Decision Support Systems 50 (2011) 370–386

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

Mining group-based knowledge flows for sharing task knowledge

Duen-Ren Liu *, Chin-Hui Lai

Institute of Information Management, National Chiao Tung University, Taiwan

ARTICLE INFO

Article history:
Received 16 December 2009
Received in revised form 9 September 2010
Accepted 26 September 2010
Available online 1 October 2010

Keywords:
Knowledge flow
Group-based knowledge flow
Knowledge graph
Knowledge sharing
Data mining
Topic
Task

ABSTRACT

In an organization, knowledge is the most important resource in the creation of core competitive advantages. It is circulated and accumulated by knowledge flows (KFs) in the organization to support workers' task needs. Because workers accumulate knowledge of different domains, they may cooperate and participate in several task-based groups to satisfy their needs. In this paper, we propose algorithms that integrate information retrieval and data mining techniques to mine and construct group-based KFs (GKFs) for task-based groups. A GKF is expressed as a directed knowledge graph which represents the knowledge referencing behavior, or knowledge flow, of a group of workers with similar task needs. Task-related knowledge topics and their relationships (flows) can be identified from the knowledge graph so as to fulfill workers' task needs and promote knowledge sharing for collaboration of group members. Moreover, the frequent knowledge referencing path can be identified from the knowledge graph to indicate the frequent knowledge flow of the workers. To demonstrate the efficacy of the proposed methods, we implement a prototype of the GKF mining system. Our GKF mining methods can enhance organizational learning and facilitate knowledge management, sharing, and reuse in an environment where collaboration and teamwork are essential.

1. Introduction

In an organization, knowledge is the most important resource used to create core competitive advantages. Generally, knowledge and expertise in an organization are codified in textual documents, e.g., papers, manuals and reports, and preserved in a knowledge database. Large amounts of such codified knowledge are circulated and accumulated in an organization to support knowledge workers engaged in diverse tasks and activities. To preserve, share and reuse these valuable assets, organizations need to adopt appropriate knowledge management strategies to support knowledge workers intelligently [26,28]. Knowledge management, which is widely utilized in organizations, is important for preserving and sharing knowledge efficiently [14,36].

Knowledge management systems (KMS) facilitate the preservation, reuse and sharing of knowledge, and also support collaboration among workers. Based on a task's specifications and the process-context of the task, the *KnowMore* system [1] provides context-aware knowledge retrieval and delivery functions to support the procedural activities of workers. The task-based *K-support* system [23,24,38] provides knowledge support adaptively to meet a worker's dynamic information needs by analyzing his/her access behavior. Moreover, knowledge workers may cooperate with each other to accomplish their tasks. Task knowledge can be transmitted, shared and accumulated from one team member/process to another. Therefore, working knowledge flows between workers in an organization, while process knowledge flows between various tasks [39,41]. Zhuge [39] proposed a management mechanism for knowledge sharing, and integrated the knowledge flow with the workflow to assist workers. Furthermore, knowledge flows (KFs) can be used to represent the long-term evolution of workers' information needs [22]. Based on those needs, the knowledge flow-based document recommendation method proactively delivers task-relevant topics and documents to the workers.

To work more efficiently, workers conducting similar (relevant) tasks or cooperative tasks generally have similar task-related information needs, and can form a group to facilitate knowledge reuse, cooperation and sharing. A group of workers can be identified explicitly by specifying which tasks are similar or cooperative tasks so that the knowledge workers conducting those tasks can form a group. Alternatively, a group of workers can be identified implicitly based on their referencing behavior as proposed in this work. Under this approach, it is assumed that workers with similar knowledge flows work on similar or cooperative tasks, and thus have similar task-related information needs. Workers in the same group may not have exactly the same knowledge flows, but they may adopt similar referencing behavior when performing tasks. Common or frequently referenced task-related knowledge items in group members' knowledge flows represent the core knowledge that the workers require to perform their tasks. Since such knowledge is implicit in the knowledge flows of a group of workers, it cannot be represented solely by an individual's personal knowledge flow. To facilitate

* Corresponding author.
E-mail address: dliu@mail.nctu.edu.tw (D.-R. Liu).

0167-9236/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2010.09.004

knowledge reuse, cooperation and sharing, discovering such common/frequently accessed knowledge is important for workers who perform similar tasks and have similar referencing behavior patterns. Accordingly, we propose group-based knowledge flow mining algorithms that model a group's frequent referencing behavior by identifying frequent topics of interest, major referencing behavior patterns, and the long-term evolution of the group's information needs.

Because the information needs of workers or groups may change over time, it is difficult to model their knowledge referencing behavior. Obviously, recognizing those needs, delivering the required task-related knowledge, and facilitating knowledge sharing/reuse are important issues that must be addressed in a knowledge intensive organization. However, to the best of our knowledge, there is no appropriate approach for analyzing and constructing KFs from the perspective of a group's information needs; and very little research effort has been expended on KF mining for task-based groups.

To address the above research gaps, we propose algorithms that integrate information retrieval and data mining techniques for mining and constructing the KFs of groups. In our previous work [22], we presented a KF mining approach that identifies each knowledge worker's KF. Here, we extend that approach and focus on discovering group-based knowledge flows (GKFs). From the group-based knowledge flow, workers can discover the knowledge frequently accessed by group members. They can also share their own knowledge with others to facilitate knowledge reuse, cooperation and sharing. From the perspective of a task's execution, a group-based knowledge flow will be an important knowledge asset when conducting a task similar or relevant to the performance of those tasks from which the group-based knowledge flow was derived. For example, a group-based knowledge flow derived from the knowledge flows of several researchers working on Social Network Analysis (SNA) related tasks would be helpful to a new researcher who has just started working on an SNA-related research task.

Specifically, we discover a group's KF from the KFs of workers who exhibit similar knowledge referencing behavior patterns. First, based on the workers' logs, we analyze each worker's referencing behavior when acquiring task-related knowledge, and then construct his/her KF as described in [22]. We then use a clustering method to identify a group of workers with similar task-related information needs based on the workers' KF similarities. Workers in the same group generally need similar codified knowledge to perform their tasks. In addition, workers in the same group may adopt different behavior when referencing task-related knowledge. Therefore, we propose GKF mining algorithms to discover the referencing behavior patterns of a group of workers. Second, we apply the concepts of graph theory to visualize the GKF as a knowledge graph in which a vertex and an edge indicate, respectively, a topic domain and a direct flow relation between two topic domains. Task-related knowledge topics and their relationships (flows) can be identified from the knowledge graph to fulfill workers' task needs when they reference task-relevant knowledge.

Frequent knowledge referencing paths (patterns) can also be identified based on the edge weights in the graph. The paths represent the workers' frequent knowledge referencing behavior and important knowledge flows in the group. Finally, to demonstrate the efficacy of the proposed method, we implement a prototype system for mining the GKF of a group of workers. The system provides useful functions that allow users to simplify the KF mining process and visualize KFs graphically.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of related works. In Sections 3 and 4, we introduce our proposed algorithms for mining a group knowledge flow (GKF), which is then used to construct a knowledge graph based on the workers' referencing behavior. Section 5 describes a prototype system that we implement based on the proposed algorithms. Then, in Section 6, we summarize our conclusions and consider future research directions.

2. Background and related work

In this section, we consider the related work of our research, including the concepts of knowledge flow, information retrieval and task-based knowledge support, document clustering and process mining.

2.1. Knowledge flow

Knowledge flows among people and processes to facilitate knowledge sharing and reuse. The concept of knowledge flow has been applied in various domains, e.g., scientific research, communities of practice, teamwork, industry, and organizations [5,21,40]. Scholarly articles represent the major medium for disseminating knowledge among scientists to inspire new ideas [40]. A citation implies that there is a knowledge flow between the citing article and the cited article. Such citations form a knowledge flow network that enables knowledge to flow between different scientific projects to promote interdisciplinary research and scientific development.

A knowledge flow model enhances the effectiveness of teamwork by accumulating and sharing knowledge among team members to facilitate peer-to-peer knowledge sharing [39]. To improve the efficiency of teamwork, Zhuge [41] proposed a pattern-based approach that combines codification and personalization strategies in order to design an effective knowledge flow network. Kim et al. [21] proposed a knowledge flow model combined with a process-oriented approach to capture, store, and transfer knowledge. Knowledge flows in communities of practice help members share their knowledge and experience about a specific domain to complete certain tasks [29]. Luo et al. [25] propose the discovery of textual knowledge flow based on the semantic link network. A context-based knowledge flow is proposed to reflect the major characteristics of a knowledge flow [13]. In an organization, knowledge workers normally have various information needs over time when performing tasks. Thus, a knowledge flow is defined from the perspective of a worker's information needs to represent the evolution of referencing behavior and the knowledge accumulated for a specific task [22].

2.2. Information retrieval and task-based knowledge support

Information retrieval (IR) facilitates access to specific items of information [7]. The vector space model [30] is typically used to represent documents as vectors of index terms, where the weights of the terms are measured by the *tf-idf* approach; *tf* denotes the occurrence frequency of a particular term in the document, while *idf* denotes the term's inverse document frequency. Terms with higher *tf-idf* weights are used as discriminating terms to filter out common terms. The weight of a term i in a document j, denoted by $w_{i,j}$, is expressed as follows:

$$w_{i,j} = tf_{i,j} \times idf_i = tf_{i,j} \times \left(\log_2 \frac{N}{n} + 1\right), \tag{1}$$

where $tf_{i,j}$ is the frequency of term i in document j, idf_i is measured by $(\log_2 N/n) + 1$, N is the total number of documents in the collection, and n is the number of documents in which term i occurs.

Information retrieval techniques coupled with workflow management systems (WfMS) have been used to support proactive delivery of task-specific knowledge based on the context of tasks within a process [2]. For example, the *KnowMore* system [1] provides context-aware delivery of task-specific knowledge; while the *Kabiria* system considers the operational context of task-associated procedures to help knowledge workers retrieve knowledge-based documents [6].

Information filtering with a similarity-based approach is often used to locate knowledge items relevant to the task-at-hand. The discriminating terms of a task are usually extracted from a knowledge item/task to form a task profile, which is then used to model a worker's information needs. Holz et al. [15] proposed a similarity-based approach that organizes desktop documents and proactively delivers task-specific information to the user; and Liu et al. [23,24] presented a *K-Support* system to provide effective task support in a task-based work environment.

2.3. Hierarchical document clustering and CLIQUE clustering methods

Document clustering or unsupervised document classification is used in many applications. Most document clustering methods apply pre-processing steps to the document set and represent each document as a vector of index terms. To cluster similar documents, the similarity between documents is usually measured by the cosine measure [7,35], which computes the cosine of the angle between the documents' corresponding feature vectors. Two documents are considered similar if the cosine similarity value is high. The cosine similarity of two documents, X and Y, is $simcos(X,Y) = \frac{\vec{X}\cdot\vec{Y}}{\|\vec{X}\|\|\vec{Y}\|}$, where $\vec{X}$ and $\vec{Y}$ are the respective feature vectors of X and Y.

Agglomerative hierarchical clustering [18,20] is a popular document clustering method. In this work, we use the single-link clustering method [17] to cluster codified knowledge (documents) into topic domains. The single-link method computes the similarity between two clusters, which is equal to the greatest similarity between any document in one cluster and any document in the other cluster.

