates the highest domain knowledge level in each required knowl-

Table 1 Node attributes.

Knowledge Domain	Expert ID	Rank	Node ID
d_0	e_{00}	1	a_0
d_0	e_{01}	4	a_3
d_0	e_{02}	5	a_{11}
d_0	e_{03}	3	a_{20}
d_0	e_{04}	2	a_4
d_1	e_{10}	5	a_5
d_1	e_{11}	1	a_6
d_1	e_{12}	4	a_7
d_1	e_{13}	3	a_8
d_1	e_{14}	2	a_4
d_2	e_{20}	3	a_{10}
d_2	e_{21}	2	a_{11}
d_2	e_{22}	1	a_{12}
d_2	e_{23}	5	a_1
d_2	e_{24}	4	a_3
d_3	e_{30}	2	a_{15}
d_3	e_{31}	3	a_{16}
d_3	e_{32}	5	a_{17}
d_3	e_{33}	1	a_{12}
d_3	e_{34}	4	a_3
d_4	e_{40}	5	a_{20}
d_4	e_{41}	2	a_8
d_4	e_{42}	3	a_3
d_4	e_{43}	1	a_{23}
d_4	e_{44}	4	a_{24}

Table 2 Average catastrophe level.

	Average Catastrophe Level
KCCD	−0.38
DC	0.59

Table 3 Knowledge domain coverage.

	Knowledge Domain Coverage
KCCD	1.0
DC	0.94

Table 4 Average knowledge level.

	Average Knowledge Level
KCCD	0.91
DC	1.32

© 2014 Information Processing Society of Japan

Table 5 Average connective level.

	Average Connective Level
KCCD	0.74
DC	0.53

Table 6 Average community performance.

	Average Community Performance
KCCD	2.12
DC	1.32

edge domain in $p_i.TPC$ without taking care of the actual community size of DC and other knowledge domain rank. However, the KCCD approach considers all members' ranks in the required knowledge domains in $p_i.TPC$.

(4) **Average Connective Level**: As mentioned earlier, the connective level is another important factor for evaluating the efficiency of knowledge collaboration. The average connective levels of formed communities in the two approaches are shown in **Table 5**. From the table, it can be seen that the KCCD approach has a higher connective level than that of the DC approach.

From the above comparison, it can be seen that the DC approach had better performance in only one aspect (i.e., the Average Knowledge Level), where the KCCD approach has better results in other aspects. The average performance of formed communities in the two approaches are shown in **Table 6**. We can claim that the KCCD approach can perform better in centralized networks than the DC approach, which is 2.12 and 1.32 respectively.

6.4 Experiment 2

Experiment 2 was conducted for evaluating the 2-Step approach in distributed network and the DC approach is also adopted in the experiments for comparison. In terms of distributed network, there is no global view, so we only can expand the network and explore the node from the neighborhood or available nodes within current potential community *PC* as defined in Definition 7 in Section 3. The potential community *PC* network expends from only requester q, and then all q's neighbors are added into. In Experiment 2, we fixed the size of the network, but applied different connection probabilities. It is mainly focused on the evaluation of performance of the 2-Step approach and the DC approach in networks with different connectivities. In this experiment, we mainly evaluate three factors: success rate, connective level and catastrophe level. As mentioned in Experiment 1, the knowledge level of DC approach does not consider the actual community size of DC, so we still do not take it into account in Experiment 2.

6.4.1 Experimental Setup

In this experiment, we use the similar dataset as Experiment 1. There are five knowledge domains and each domain has 20 expert candidates. Therefore, the number of nodes in the network was set to 100, and the probability of the link between a pair-node was set in different values, i.e., $0.01, 0.02, \ldots, 0.2$, where different probability of links between a pair-node indicated different connectivities in the network. For each connection probability, p_i and q were randomly created for 100 times. At each time, the 2-Step approach and the DC approach were used to discover a community for solving p_i, respectively. The smaller the probability of links between a pair-node is, the less the community connectivity contributes to the performance of KCC, so we will not use it as an evaluation factor in this experiments.