We also apply the CLIQUE clustering method [3,17] to derive worker groups. CLIQUE starts with the definition of a unit-elementary rectangular cell in a subspace and uses a bottom-up approach to find units whose densities exceed a threshold. All the subspaces are sorted by their coverage and those with less coverage are pruned. Therefore, a cluster is defined as a maximal set of connected dense units.

2.4. Process mining

In a workflow system, a process mining technique is used to extract the description of a structural process from a set of real process executions [33]. It then infers the relations between the tasks/activities and generates a process model from event-based data (log data) automatically [4,32,34]. The relations between processes (tasks/activities) are defined as causal relations and parallel relations [31], and are modeled by a directed graph [4,12] or an instance graph [34]. Because a workflow log contains information about workflow processes, a loop may occur in a process. Most process mining algorithms assume that loops do not exist [12,34]. However, some algorithms have been proposed to handle the problem of process loops [4,11,33]. For example, Agrawal et al.'s algorithm [4] builds a general directed graph with cycles for mining process models from the logs of executed processes. The algorithm labels multiple instances of the same activity with different identifiers to differentiate them in the workflow graph. Vertices with different instances of the same activity form an equivalent set and can be merged to form one vertex. A directed edge is added if there is an edge between two vertices of different equivalent sets.

Process mining is used in various applications. Discovering frequently occurring temporal patterns in process instances facilitates intelligent and automatic extraction of useful knowledge to support business decision-making [16]. Similarly, data mining techniques are exploited in workflow management contexts to mine frequent workflow execution patterns [12]. The sequence of activities within a process, the execution cost and the reliability of the process can be predicted by using the process path mining technique [8]. Based on the process patterns and process paths, unexpected and useful knowledge about the process is extracted to help the user make appropriate decisions. In addition, a formal approach is proposed to discover process models from business policies [37].

3. Group-based knowledge flow mining

A knowledge flow (KF) represents a knowledge worker's long-term information needs and accumulated task-related knowledge when he/she performs a task. In a previous work, we proposed a KF mining method to obtain each worker's KF from his/her work log [22]. We also presented document recommendation methods to support workers' in the execution of tasks and facilitate knowledge sharing in an organization. In the context of collaboration, workers usually have similar referencing behavior patterns, in which they share common topics or documents they find useful, or they reference task-related knowledge in a similar order. To model the common referencing behavior of a group, we propose a method for mining a group-based knowledge flow (GKF) from the KFs of a group of workers.

Fig. 1 provides an overview of the proposed method for mining GKFs. Based on the workers' KFs, workers with similar topic-level KFs are clustered together to form a task-based group. Members of the group have task-related knowledge or similar referencing behavior in terms of the topics of interest and the order the topics were referenced in their KFs. To identify similar referencing behavior from the KFs, we propose KF mining algorithms based on process mining and graph theory to discover a group's knowledge flow. The algorithms identify common information needs and referencing patterns from the KFs of a group of workers, and then build a group-based knowledge flow (GKF) model. Then, a frequent knowledge path is identified from the model to represent the referencing (learning) patterns of the group and to support group members, especially the novices, in accessing and learning a group's knowledge. In this work, we focus on two issues: 1) how to construct a group-based knowledge flow (GKF) model for a group of knowledge workers with similar KFs; and 2) how to identify frequent referencing patterns (paths) from the GKF model.

A group-based knowledge flow (GKF) is derived from the knowledge flows (KFs) of the group members. Mining and constructing the GKF is the main focus of this work. As such, it extends our previous work on mining an individual's knowledge flows [22]. To ensure that the explanation of our proposed GKF mining method is clear, we provide some fundamental definitions and concepts involved in generating knowledge flows. Thus, we provide a summary of the fundamental definitions and methods used to generate a worker's codified level and topic-level KFs in Section 3.1. In Section 3.2, we cluster workers with similar KFs into groups, based on the KFs described in Section 3.1. The information in Sections 3.1 and 3.2 is a summary of the fundamental concepts of knowledge flow mining discussed in our previous paper [22]. The concepts provide the basis for this work. Readers may refer to our previous publication for further details. In Section 3.3, we describe the steps of the proposed group-based KF mining. Moreover, several important concepts and features used in the GKF mining algorithms are presented.

3.1. Knowledge flow mining

From the perspective of information needs, a worker's knowledge flow (KF) represents the evolution of his/her information needs and preferences during a task's execution. Workers' KFs are identified by analyzing their knowledge referencing behavior based on their historical work logs, which contain information about previously executed tasks, task-related documents and when the documents were accessed.

A KF consists of two levels: a codified level and a topic level. The knowledge in the codified-level indicates the knowledge flow

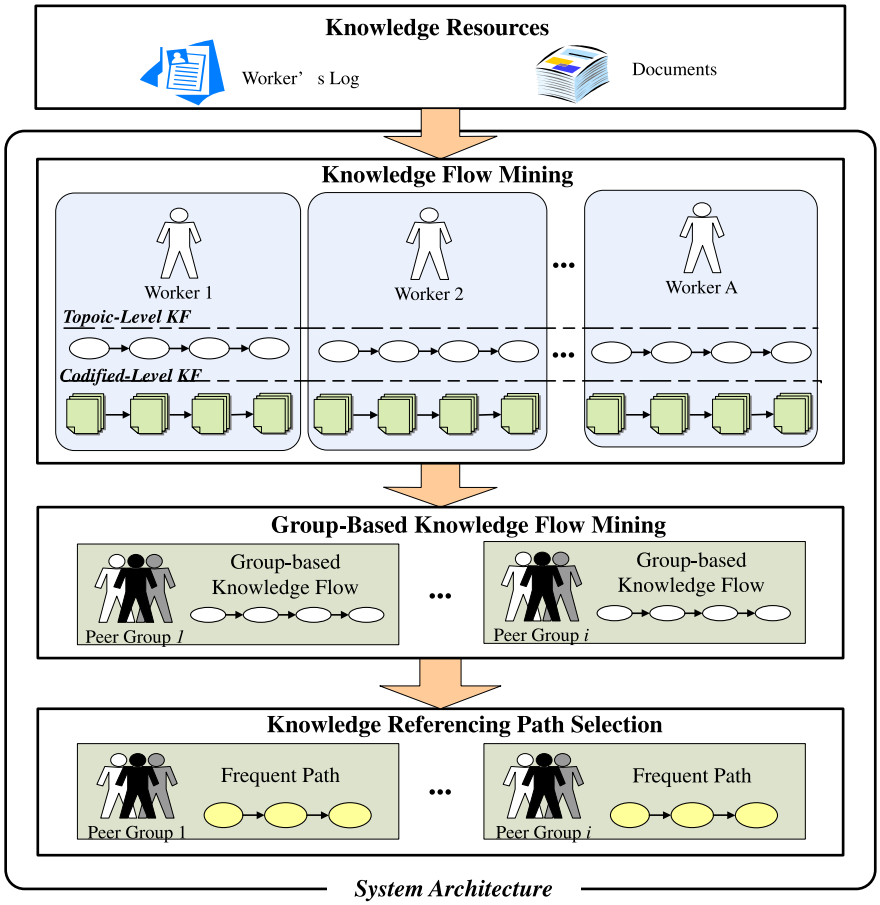

Fig. 1. An overview of mining group-based knowledge flows.

between documents based on the access time. In most situations, the knowledge obtained from one document prompts a knowledge worker to access the next relevant document (codified knowledge). Hence, the task-related documents are sorted by their access time to obtain a document sequence as the codified-level KF. Documents are clustered into topic domains by using the agglomerative hierarchical clustering method described in Section 2.3. Each topic may contain several task-related documents. The codified-level KF can be abstracted to form a topic-level KF, which represents the transitions between various topics. Formally, we define the knowledge flows as follows.

Definition 1. Knowledge flow (KF).

Let the knowledge flow of a worker, w, for a specific task be $KFlow_w = \{TKF_w, CKF_w\}$, where TKF_w^v is the worker's topic-level KF and CKF_w is his/her codified-level KF.

Definition 2. Codified-level KF.

A codified-level KF is a time-ordered sequence arranged according to the access times of the documents it contains. Formally, it is defined as

$$CKF_w = <d_{w,}^{t_1}, d_w^{t_2}, \cdots, d_w^{t_f}> \text{ and } t_1<t_2<\cdots<t_f,$$

where $d_w^{t_j}$ denotes the document that the worker w accessed at time t_j for a specific task. Each document can be represented by a document profile, which is an n-dimensional vector containing weighted terms that indicate the key content of the document described in Section 2.2.

Definition 3. Topic-level KF.

A topic-level KF is a time-ordered topic sequence derived by mapping documents in the codified-level KF to corresponding topics. Formally, it is defined as

$$TKF_w = <TP_w^{t_1}, TP_w^{t_2}, \cdots, TP_w^{t_f}>, t_1<t_2<\cdots<t_f,$$

where $TP_w^{t_j}$ denotes the corresponding topic of the document that worker w accessed at time t_j for a specific task. Each topic is represented by a topic profile, which is an n-dimensional vector containing weighted terms that indicate the key content of the topic. The topic profile of a topic is derived from the document profiles of documents contained in that topic by using the centroid approach.

By analyzing a worker's referencing behavior for a specific task, the corresponding knowledge flow is derived by the knowledge flow extraction method. The codified-level KF is extracted from the documents recorded in the worker's work log. The documents are arranged according to the times they were accessed to obtain a document sequence. The topic-level KF, which is derived by mapping documents in the codified-level KF of a specific task into corresponding clusters, is represented by a topic sequence.

3.2. Clustering similar workers based on their knowledge flows

To find a target worker's neighbors, his/her topic-level KF is compared with those of other workers to compute the similarity of their KFs. The resulting similarity measure indicates whether the KF referencing behavior of two workers is similar. Since the KFs are sequences, the sequence alignment method [9,27], which computes

the cost of aligning two sequences, can be used to measure the similarity of two KF sequences. Based on this concept, we use a hybrid similarity measure, comprised of the KF alignment similarity and the aggregated profile similarity, to evaluate the similarity of two workers' KFs, as shown in Eq. (2).

$$sim\left(TKF_i, TKF_j\right) = \alpha \times sim_a\left(TKF_i, TKF_j\right) + (1-\alpha) \times sim_P\left(AP_i, AP_j\right), \tag{2}$$

where $sim_a(TKF_i, TKF_j)$ represents the KF alignment similarity, $sim_p(AP_i, AP_j)$ represents the aggregated profile similarity, and α is a parameter used to adjust the relative importance of the two types of similarity. Here, we give a brief explanation. Further details are provided in [22].

The KF alignment similarity is comprised of two parts: the KF alignment score, which measures the topics in the sequence; and the join coefficient, which estimates the topic's coverage in two compared topic-level KFs. We modify the sequence alignment method [9] to derive the KF alignment score. We also estimate the overlap of the topics in two compared topic-level KFs by using Dice's coefficient [35]. The rationale is that if the topic overlap is high, the KF alignment similarity of the two compared KFs will also be high. The KF alignment similarity, $sim_a(TKF_i, TKF_j)$, is defined as follows:

$$sim_a\left(TKF_i, TKF_j\right) = Norm(\eta) \times \frac{2 \times |T_i \cap T_j|}{|T_i| + |T_j|}, \tag{3}$$

where TKF_i and TKF_j are the topic-level KFs of workers i and j respectively; η is the KF alignment score; $Norm$ is a normalization function used to transform the value of η into a number between 0 and 1; T_i and T_j are the sets of topics in TKF_i and TKF_j respectively; and $T_i \cap T_j$ is the intersection of topics common to TKF_i and TKF_j.

The aggregated profile similarity, defined as $sim_p(AP_i, AP_j)$, computes the similarity of two workers' KFs based on their aggregated profiles; AP_i and AP_j are the vectors of the aggregated profiles of workers i and j respectively. We use the cosine formula to calculate the similarity between two aggregated profiles. The aggregated profile of a worker i is defined as Eq. (4).

$$AP_i = \sum_{t=1}^{T} tw_{t,T} \times DP_t, \tag{4}$$

where $tw_{t,T}$ is the time weight of a document referenced at time t in the KF; T is the index of the time the worker accessed the most recent documents in his KF; and DP_t is the profile of the document referenced by worker i at time t. The aggregation considers the time decay effect of the documents. Hence, if a document was referenced in the recent past, it is given a higher time weight. The time weight of each document profile is defined as $tw_{t,T} = \frac{t-St}{T-St}$, where St is the start time of the worker's KF.

In this paper, we use the CLIQUE clustering method [3,17] as described in Section 2.3 to cluster knowledge workers based on a similarity matrix of their KFs. Each entry in a similarity matrix represents the degree of KF similarity between two workers, derived by Eq. (2). Workers in the same cluster are highly connected with each other because they have similar referencing behavior and information needs in topic domains. To identify each group's GKF, we apply our group-based knowledge mining method to process the clustering results.

3.3. The group-based knowledge flow mining process

The proposed method comprises three phases: worker clustering, group-based knowledge flow (GKF) mining, and identifying knowledge-referencing paths, as shown in Fig. 2. Based on the extracted KFs (Section 3.1), the worker clustering step (Section 3.2) is used to cluster workers with similar KFs as an interest group because they have similar information needs and task-related knowledge to fulfill their tasks. Given the KFs of the workers, we formalize the GKF model to represent the group's information needs by applying the proposed GKF mining algorithms, as described in Section 4. The group-based knowledge flow (GKF) represents the information needs and common referencing behavior of a group of workers. Based on GKF, workers can share their task knowledge to complete the target task. Moreover, managers can comprehend the information needs of workers and groups to provide knowledge support adaptively.

The GKF is represented by a directed acyclic graph comprised of vertices and edges. Each vertex denotes a topic in a KF, while each directed edge represents the referencing order of two topics. We use graph theory to model a GKF. A GKF graph models the relations between topics, the direction of the knowledge flow and the frequent knowledge paths to describe a group's information needs and referencing behavior. Moreover, a GKF contains several knowledge referencing paths, which indicate the referencing behavior patterns of the group of workers. In addition, frequent referencing behavior patterns of the group of workers, i.e., the paths with scores higher than a user-specified threshold, can be identified from the GKF. Before describing the details of the GKF mining

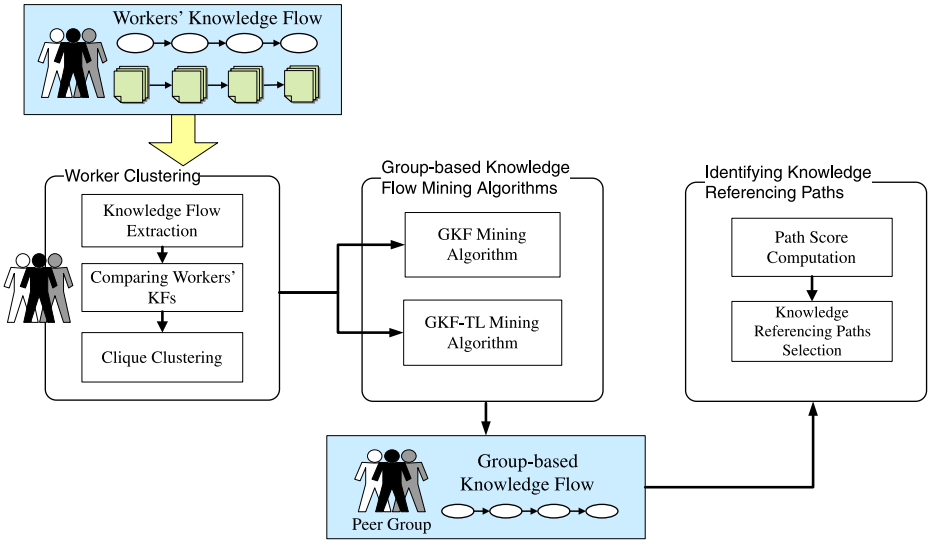

Fig. 2. The procedure of the proposed GKF mining method.

algorithms, we first define several important concepts and features used in the algorithms as follows.

Definition 4. Knowledge graph.

A knowledge graph is defined as $G=(V, E)$, where V is a finite set of vertices, and E is a finite set of directed edges connecting two topics. Each vertex in V denotes a topic in the knowledge domain, and each edge in E denotes the knowledge flow from one topic to the other topic.

Example. Given a directed knowledge graph comprised of two vertices (topics) v_x and v_y and an edge $e_{x,y}$, the edge is used to connect vertices v_x to v_y directly. In addition, v_x is said to be an adjacent predecessor of v_y, while v_y is said to be an adjacent successor of v_x.

Definition 5. Knowledge sub-graph.

Given a knowledge graph $G=(V, E)$, a knowledge sub-graph of G is a graph $G'=(V', E')$, where V' and E' are subsets of V and E respectively, i.e., $V' \subset V$ and $E' \subset E$.

A GKF graph represents the referencing behavior of a group of workers as a directed knowledge graph, which consists of a finite set of vertices and edges, defined as follows.

Definition 6. Group-based knowledge flow (GKF).

As mentioned earlier, a GKF is derived from the KFs of workers who are in the same cluster and therefore have similar information needs. A GKF is defined as $GKF=\{G, W, TKF\}$, where G is a directed knowledge graph; $W=\{w_i|\forall i, i=1\cdots n\}$ is a set of n workers who have similar KFs; and $TKFS=\{TKF_j|\forall j, j=1\cdots n\}$ is a set of topic-level KFs of the workers in W.

The properties of TKF and the directed knowledge graph G are defined as follows.

Definition 7. Flow relation and direct flow relation.

In a flow relation of a topic-level KF (TKF), topic x is followed by topic y, denoted by $x>y$, if topic x was accessed before topic y in the TKF. A topic x is followed directly by another topic y if there does not exist a distinct topic such that x is followed by z and z is followed by y. Thus, the relation between topics x and y is a direct flow relation, defined as $x \rightarrow y$.

Definition 8. Path.

Given a directed graph G, if there is a path from a vertex v_x to another vertex v_y, the path is denoted as $v_x \sim > v_y$.

Definition 9. Topic cycle.

Let a flow relation $x>y$ appear in a TKF and a flow relation $y>x$ also appear in another TKF. The relations are represented by their corresponding paths, $v_x \sim > v_y$ and $v_y \sim > v_x$, on the graph of the GKF. Such relations form a topic cycle between the vertices of v_x (topic x) and v_y (topic y) in the GKF.

Definition 10. Topic loop.

Let x be a duplicate topic in a TKF and let two flow relations $x>y$ and $y>x$ appear in the TKF. These relations are represented by their corresponding paths, $v_x \sim > v_y$ and $v_y \sim > v_x$, on the graph of GKF. Such relations form a topic loop between the vertices of v_x (topic x) and v_y (topic y) in the GKF.

Definition 11. Strongly connected component (SCC).

A strongly connected component is a maximal strongly connected sub-graph in which every vertex is reachable from every other vertex in the sub-graph.

Definition 12. Knowledge referencing path.

Given a directed graph $G=(V, E)$ of a GKF, if there is a path from a start vertex to an end vertex, it is a knowledge referencing path. Such a path is defined as $p=\{s, d, V_p, E_p\}$, where s is a start vertex, d is an end vertex, and V_p is a set of topics on the path p. E_p is a set of edges, where each edge is an ordered pair (v_i, v_j); v_i and $v_j \in V_p$, $v_i \neq v_j$ and v_i is an adjacent predecessor of v_j.

Definition 13. Frequent referencing path.

Given a set of referencing paths derived from the graph of the GKF, a path p is said to be frequent if its path score, which is derived based on the frequency count of edges on the path, is greater than a certain threshold. A frequent referencing path indicates that workers accessed task-related knowledge in a particular topic order frequently.

Problem Statement: Given the TKFs of a group of workers, the GKF mining algorithms finds the GKF from the TKFs. The GKF is represented by a directed graph, which is used to model the referencing behavior of a group of workers.

4. GKF mining algorithm

To derive a GKF model from a set of TKFs, we propose two algorithms: one for cases where there are no duplicate topics in a TKF; and the other for cases where there are duplicate topics. Both algorithms model a group's information needs as a group-based knowledge flow. The referencing path of a GKF details the order in which topics are accessed when workers search for task-related knowledge. In Section 4.1, we present a GKF mining algorithm for cases without duplicate topics. The GKF mining algorithm for dealing with duplicate topics is presented in Section 4.2.

4.1. GKF mining algorithm without considering duplicate topics

We assume that a topic in a TKF appears just once in this algorithm. That is, there is no duplicate topic in each TKF; hence, there will not be a topic loop in the GKF. However, the order of topics in different TKFs may vary, so topic cycles, which form strongly connected components, may appear in the graph G.

In a strongly connected component (SCC), where each vertex is reachable from every other vertex, it is difficult to determine the ordering relation among the vertices. To resolve the problem, the algorithm applies the *Topic_Relation_Identification* procedure to identify the vertex relation in the SCC. The relation, which can be classified as either a parallel relation or a sequential relation to characterize the topic relations in the GKF, represents part of the topic ordering in workers' referencing behavior.

The GKF mining algorithm discovers frequent referencing of topics from the TKFs of a group of workers. To discover frequent referencing behavior patterns, which are modeled as frequent edges or frequent referencing paths on the GKF graph, the algorithm use the edge deletion procedure to remove infrequent edges whose weights are lower than a user specified threshold. A start vertex and an end vertex are added to the discovered graph to indicate the start and end of the referencing behavior paths of the workers. Note that a topic is represented as a vertex on the graph. It would be odd to generate a GKF in which topic references were incomplete; that is, where a topic reference does not originate at the start vertex or reach the end vertex. The algorithm ensures that every topic can be referenced successfully from the start vertex to the end vertex. Thus, an infrequent edge can only be deleted if its removal does not make any vertex unreachable from the start vertex or to the end vertex.

Several knowledge paths may exist on a GKF graph. The paths represent the group's frequent referencing behavior when learning/

referencing knowledge. Thus, the discovered graph can be used to inform a group of workers about topics of interest and the referencing behavior related to those topics.

The steps of the proposed algorithm are shown in Fig. 3. To generate a GKF model for a specific group (task), a set of TKFs is taken as the algorithm's input, and the graph of the *GKF* is the output result. In the GKF graph, a topic domain in a TKF is represented as a knowledge vertex, and each flow that directly orders the knowledge between two topics is represented as an edge. For example, given a TKF <A, B, E, C>, the four topics A, B, E and C are represented as four knowledge vertices, i.e., v_A, v_B, v_E and v_C, respectively; and the direct flows between two knowledge vertices are represented as three directed edges, i.e., $e_{A,B}$, $e_{B,E}$, and $e_{E,C}$, in the graph of *G*. Note that an edge is used to order the flow between two topics directly, e.g., the edge $e_{A,B}$ orders the flow from topic *A* to topic *B*. In contrast, if two topics have no direct flow relation, no edge exists between them. In the same example, there is no flow relation between topic *A* and topic *E* , so an edge $e_{A,E}$ does not exist.