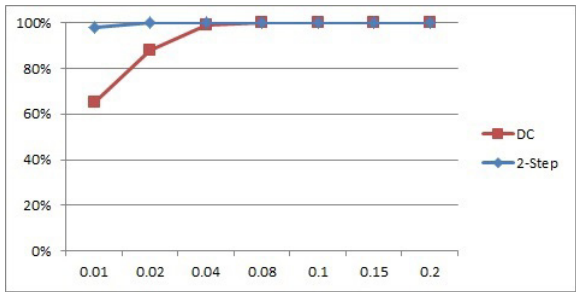


Fig. 7 The success rate in different connective networks.

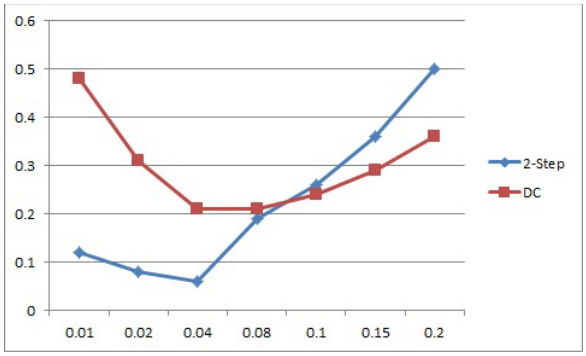


Fig. 8 Average connective level in different connective networks.

6.4.2 Experimental Results

The success rates of the 2-Step approach and DC approach in different Connective Networks are shown in **Fig. 7**. The X axis in Fig. 7 represents the connection probability, while the Y axis indicates the percentage of success rate to discover a proper KCC for solving p_i. From Fig. 7, it can be seen that when connection probability is less than 0.08, the 2-Step approach had better performance compared with the DC approach. For both approaches, higher connection probabilities could lead to better performance. The 2-Step approach could achieve 100% success rate after the probability reached 0.04. The DC approach could reach 100% success rate after the probability was increased to 0.08.

The average connective levels of the 2-Step approach and DC approach in different Connective Networks are shown in **Fig. 8**. The X axis in Fig. 8 represents the connection probability, while the Y axis indicates the value of average connective level of a proper KCC for solving p_i. From Fig. 8, it can be seen that once both two approaches can achieve 100% success rate after the probability reached 0.08, the 2-Step approach not only had higher average connective level than the DC approach, but also increased more dramatically.

Figure 9 shows the average catastrophe levels of the 2-Step approach and DC approach in different connective networks. The X axis in Fig. 7 represents the connection probability, while the Y axis indicates the value of the average catastrophe level of a proper KCC for solving p_i. From Fig. 8, it can be seen that the average catastrophe level of the 2-Step approach always maintained

© 2014 Information Processing Society of Japan

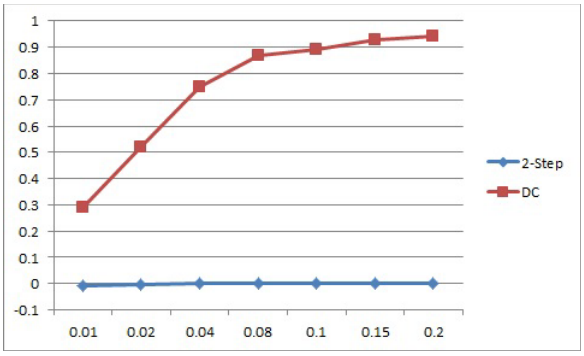


Fig. 9 Average catastrophe level in different connective networks.

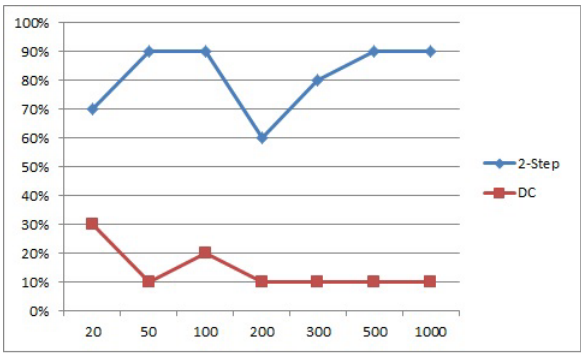


Fig. 10 The success rate in different size networks.

at 0. However, the average catastrophe level of the DC approach increased significantly by rising the connection probability from 0.01 to 0.2.

6.5 Experiment 3

In terms of the distributed networks, Experiment 3 compares the succuss rate of 2-Step approach and DC approach in different sizes of networks.