The algorithm for building the GKF model involves several steps. First, a start vertex *s* and an end vertex *d* are added to the directed graph. Second, each topic in a TKF is regarded as a vertex and is added to a vertex set *V* if it does not exist in *V* already. Then, to connect the vertices in *V*, the edges related to the inserted vertex are added to the edge set *E* as follows. Let $x \rightarrow y$ be a direct flow relation from topic *x* to topic *y*, which denotes that topic *x* is followed immediately by topic *y* in a TKF_w. When adding the edge $e_{x,y}$ to *E*, the algorithm has to check two additional conditions for the edge to connect the start/end vertex with other vertices. First, if the vertex *y* is the first topic in a TKF, the edge $e_{s,y}$ from the start vertex *s* to the vertex *y* is added to *E*; then, if the vertex *y* is the last topic in the TKF, the edge $e_{y,d}$ from the vertex *y* to the end vertex *d* is added to *E*. When adding an edge to *E*, the algorithm counts the frequency of the edge. Adding all the vertices and their related edges to *V* and *E* respectively yields the initial graph of the GKF model.

Example. This example illustrates how to build a GKF graph by using the GKF algorithm without considering duplicate topics in a TKF. Five workers who have similar TKFs form a group. Their topic-level KFs are listed in Table 1.

Table 1
Five workers and their TKFs.

Worker	Topic-level KF (TKF)
John	<A, B, C, D, E>
Mary	<A, C, G, F, D, E>
Lisa	<B, A, C, E>
Tom	<A, B, C, D>
Bob	<C, B, G, F, D>

The topic domains in each topic-level KF (TKF) are arranged as a topic sequence according to the times they were referenced. Based on the TKF of each worker, the proposed algorithm derives the group's GKF, which is represented by a directed graph, as shown in Fig. 4. The topic domains, including the start and end vertices are represented by circles; an edge is represented by an arrow, which indicates the direction of knowledge flow from one knowledge vertex to another; and the number on each edge is the edge's frequency count.

In the initial graph, a strongly connected component (SCC) may be evident when some vertices appear in reverse order in any two TKFs. A strongly connected component G_s is a maximal strongly connected sub-graph that contains a path from each vertex to every other vertex in G_s. Because the vertices in a connected component are strongly connected, it is difficult to determine the ordering relationships between them. Even so, such relationships can be used to represent the characteristics of a TKF and they are important for modeling workers' referencing behavior. Thus, we use a procedure called *Topic_Relation_Identification* to determine the relationships among vertices in any strongly connected component.

In an SCC, two kinds of relations can be identified, namely, parallel and sequential relations. Any two vertices in an SCC indicate that two topics, *x* and *y*, may be referenced by different TKFs with the ordering $x>y$ and $y>x$. This ordering is an example of a parallel relation, where either $v_x \sim > v_y$ or $v_y \sim > v_x$ would be appropriate; thus, there is no strict ordering between v_x and v_y. The referencing order of the vertices is not obvious, and the knowledge items represented by the vertices may be referenced simultaneously. As the vertices in an SCC are not in a specific order, conventional workflow mining methods consider the association between the vertices as a parallel relation. However, in contrast to such methods, a sequential relation pattern (SRP) rather than a parallel relation pattern (PRP) may be extracted if most of the referencing behavior in the SCC fits the SRP. That is, the SRP represents the most frequent knowledge referencing pattern in the SCC.

We explain how to recognize the above relations in Section 4.1.1, and how to evaluate, the weight of each edge when measuring the importance of a flow in the GKF in Section 4.1.2. Then, we transform

GKF mining algorithm
Input: A set of *n* workers in *W* and their KFs, $TKFS = \{TKF_w \mid w=1...n\}$;
Output: $GKF = \{G, W, TKFS\}$;
A directed graph $G = \{V, E\}$, where $V = \phi$ and $E = \phi$;
Add a start vertex *s* and an end vertex *d* to *V*;
For each TKF_w in *TKFS* {
  Add each topic v_y to *V* according to the sequence order in TKF_w;
  Add an edge between the start vertex and the first topic in TKF_w in *E*;
  Add an edge between the last topic in TKF_w and the end vertex in *E*;
  For each vertex $v_x \in V$ and $v_x \rightarrow v_y$ in TKF_w {
    Add an edge between vertex v_x and v_y in *E*; }
  Update the frequency of each edge in *E*;
}
Identify the strongly connected components (SCC) from *G*;
For each SCC G_s, where $G_s = (V_s, E_s)$, $V_s \in V$ and $E_s \in E$
  Topic_Relation_Identification(*TKFS*, *G*, G_s);
Calculate the weights of all the edges in *E*;
Transform the graph *G* into a new graph G_N by mapping each SCC in *G* as a vertex v_{Gs}
in G_N and mapping edges connected to G_s in *G* as edges connected to v_{Gs} in G_N, where
$G_N = (V_N, E_N)$;
L = Topological Sorting (V_N, E_N);
P = Edge Deletion (L, G, G_N);

Fig. 3. The algorithm for mining a GKF when TKFs do not contain duplicate topics.

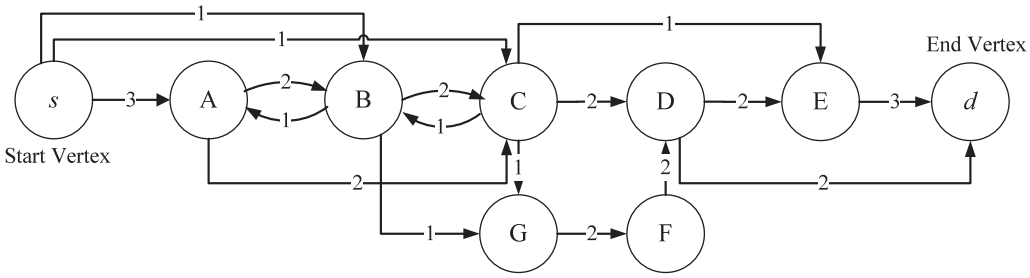

Fig. 4. The initial graph of the GKF model.

the initial graph of the GKF into a new directed acyclic graph G_N in which a strongly connected component G_s is regarded as a vertex.

After graph transformation, the topological sorting and edge deletion procedures are applied on G_N to remove any infrequent edges. An infrequent edge indicates that only a few workers in the group adopt a particular reference behavior pattern. Since such patterns are not representative of the group's general referencing behavior, they can be removed. The topological sorting procedure is used to sort all vertices in V_N in topological order, as discussed in Section 4.1.3. Based on the sorting result, the edge deletion procedure (described in Section 4.1.4) checks all the edges and removes infrequent and unqualified edges from E_N and E. After edge deletion, the graph G represents the group-based knowledge flow.

4.1.1. Topic relation identification

The topic relation identification procedure determines the relations between vertices in a strongly connected component, as shown in Fig. 5. Let the strongly connected component $G_s = (V_s, E_s)$, where V_s is a vertex set and E_s is an edge set. Parallel and sequential relations can be discovered from a strongly connected component $G_s = (V_s, E_s)$ based on the frequency count of knowledge flow sequences (*KFSs*). To determine and rebuild the relationships between vertices in V_s, all

```
1   Topic_Relation_Identification (TKF, G, Gs) {
2     Identify all possible non-duplicate flow sequences of length | Vs | from Gs, where KFS =
3         {KFSx | x= 1..n};
4     //Identify a sequence of vertices in Vs from a TKF and compare it with sequences in KFS
5     For each TKFw {
6       Identify a non-duplicate sequence SQw in TKFw that contains the common vertices in
7           Vs and TKFw, i.e., V(SQw) = Vs ∩ V(TKFw);
8       Compare SQw with each KFSx in KFS. If SQw is a subsequence of KFSx, increase the
9           frequency count of KFSx , i.e., fKFSx;
10    }
11    Sort all KFSx and select top-2 frequent flow sequences KFSa and KFSb;
12    Add a preceding pseudo node vγ and a succeeding pseudo node vρ of Gs to V;
13    //parallel relation (and/or split)
14    If (|fKFSa - fKFSb| ≤ ε) {
15        For each edge ei,j in Es {
16          If (vi → vj exists in a TKFw and vj > vi exists in another TKFy)
17            Remove the edge ei,j from E and Es;
18        }
19        For each vertex vi in Vs {
20          For each adjacent predecessor vk of vi, where vk ∈V and vk ∉Vs {
21            Replace the edges ek,i with the edges ek,γ and eγ,i, and update their frequency
22              counts; }
23          For each adjacent successor vl of vi, where vl ∈V and ∉Vs {
24              Replace the edges ei,l with the edges ei,ρ and eρ,l, and update their frequency
25              counts; }
26        }
27    }
28    else {    //sequential relation
29        If (fKFSa > fKFSb) or (fKFSb > fKFSa)
30            Let KFSy be the most frequent flow sequence;
31        Let vi/ vj be the first/ last vertex in KFSy;
32        Remove all edges in Es from Es and E;
33      For each vg → vh in KFSy {add edges eg,h to Es and E};
34        For each vertex vf in Vs {
35            For each adjacent predecessor vk of vf, where vk ∈ V and vk ∉ Vs {
36                Replace edge ek,f with edges ek,γ and eγ,i, and update their frequency counts; }
37            For each adjacent successor vl of vf, where vl ∈ V and vl ∉ Vs {
38                Replace edge ef,l with edges ej,ρ and eρ,l, and update their frequency counts; }
39        }
40    }
41    Return G;
42  }
```

Fig. 5. The topic relation identification procedure.

possible non-duplicate *KFSs* of length $|V_s|$, which contain all vertices in V_s, are identified from G_s. The derived *KFSs* are then compared with a non-duplicate sequence, i.e., SQ_w, in a TKF_w, which contains a set of vertices that are common to both V_s and the vertex set of $V(TKF_w)$, i.e., $V(SQ_w) = V_s \cap V(TKF_w)$. $V(SQ_w)/V(TKF_w)$ denotes the set of vertices in the sequence SQ_w/TKF_w. When the sequence SQ_w is a subsequence of a *KFS*, the frequency count of the *KFS* is increased. Next, all the *KFSs* are sorted in descending order of their frequencies and the top-2 frequent *KFSs* are selected to elicit the relations of vertices in V_s. The preceding pseudo node v_γ and the succeeding pseudo node v_ρ of G_s are also added to V.

If the difference in the frequency counts of the selected *KFSs* is lower than a user-specified threshold ε, the order of the vertices in V_s is not significant. In this case, the vertex relation is defined as parallel. For example, let us consider a strongly connected component where vertex v_x, vertex v_y and vertex v_z are in V_s; and let the user-specified threshold $\varepsilon = 2$. When the frequency counts of two *KFSs* $<v_x, v_y, v_z>$ and $<v_z, v_y, v_x>$ are 7 and 6 respectively, the relation between vertex v_x, vertex v_y and vertex v_z is parallel because the difference in their frequency counts is lower than the threshold. However, if the difference is greater than a user-specified threshold, the *KFS* with the largest frequency count can be used to represent the relationship of vertices in V_s based on the majority principle. The ordering of these vertices is defined as a sequential relation. Next, we explain how to identify the order of vertices in a strongly connected component, i.e., parallel relations and sequential relations.

4.1.1.1. Identifying parallel relations in a SCC. For parallel relations, the order of the vertices in V_s is not important. The *Topic_Relation_Identification* procedure checks each edge in E_s for each TKF. Let $e_{i,j}$ be an edge in E_s that connects vertex v_i to vertex v_j directly. If this direct flow relation $v_i \rightarrow v_j$ appears in a TKF and a flow relation $v_j > v_i$ exists in another TKF, the edge $e_{i,j}$ is removed from E and E_s, and the relation between vertex v_i and vertex v_j is regarded as parallel. That is, there is no specific ordering between vertex v_i and vertex v_j, and their corresponding topics can be referenced in any order.

After adding a preceding pseudo node v_γ and a succeeding pseudo node v_ρ to G, the edges connected to the vertices in V_s are redirected through the pseudo nodes. To connect a vertex in V to the pseudo nodes, each adjacent predecessor v_k of v_i, where $v_k \notin V_s$ and $v_i \in V_s$, and each adjacent successor v_l of v_i, where $v_l \notin V_s$ and $v_i \in V_s$, are examined. For vertex v_k, if edge $e_{k,i}$, which connects vertex v_k to vertex v_i , exists in E, it is removed. Then, the edges $e_{k,\gamma}$ and $e_{\gamma,i}$ are added to E and their frequency counts are calculated. If the two edges already exist in E, their frequency counts are simply updated. Briefly, the edge $e_{k,i}$ is replaced by edges $e_{k,\gamma}$ and $e_{\gamma,i}$ to make a connection with vertex v_k and vertex v_i through the pseudo node v_γ. Similarly, for a vertex v_l, if edge $e_{i,l}$ exists in E, it is removed. Then, the edges $e_{i,\rho}$ and $e_{\rho,l}$, are added to E and their frequency counts are calculated. If the edges already exist in E, their frequency counts are simply updated.

Example. In Fig. 4, there is a strongly connected component G_s comprised of $V_s = \{A, B, C\}$ and $E_s = \{e_{A,B}, e_{B,A}, e_{B,C}, e_{C,B}, e_{A,C}\}$. Let the threshold ε be 1. The graph of the GKF after topic relation identification is shown in Fig. 6. Based on the *Topic_Relation_Identification* procedure, two pseudo nodes, v_γ and v_ρ, are added to G. Then, the edges in E_s are examined to determine which ones should be removed. Two non-duplicate sequences are discovered in G_s, i.e., <A, B, C>, <C, B> and <B, A, C>; their frequency counts are 2, 1 and 2 respectively. Because the difference in the frequency counts of the top-2 sequences is equal to 1, the relation between vertex v_A, vertex v_C, and vertex v_B is regarded as parallel, and the edges $e_{A,B}$, $e_{B,A}$, $e_{B,C}$ and $e_{C,B}$ are removed from the graph.

Meanwhile, the relation between vertex v_A and v_C is regarded as sequence because A → C exists in one TKF, but there is no C > A in any other TKF. Thus, $e_{A,C}$ is not removed from the graph. The incoming edges of vertex v_A, vertex v_B and vertex v_C are changed to make connections through pseudo node v_γ. Similarly, the outgoing edges of vertex v_A, vertex v_B and vertex v_C are changed to make connections through pseudo node v_ρ. Then, the frequency counts of these edges are updated (Fig. 6).

4.1.1.2. Identifying sequential relations in a SCC. If the difference between the frequency-counts of the selected top-2 *KFSs* is greater than a user-specified threshold, the ordering of the vertices in the KFSs is regarded as a sequential relation. That is, based on the majority principle w.r.t. knowledge referencing behavior discussed earlier, the vertices in V_s follow the ordering of the *KFS* with the highest frequency. Let KFS_y be the knowledge flow sequence with the highest frequency count; and let v_i and v_j be, respectively, the first and last vertices in the sequential order of KFS_y. All the edges in E_s are removed from E_s and E. Then, for each direct flow relation $v_g \rightarrow v_h$ in KFS_y, an edge $e_{g,h}$ is added to E_s and E. Similarly, the edges connected to the vertices in V_s are redirected through the pseudo nodes.

For each adjacent predecessor v_k of v_f , where $v_k \in V$, $v_k \notin V_s$, and $v_f \in V_s$, the edges $e_{k,\gamma}$ and $e_{\gamma,i}$ are added to E, and their frequency counts are calculated. If the edges already exist in E, their frequency counts are simply updated. The edge $e_{k,f}$, which connects vertex v_k to vertex v_f , is removed from E and replaced by the connections from v_k to v_γ and from v_γ to v_i, the first vertex of KFS_x. That is, the edge $e_{k,f}$ is replaced by edges $e_{k,\gamma}$ and $e_{\gamma,i}$, which connect with vertex v_k and vertex v_i respectively through the pseudo node v_γ. Similarly, for each adjacent successor v_l of v_f , where $v_l \in V$ and $v_l \notin V_s$, and $v_f \in V_s$, we use the same method to establish connections from the last vertex in KFS_x to the vertex v_l through the pseudo node v_ρ. The connection from v_f to v_l is replaced by the connections from the last vertex of KFS_x, i.e., v_j, to the pseudo node v_ρ and from v_ρ to v_l.

Example. Table 2 lists the knowledge flows of a group of seven workers. The GKF mining algorithm, described in Section 4.1, is used to generate the graph of the group-based KF and a strongly connected component with vertices v_B, v_C, and v_D is identified from the GKF graph. Then, the *Topic_Relation_Identification* procedure is applied to determine the relation between those vertices. As shown in Fig. 7, the relation is sequential with the ordering v_B, v_C, and v_D. In addition, the edges connected to any vertex in V_s are changed. For example, the

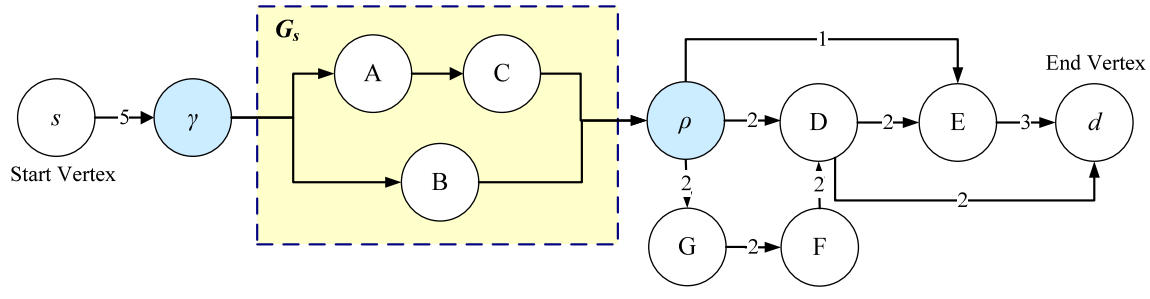

Fig. 6. A parallel relation in a GKF graph.

Table 2
The TKFs of seven knowledge workers.

Worker	Topic-level KF (TKF)
W1	<A, F, B, C, D, H>
W2	<A, G, B, C, D, I>
W3	<F, B, C, D, H>
W4	<A, F, C, D, B, K, H>
W5	<F, C, D, B, K, H>
W6	<A, G, B, C, K, H>
W7	< F, B, C, D>

edge $e_{B,K}$ is changed to edge $e_{D,\rho}$ and edge $e_{\rho,K}$ such that there is a path from vertex v_B to vertex v_K via the pseudo node v_ρ.

4.1.2. Measuring the importance of an edge

Our objective is to derive the referencing behavior of a group of workers by constructing a frequent knowledge path in a GKF graph. However, some infrequent edges in the graph may not be suitable for building the path. To measure the importance of each edge in a graph, the frequency count of each edge is normalized by the maximum edge frequency in E. The weighting function measures the importance of an edge in a GKF model, as defined in Eq. (5).

$$we_{x,y} = \frac{f_{x,y}}{\max\{f_{i,j} | \forall i,j, e_{i,j} \in E\}}, \quad (5)$$

where $we_{x,y}$, which ranges from 0 to 1, is the weight of the edge $e_{x,y}$ that represents a direct flow from vertex v_x to vertex v_y; $f_{x,y}$ is the frequency of the edge $e_{x,y}$; E is the edge set of the graph; and the denominator is a maximum function that derives the frequency count of the most frequent edge in the graph. The more frequently an edge occurs, the more important it is deemed to be. The most frequent edge represents the frequent referencing behavior of most members of the group. Thus, it is suitable for describing the group's referencing behavior.

Example. The weight of each edge in Fig. 6 is calculated by using the edge weighting method. The edge is then labeled with the weight to indicate its importance in the graph, as shown in Fig. 8.

4.1.3. Graph transformation

To simplify a strongly connected component in a graph, the proposed algorithm transforms the original GKF graph into a new graph G_N. After the transformation, the graph G_s is regarded as a vertex v_{Gs} in G_N. We create two pseudo nodes, v_γ and v_ρ , to represent, respectively, the split operator and the join operator of G_s. In addition, the incoming/outgoing edges of G_s, which connect to the pseudo nodes v_γ (the split operator)/v_ρ (the join operator), are merged to form a new edge whose weight is also updated. The weight of the incoming edge of v_{Gs}, which combines the incoming edges of G_s, is derived by combining the edge weights of the incoming edges of the node v_γ. Similarly, the weight of the outgoing edge of v_{Gs} is derived by combining the edge weights of the outgoing edges of the node v_ρ.

Example. We transform the graph G_s in Fig. 8 into a new graph for further analysis, as shown in Fig. 9. To simplify the strongly connected component, all the vertices in G_s are wrapped as a vertex v_{Gs} in the new graph. The incoming edges and outgoing edges of any vertex in G_s and the weights of those edges are adjusted. In Fig. 8, edge $e_{\gamma,A}$ and edge $e_{\gamma,B}$ are merged to form a new edge $e_{\gamma,Gs}$ in Fig. 9 and their edge frequencies are combined as 1. In the same way, edge $e_{C,\rho}$ and edge $e_{B,\rho}$ are combined to form an edge $e_{Gs,\rho}$.

4.1.4. Topological sorting

The frequent referencing behavior of a group of workers is derived by mining the group's knowledge flow from a GKF graph. The workers may reference topics in a different order when performing tasks, but some referencing behavior is more frequent because the majority of workers in the group reference topics in the same order. In the GKF graph, a frequent knowledge path from the start vertex to the end vertex represents the workers' frequent referencing behavior. For any vertex v_i on the path, vertex v_i is reachable from the start vertex and the end vertex is reachable from vertex v_i. Note that a path with infrequent edges denotes an infrequent referencing behavior pattern.

To derive a group's frequent referencing behavior, a topological sorting procedure is used to sort all vertices in the graph, after which infrequent edges whose weights are lower than a specified threshold are deleted. In graph theory, topological sorting [10,19] is a very efficient way to arrange the vertices of a directed acyclic graph in topological order in linear time. The key property of the topological order is that, for any two vertices x and y, if x is a predecessor of y in the graph then x precedes y in the topological order.