6.5.1 Experimental Setup

A random network, which contains different numbers of nodes, is generated in the experiment. A random knowledge domain and the knowledge level are assigned to each node. Then a multi-domain problem p_i is created and assigned to a random node q. q then becomes a requester, and it may or may not have a required expertise for solving p_i.

In Experiment 3, node numbers of the network (n) were set to 20, 50, 100, 200, 300, 500 and 1,000, and the probability of two nodes, which can be connected, was set to $(2/n)\%$. Then p_i and q were generated/selected for ten times in each size of networks. The goal of this experiment was to compare the performances of the 2-Step approach and the DC approach in different size of networks.

6.5.2 Experimental Results

The result is shown in **Fig. 10**. The X axis in Fig. 10 represents the number of nodes in a network, and the Y axis represents the success rate to discover a KCC. From Fig. 10, it can be found that in all cases, the 2-Step approach had better performance than the DC approach. The best result for DC was 30%, appeared in the experiment with 20 nodes. When a network contains more than 200 nodes, the best result of the DC approach reduced to 10%. The experimental result obviously shows that the DC had better performance in centralized networks. On the other side, the 2-Step approach could always have better performance with above 50% success rate. Also, the experimental result indicates that the performance of the 2-Step approach was fractionally impacted by the size of network. Due to the low success of DC approach, it is meaningless to evaluate the other factors.

7. Conclusion and Future Work

KCC discovery is an important problem for collective intelligence systems. To discover a KCC successfully, the consideration of a network structure is necessary, and the consideration of knowledge collaboration is also essential. In this paper, two approaches (i.e., the KCCD approach and the 2-Step approach) for mining knowledge collaborative communities are introduced. The KCC approach is used for centralized networks to discover suitable KCCs for solving multi-domain problems. The 2-Step approach is designed for distributed networks. Comparing with some existing community mining approaches, the proposed approaches consider more factors when forming KCCs, and can perform better in networks with different scalabilities and connectivities. This has been supported by the experimental results.

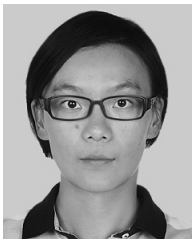
As mentioned in Section 6, we do not conduct experiments on realistic datasets, because both KCCD and 2-Step approach require four basic attributes: Knowledge Domain, Expert, Knowledge Level and Collaboration History, in the dataset. This might limit their application in some domains. Therefore, in future work, we will do more research to mine such attributes from general social networks without requiring further information to extend the usage scope of our KCCD and 2-Step approaches.

References

[1] Adler, P. and Heckscher, C., Collaborative community, *Ask Magazine*, Vol.23, pp.41–45 (2006).
[2] Coscia, M., Giannotti, F. and Pedreschi, D.: A classification for community discovery methods in complex networks, *Statistical Analysis and Data Mining*, Vol.4, No.5, pp.512–546 (2011).
[3] Datta, A., Yong, J.T.T. and Ventresque, A.: T-recs: Team recommendation system through expertise and cohesiveness. *Proc. 20th International Conference Companion on World Wide Web*, pp.201–204, ACM (2011).
[4] Dorn, C., Skopik, F., Schall, D. and Dustdar, S.: Interaction mining and skill-dependent recommendations for multi-objective team composition, *Data & Knowledge Engineering*, Vol.70, No.10, pp.866–891 (2011).
[5] Faraj, S., Jarvenpaa, S.L. and Majchrzak, A.: Knowledge collaboration in online communities, *Organization Science*, Vol.22, No.5, pp.1224–1239 (2011).
[6] Lih, A.: Wikipedia as participatory journalism: Reliable sources? metrics for evaluating collaborative media as a news resource, *Proc. 5th International Symposium on Online Journalism*, Austin, USA (April 2004), available from ⟨http://online.journalism.utexas.edu/2004/papers/wikipedia.pdf⟩ (accessed 2013-05-19).
[7] Lin, Y.-R., Sundaram, H., Chi, Y., Tatemura, J. and Tseng, B.L.: Blog community discovery and evolution based on mutual awareness expansion, *Proc. IEEE/WIC/ACM International Conference on Web Intelligence*, pp.48–56, IEEE Computer Society (2007).
[8] Malone, T.W., Laubacher, R. and Dellarocas, C.: The collective intelligence genome, *IEEE Engineering Management Review*, Vol.38, No.3, p.38 (2010).
[9] Newman, M.E.J.: Detecting community structure in networks, *The European Physical Journal B - Condensed Matter and Complex Systems*, Vol.38, No.2, pp.321–330 (March 2004).
[10] Ohira, M., Ohoka, T., Kakimoto, T., Ohsugi, N. and Matsumoto, K.: Supporting knowledge collaboration using social networks in a large-scale online community of software development projects, *12th Asia-Pacific Software Engineering Conference*, pp.835–840, Taipei, Taiwan (Dec. 2005).