In this work, we use topological sorting to arrange all vertices in G_N, which is a directed acyclic graph before the edge deletion procedure is applied. Then, the edge-deletion procedure examines the vertices in topological order to identify the infrequent incoming edges of each vertex that should be removed. However, before removing an infrequent edge, the procedure needs to ensure that each vertex in the GKF satisfies two criteria. First, any vertex v_i on a knowledge path must be reachable from the start vertex and the end vertex must be reachable from vertex v_i. Second, removing the edges of a vertex v_i does not affect the path from the start vertex to the preceding vertices of v_i in the topological order. In other words, topological ordering guarantees that 1) a predecessor will be processed before a successor; and 2) the predecessor's reachability (i.e., from the start vertex to v_i) will not be affected by its successors. Thus, when an infrequent edge of any vertex v_i in G is removed, there is no need to verify the reachability of the predecessors of vertex v_i from the start vertex. On the other hand, the path from the predecessors of vertex v_i to the end vertex will be affected by removing an infrequent edge of v_i; therefore, the predecessors should be examined again to ensure that they can still reach the end vertex.

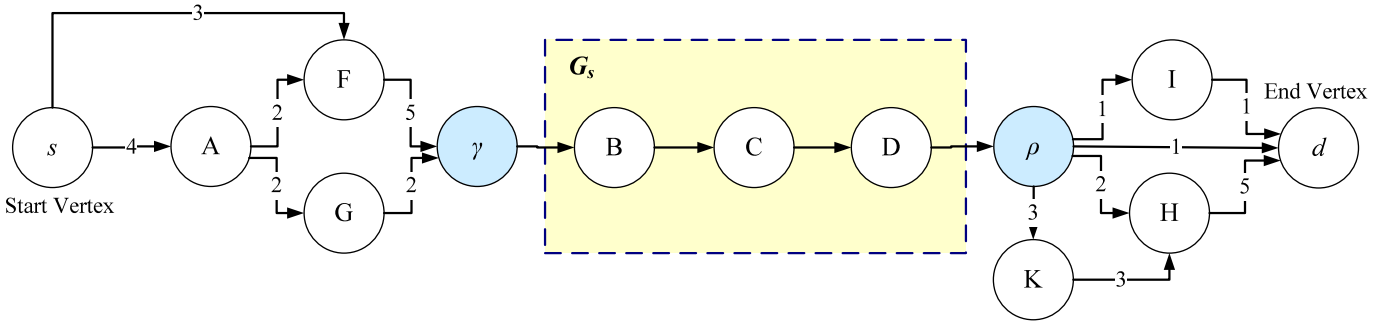

Fig. 7. A sequential relation in a GKF graph.

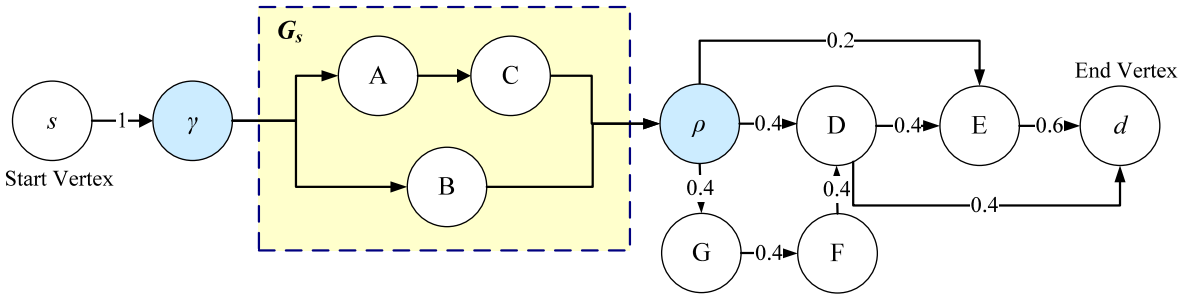

Fig. 8. The edge weights in a GKF graph.

Example. In Fig. 9, all the vertices are sorted in topological order, and the resulting list is <*s*, γ, G_s, ρ, G, F, D, E, *d*>. According to the list, v_s is the first vertex to be checked, v_{Gs} is the second vertex and so on. The algorithm examines all the vertices in topological order and removes infrequent edges from the graph G_N via the edge deletion procedure.

4.1.5. Using the edge deletion procedure to remove infrequent edges

Based on the results of topological sorting of V_N, the edge deletion procedure examines the vertices and determines which incoming edges should be removed from them. It then removes infrequent edges whose weight is lower than a user-specified threshold, as shown in Fig. 10. The inputs of this procedure are the sorted list L derived by topological sorting and the edge set E_N of the GKF graph. The algorithm checks the incoming edges of each vertex in ascending order of their weights, and those whose weights are lower than a user-specified threshold η are candidates for removal. If an edge is removed, it means that the knowledge referencing behavior between two vertices (topics) is infrequent among the group of workers.

However, an infrequent edge should only be deleted from the graph if removing it would not make any vertex unreachable. Let Q be the set of vertices that have been checked in topological order to remove their infrequent incoming edges. For a vertex v_y, if one of its incoming edges is removed and there is no other path from the start vertex to v_y, the removed edge should be returned to the edge sets E and E_N. In addition, the vertices checked before v_y should be reexamined to ensure that there is a path from a checked vertex v_i in Q to the end vertex. If removing an edge violates the above condition, the edge should be returned to the edge sets E and E_N.

Because of the characteristics of topological sorting, the edge deletion procedure ensures that 1) any vertex in the graph G_N can be reached from the start vertex; and 2) removing an edge of a vertex does not affect any path from the start vertex to the predecessors of the vertex. In other words, there exists at least one path from each vertex to the end vertex. Moreover, we can obtain several frequent knowledge paths from the GKF graph to help workers learn the group's knowledge. The following example explains how to remove an edge from the GKF graph.

Example. In Fig. 9, let vertex v_E be the examined vertex and let the user-specified threshold be 0.3. The vertex v_E has two incoming edges: $e_{\rho,E}$ with weight 0.2 and $e_{D,E}$ with weight 0.4. The edge $e_{\rho,E}$ qualifies for removal, because its weight is lower than 0.3 and removing it would not make any vertex unreachable. Fig. 11 shows the resulting graph, which represents the GKF of the group. The graph is used to visualize the knowledge flows among the frequent topics and model the referencing behavior of the group.

4.1.6. Properties of the GKF

The generated knowledge graph has several properties. We define and prove the associated lemmas below.

Lemma 1. *Let v_s be the start vertex in a graph, G_N, of a group-based knowledge flow. For any vertex v_h in G_N, there exists a path $P_{s,h}$ from vertex v_s to v_h.*

Proof. In the edge deletion procedure, removal of an incoming edge from a vertex v_h depends on the weight of the edge. All vertices in G_N are visited in topological order and their incoming edges are examined. For any vertex v_h, an incoming edge should be removed if its weight is lower than a user-specified threshold. However, if removing an edge from v_h also removes the path $P_{s,h}$ from G_N, that edge should be returned to the vertex.

When deleting an incoming edge of a vertex, the edge deletion procedure ensures that 1) there is a path $P_{s,h}$ from the start vertex v_s to vertex v_h; and 2) removing an incoming edge from a successor of v_h does not affect the path $P_{s,h}$. The proof is as follows. Let a vertex v_k be a succeeding vertex of v_h in the topological order. Based on the topological order, the edge deletion procedure processes the vertex v_h before vertex v_k and there exists a path $P_{s,h}$. Assume that a path $P_{s,h}$ does not exist from v_s to v_h, because an incoming edge of v_k has been deleted. Thus, a path must have existed from vertex v_s through v_k to v_h before the edge was deleted. Consequently, v_k must be a predecessor of v_h. However, this statement contradicts the algorithm's processing of vertices in topological order. That is, v_k is a succeeding vertex of v_h and the path $P_{s,h}$ exists in G_N. Thus, removing

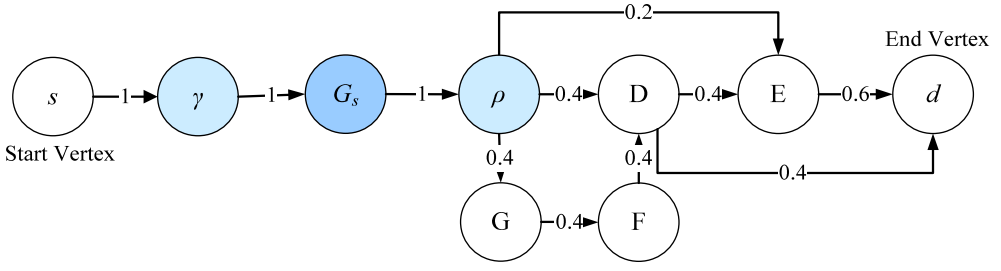

Fig. 9. The result of graph transformation.

```
1   Edge Deletion (L, G, G_N) {
2     Q = φ ; // the checked set of vertices
3     For each vertex v_y in G_N , according to the vertex's order in the sorted list L {
4           For each incoming edge e_x,y of v_y , according to its weight in ascending order {
5                 If (the weight of edge e_x,y < threshold η) {
6                     Remove the edge e_x,y from E and E_N;
7                   If (no path p_s,y exists from the start vertex s to vertex v_y in G_N) or (there
8                       exists a vertex v_j, v_j ∈ Q and no path p_j,d exists from vertex v_j to the
9                       end vertex d)
10                        Add the edge e_x,y to E and E_N;
11                }
12          }
13          Add vertex v_y to Q;
14    }
15  }
```

Fig. 10. The edge deletion procedure.

an incoming edge from a succeeding vertex of v_h does not affect the path $P_{s,h}$. According to the algorithm and the above explanation, for any vertex v_h in G_N, there exists a path $P_{s,h}$ from vertex v_s to v_h.

Lemma 2. *Let v_d be an end vertex in the graph of the group-based knowledge flow G_N. For any vertex v_h in G_N, there exists a path $P_{h,d}$ from vertex v_h to v_d.*

Proof. Let vertex v_k be the succeeding vertex of the vertex v_h. Removing an incoming edge of vertex v_k will affect the reachability of the end vertex v_d from vertex v_h. When the edge deletion procedure removes an incoming edge of vertex v_k, it has to check whether the path $P_{h,d}$ from vertex v_h to the end vertex v_d exists. If it does not exist, the incoming edge should not be removed. Therefore, the procedure ensures that a path $P_{h,d}$ exists from vertex v_h to the end vertex v_d.

Lemma 3. *Let $G_N = \{V_N, E_N\}$ be the directed graph of a group-based knowledge flow. All vertices in V_N can be visited by traversing vertices from the start vertex v_s to the end vertex v_d. Then, for any vertex v_h in V, there exists a path from v_s to v_d through v_h.*

Proof. According to Lemmas 2 and 3, for any vertex v_h in V_N, there exists a path $P_{s,h}$ from the start vertex v_s to v_h and a path $P_{v,d}$ from v_h to end vertex v_d. Therefore, there exists a path from v_s to v_d through v_h.

Lemma 4. *For any infrequent edge $e_{h,k}$ on an infrequent path of G_N, either the path from the start vertex v_s to vertex v_k or the path from the vertex v_h to the end vertex v_d must pass through the edge $e_{h,k}$.*

Proof. Let vertex v_h be a predecessor of vertex v_k in the topological order, and let $e_{h,k}$ be an infrequent edge from vertex v_h to vertex v_k in G_N. Assume that there exist two paths, one from start vertex v_s to vertex v_k and the other from vertex v_h to the end vertex v_d, neither of which passes through the edge $e_{h,k}$. Our algorithm removes any infrequent edge if doing so will not make any vertex unreachable. Thus, the algorithm will remove the edge $e_{h,k}$. However, this contradicts the statement that $e_{h,k}$ exists in G_N. Consequently, for any infrequent edge $e_{h,k}$ of an infrequent path of G_N, either the path from the start vertex v_s to vertex v_k or the path from the vertex v_h to the end vertex v_d must pass through the edge $e_{h,k}$.

The vertex V_{GS} in graph G_N represents a corresponding strongly connected component G_S in G. All vertices in G_S with parallel relations or sequential relations are reachable. Lemmas 2–5 also hold for G.

4.2. The GKF mining algorithm for dealing with topic loops

The GKF mining algorithm for dealing with topic loops (GKF-TL) is based on the GKF algorithm introduced in Section 4.1, which assumes there are no topic loops in workers' KFs when it generates the graph of the group-based KF. A topic loop means that a specific topic appears repeatedly in a TKF because it is referenced by a worker several times. This may happen because the worker needs the knowledge at different times during a task's execution. For example, given a worker's topic-level KF <A, B, A, C, D>, if topic A is referenced twice, it is appears as a topic loop in the corresponding graph of the TKF. Because the loop problem in a workflow mining domain is difficult to resolve, no matter what the application domain, many researchers ignore the problem [12,34]. Agrawal et al. [4] proposed an algorithm for workflow systems that builds a general directed graph with cycles for mining process models from workflow logs. The algorithm gives activities different labels to differentiate them in a workflow instance. The problem of dealing with topic loops in TKFs is analogous to that of workflow systems. Thus, we adopt the above approach to solve the loop problem. Specifically, we propose an algorithm that considers duplicate topics (topic loops) in each TKF to build a directed graph for modeling the referencing behavior of a group of workers.

Different from our knowledge flow mining approach, the workflow mining approach [4] does not consider strongly connected components (SCC) for differentiating the sequential and parallel relations. That is, it did not consider the issue of handling loops involving strongly connected components. We resolve the loop problems involving the vertices in SCC.

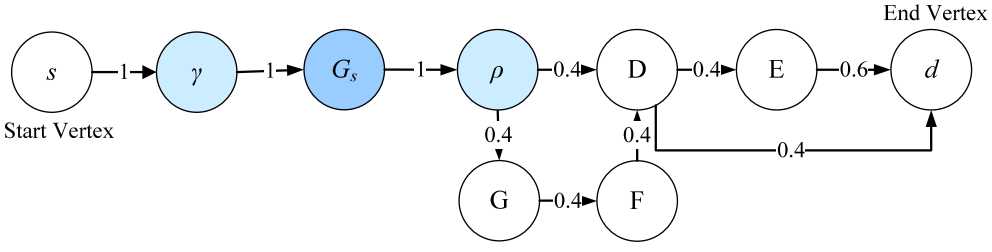

Fig. 11. The final graph G_N of the GKF model.

The GKF-TL algorithm differs from the GKF algorithm. First, it identifies duplicated topics in a TKF and gives them different labels in order to solve the loop problem. For example, given a KF <B, A, B, C, B>, because topic B appears three times, it is transformed into three instances, i.e., B1, B2 and B3, such that the original KF becomes <B1, A, B2, C, B3>.

After infrequent edges have been removed from the graph G, it is transformed into a new graph G_T as follows. The vertices with different instances of the same topic form an equivalent set and can be merged to make one vertex. For a topic *TP* in a TKF, each vertex in the equivalent set of *TP* is an instance of the topic. Then, a directed edge is added to the new graph G_T if there is an edge between two vertices of different equivalent sets in graph G. Initially, the merging process is applied to vertices of each equivalent set in G when a strongly connected component is not involved. To merge vertices involving a strongly connected component G_s, the steps are as follows.

Let vertices v_i/v_j be instances in the equivalent sets Q_a/Q_b, and let v_k be an another instance in Q_a as well as a vertex in a strongly connected component, i.e., $v_k \in G_s$, where v_γ and v_ρ are two pseudo nodes of G_s. Note that because v_k and v_i are instances of the same topic, they are in the same equivalent set and are thus merged to form one vertex. In addition, v_i is in G_s, since v_k is in G_s. Generally, the vertices of an equivalent set Q_a in G are combined as a vertex v_a in the new graph G_T, while the vertices of an equivalent set Q_b are merged to form one vertex v_b. For a strongly connected component G_s with pseudo nodes v_γ and v_ρ, if a directed edge $e_{i,j}$ between v_i and v_j exists in G, a directed edge $e_{\rho,b}$ is added to the new graph G_T. Similarly, if a directed edge $e_{j,i}$ exists in G, a directed edge $e_{b,\gamma}$ is added to G_T.

Next, we consider how to combine vertices involving two strongly connected components. Let v_k/v_l be vertices in strongly connected components G_a/G_b; $v_{\gamma a}$ and $v_{\rho a}$ be pseudo vertices that connect with graph G_a; $v_{\gamma b}$ and $v_{\rho b}$ be pseudo vertices that connect with G_b; and Q_a/Q_b be the corresponding equivalent sets of vertices in G_a/G_b. In addition, let vertex v_i and v_k (resp. v_j and v_l) be instances of the equivalent sets Q_a (resp. Q_b). Vertices in Q_a/Q_b are merged as vertex v_a/v_b. Because v_k/v_l is in G_a/G_b, v_i/v_j also belongs to G_a/G_b; however, some edges need to be adjusted. If there is a directed edge $e_{i,j}$ from v_i to v_j in graph G, an edge $e_{\rho a,\gamma b}$ with the same direction as edge $e_{i,j}$ is added to the new graph G_T. Similarly, if a directed edge $e_{j,i}$ exists in graph G, a directed edge $e_{\rho b,\gamma a}$ is added to G_T. These new added edges are used to merge two equivalent sets in different strongly connected components and make a connection between them. Note that the weights of the edges are updated during the merging process.

Note that we assume the instances of a topic exist in at most one strongly connected component after the vertices of each equivalent set have been merged to form one vertex. We defer consideration of the case where the same topic belongs to more than one strongly connected component to a future work. Next, we provide an example of implementing the GKF-TL algorithm.

4.2.1. Example of applying GKF-TL mining algorithm

The following example considers a group of four workers with similar KFs. Their topic-level KFs (TKFs) are listed in Table 3. Each element in a TKF is used to represent a topic domain. Thus, the elements in a TKF are arranged as a topic sequence based on the times they were referenced. As a topic may appear more than once in a specific KF, because the worker needs the knowledge at different times, we apply the GKF-TL mining algorithm to deal with topic loops.

Table 3
The TKFs of four workers.

Worker	Topic-level KF (TKF)	TKF'
John	<A, B, A, C, D, F>	<A1, B1, A2, C, D, F>
Mary	<B, A, B, C, D>	<B1, A1, B2, C, D>
Lisa	<B, A, D, F>	<B1, A1, D, F>
Tom	<A, B, A, E, G, D>	<A1, B1, A2, E, G, D>

A topic that appears more than once in a TKF is labeled as a different instance of the topic, and a TKF with duplicate topics is transformed into a TKF'. Then, the algorithm uses TKF' to build the initial graph of the GKF model. In this example, we set the user-specified thresholds for topic relation identification and edge deletion as $\varepsilon = 1$ and $\theta = 0.3$ respectively. The initial graph derived before graph transformation is shown in Fig. 12. A strongly connected component is discovered in the initial graph. To resolve the vertex relation problem in the strongly connected component, the algorithm applies the topic relation identification procedure detailed in Fig. 5. The vertex relation in the strongly connected component is shown in G_s in Fig. 12. The number on each edge represents the edge's weight. Recall that the weight is derived by Eq. (5) to indicate the importance of the edge.

Fig. 13 shows the result of removing the infrequent edges from the graph in Fig. 12. The sub-graph G_s in the initial graph is transformed into a vertex v_{Gs}; and the edge that connects a vertex in G_s with another vertex, i.e., $e_{\rho,D}$, is removed because its weight is less than 0.3.

Finally, the algorithm merges vertices that are different instances of the same topic into one vertex. For example, in Fig. 12, vertices v_{B1} and v_{B2} are different instances of the same topic, so they are merged to form the vertex v_B. Moreover, the edge $e_{\rho,B2}$ is replaced by an edge connecting v_ρ to v_γ; and the edge $e_{B2,C}$ is changed to edge $e_{\rho,c}$. The vertices v_{A1} and v_{A2} are two instances of topic A; hence they are merged to form vertex v_A, and their edges are changed accordingly. Fig. 14 shows the final GKF graph, which considers the duplicate topics in each worker's TKF. To illustrate all knowledge paths in the graph, the vertex v_{Gs} is converted into the original graph G_s.

4.3. Identifying knowledge referencing paths in a GKF graph

We have developed a method for identifying frequent knowledge paths from the GKF graph to describe the information needs of a group of workers, i.e. their knowledge referencing behavior. A knowledge path, which represents the knowledge referencing behavior of a group of workers, consists of several vertices and edges that can be traversed from the start vertex to the end vertex. To identify a frequent knowledge path, a path score derived from the weights of the edges on a path is used to evaluate each path. Paths with scores higher than a user-specified threshold are regarded as frequent knowledge paths in the GKF and are selected for the group. Specifically, such knowledge paths (patterns) are used to represent the frequent knowledge referencing behavior and important knowledge flows. The paths can also be provided to help group member access and learn group-related knowledge.

We are interested in finding frequent knowledge paths because they represent the frequent referencing behavior patterns of a group of workers. The discovered paths will be important references for workers, especially for novices in the group. A path's score indicates its importance and is based on the weights of the edges on the path, as defined in Eq. (6).

$$ps_i = Min\{we_{x,y} \mid \forall e_{x,y} \in path_i\}, \tag{6}$$

where ps_i is the path score of the path i; and $we_{x,y}$ is the weight of edge $e_{x,y}$, which belongs to the path i and represents a direct flow relation between vertex x and vertex y. Based on the weights of all the edges on a specific path, a path score is derived from the minimal weight among the edges to indicate the path's level of importance. Note that the edge weight derived by Eq. (5) denotes the importance of the direct flow in a GKF. A large edge weight means that the referencing flow between topics is highly significant for the group of workers.

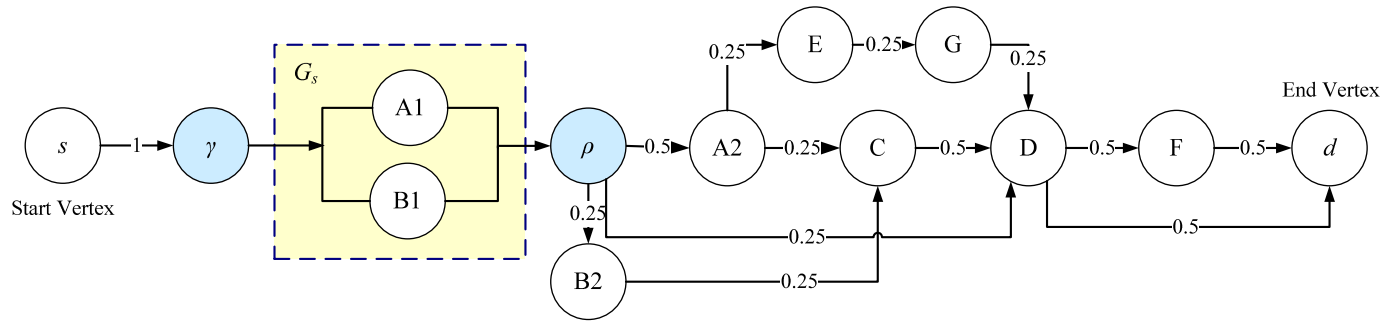

Fig. 12. The initial graph of the GKF model with topic loops.