© 2014 Information Processing Society of Japan

[11] Parthasarathy, S., Ruan, Y. and Satuluri, V.: Community discovery in social networks: Applications, methods and emerging trends, *Social Network Data Analytics*, pp.79–113, Springer (2011).
[12] Patil, R., Fikes, R., Patel-Schneider, P., McKay, D.P., Finin, T., Gruber, T. and Neches, R.: The darpa knowledge sharing effort: A progress report, Bernhard Nebel, editor, *Proc. 3rd International Conference on Principles of Knowledge Representation And Reasoning*, Morgan Kaufman, Cambridge, Massachusetts, USA (Oct. 1992), available from ⟨http://ebiquity.umbc.edu/_file_directory_/papers/280.pdf⟩ (accessed 2013-05-19).
[13] Shekar, B. and Natarajan, R.: A transaction-based neighbourhood-driven approach to quantifying interestingness of association rules, *4th IEEE International Conference on Data Mining, ICDM'04*, pp.194–201, IEEE (2004).
[14] Wang, F., Li, T., Wang, X., Zhu, S. and Ding, C.: Community discovery using nonnegative matrix factorization, *Data Mining and Knowledge Discovery*, Vol.22, No.3, pp.493–521 (2011).
[15] Watts, D.J.: *Six Degrees: The Science of a Connected Age*, W.W. Norton & Company, 1st edition (Feb. 2003).
[16] Weiss, A.: The power of collective intelligence, *netWorker*, Vol.9, No.3, pp.16–23 (Sep. 2005).
[17] Ye, Y., Nakakoji, K. and Yamamoto, Y.: The economy of collective attention for situated knowledge collaboration in software development, *Proc. 2008 International Workshop on Cooperative and Human Aspects of Software Engineering*, pp.109–112, ACM, Leipzig, Germany (May 2008).
[18] Ye, Y., Yamamoto, Y. and Kishida, K.: Dynamic community: A new conceptual framework for supporting knowledge collaboration in software development, *Proc. 11th Asia-Pacific Software Engineering Conference*, Busan, Korea, pp.472–481 (Nov. 2004).
[19] Ye, Y., Yamamoto, Y. and Nakakoji, K.: Expanding the knowing capability of software developers through knowledge collaboration, *International Journal of Technology, Policy and Management*, Vol.8, No.1, pp.41–58 (2008).
[20] Yu Z. and Fan, Z.: Simulation and study of catastrophe phenomenon in knowledge collaboration complexity network, *Proc. 2008 2nd International Symposium on Intelligent Information Technology Application*, Vo.1, pp.518–521, IEEE Computer Society, Shanghai, China (Dec. 2008).



Jing Jiang is a Master of Philosophy student at Auckland University of Technology. Her research interests are in the areas of trust management, service recommendation, community detection and data mining.

Quan Bai received his Ph.D. in computer science at the University of Wollongong in 2007. He is presently a senior lecture at Auckland University of Technology. He specialises in multi-agent coordination, trust-based computing and service composition.

Minjie Zhang received her B.Sc. degree from Fudan University, P.R. China, in 1982 and Ph.D. degree in computer science degree from the University of New England, Australia, in 1996. Now, she is an associate professor of computer science at the University of Wollongong, Australia. Her research interests include multiagent systems, agent-based simulation, and modeling in complex domains. She is a member of IEEE.

Shaojie Yuan received his MCompSc-Res and GDipIT from the University of Wollongong in 2010 and 2006, respectively. Currently he is an IT officer within the Australian Council on Healthcare Standards. His research interest is in knowledge discovery, collaborative network, tasks allocation in multi-domain problem solving and data mining.

© 2014 Information Processing Society of Japan