5. The prototype system for mining group-based knowledge flows

In this section, we develop a prototype system to demonstrate the proposed methods for mining group-based knowledge flows (GKFs), which are generally difficult to formalize. To address the problem, our system provides a mining function and modules to identify GKFs. The referencing paths with scores higher than a user-specified threshold are identified to represent the frequent knowledge referencing patterns of the group.

We use a dataset from a research laboratory in a research institute. It contains information about 14 knowledge workers, 424 research documents, and a usage log that records the time documents were accessed and the workers' document needs. Each worker may perform a number of tasks, e.g., conducting a research project and writing research papers.

5.1. System implementation

To implement our prototype system for group-based KF mining, we use Microsoft Visual Studio 2005 (with C#) to develop the system and Microsoft SQL Server 2005 as the database system to store the dataset. Because the dataset contains workers' logs, it should be preprocessed to generate each worker's codified-level KF and topic-level KF. To obtain the KF, documents in the dataset are grouped into eight clusters by using the agglomerative hierarchical clustering method described in Section 2.3. Based on the clustering results, a topic-level KF is generated by mapping the codified knowledge into its corresponding clusters for each knowledge worker, as described in Section 3.1. Then, the two types of KF, the topic-level KF and the codified-level KF, are derived to describe the information needs of a worker. We use such KFs to build a prototype system to demonstrate the method for mining the knowledge flows of a group of workers.

Our system has two major functions: worker clustering and group-based knowledge flow mining. The former identifies a group of workers based on the similarities of their knowledge flows, as described in Section 3.2. The latter identifies a group's knowledge flow and uses a directed graph to present the mining results, as described in Section 4. An interface that can visualize the KF is necessary. Note that our system can be applied in any knowledge intensive organization to help workers obtain and learn knowledge. Next, we describe the system in detail.

The knowledge flow mining system is comprised of three modules: the main module, the worker clustering module and the GKF model. Each module has functions to help the user (a manager/worker) build a knowledge flow easily. The system provides essential functions for building the GKF model, e.g., the system settings, the KF alignment similarity and clustering functions. The system setting is used to initialize the system environment, e.g., database selection. The KF similarity function calculates the similarity between two workers' knowledge preferences based on their knowledge flows and creates a similarity matrix of the workers. Then, the worker clustering method uses the similarity matrix to cluster workers who have similar KFs. The system also provides an interface to show the topic-level KFs of all workers and the results of worker clustering. To simplify the presentation of the KFs, we use a number to represent a topic domain that consists of topic-related terms.

Next, using the proposed algorithm, the system builds a group-based knowledge flow (GKF) for a group of workers, as shown in Fig. 15. All the workers in a cluster have similar KFs, which are used to generate a GKF graph to characterize the referencing behavior of the group. In the graph, each circle is a topic domain represented by a number, while each directed edge indicates the flow of knowledge between two topics. The topic domain contains a topic profile, which consists of several representative terms and their term weights. Fig. 15 shows the profile of topic domain 53 in a small window. The listed terms represent the knowledge of the topic.

In addition, the number on an arrow indicates the importance of a flow relation in this group's topics. From the GKF graph, we observe that 6 topics, i.e., 4, 17, 19, 21, 27, and 29, in a strongly connected component can be referenced in parallel. That is, there is no specific order among the topics accessed by this group of workers. Moreover,

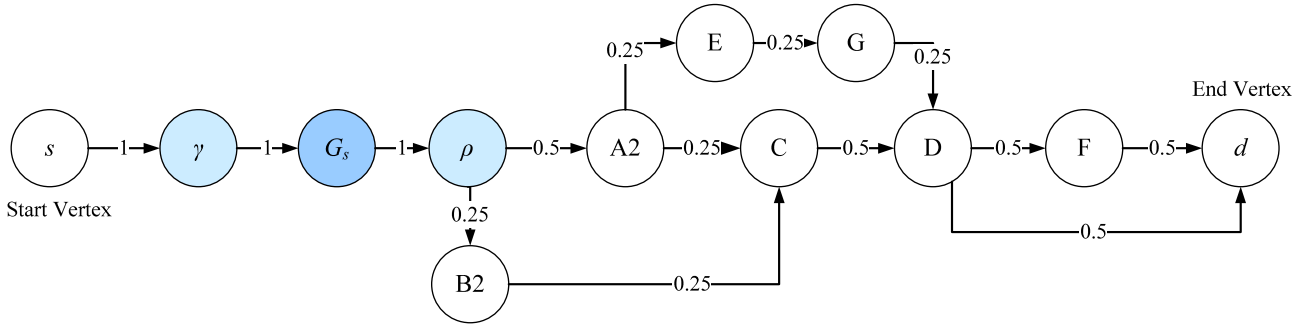

Fig. 13. The graph of the GKF model with topic loops.

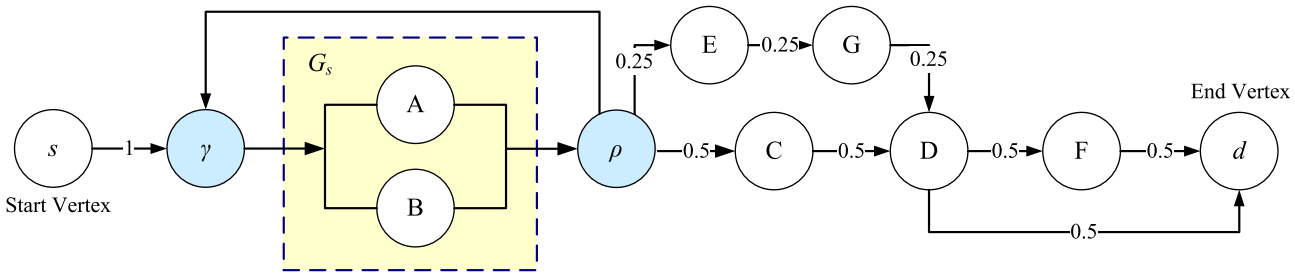

Fig. 14. The final GKF graph, which considers the duplicate topics in each worker's TKF.

the following topics of these parallel topics are topic 53 and topic 36. Such two topics represent two different knowledge domains. Thus, the task-related knowledge may flow through 2 paths from the start vertex to the end vertex. In Fig. 15, the listed paths, which consist of several relevant topics and directed edges, are the knowledge referencing paths of this group. The paths with scores larger than a user-specified threshold are frequent referencing behavior patterns. The paths can be regarded as knowledge references for group members to share needed task knowledge.

5.2. Discussion

GKF mining by task-based groups has several advantages in a knowledge intensive organization. The group-based knowledge flow (GKF) is derived from the knowledge flows of group members and represents the core knowledge required to perform their tasks. Identifying the frequent topics of interest and major referencing behavior patterns of group members can facilitate knowledge reuse, cooperation and sharing among workers who perform similar or relevant tasks and have similar referencing behavior patterns.

The GKF, which represents frequently accessed task-related knowledge topics and the order in which they were referenced, can provide important knowledge resources to fulfill workers' task needs when they reference task-relevant knowledge. Based on the GKF, the frequently accessed task-related knowledge could be reused and shared with the group to support workers in the performance of their tasks. The frequent knowledge paths in the GKF help a worker access/learn task-related knowledge, overcome obstacles encountered when performing a task, and enhance his/her learning efficiency and work productivity. From the group-based knowledge flow, workers can discover the knowledge frequently accessed by other group members. Each worker can also share his/her knowledge with others to facilitate knowledge reuse, cooperation and sharing. In addition, the GKF could complement each worker's knowledge flow.

From the perspective of a task's execution, a group-based knowledge flow will be an important knowledge asset when performing a task similar or relevant to the performance of those tasks from which the group-based knowledge flow was derived. For a novice who needs to execute a task similar to the tasks executed by group members, the GKF can provide a reference for accessing and

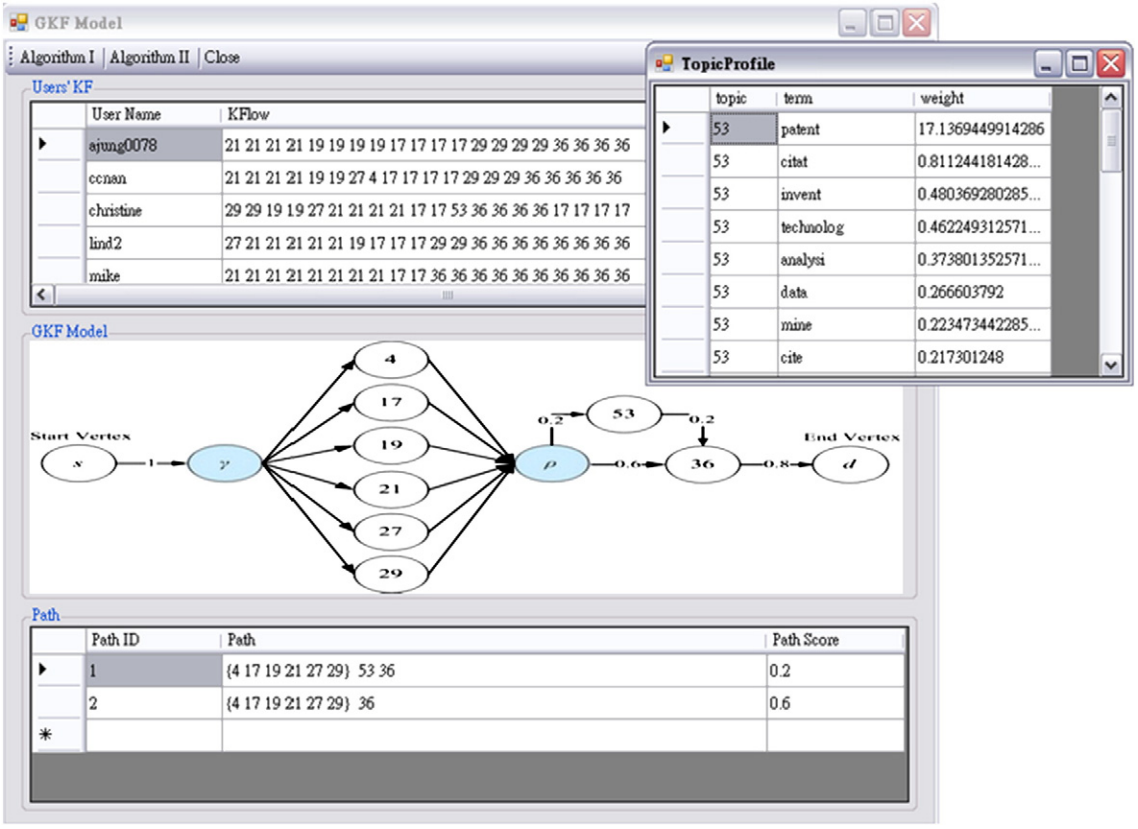

Fig. 15. The GKF graph and knowledge referencing paths for a specific group.

learning the required knowledge. For example, a group-based knowledge flow derived from the knowledge flows of several researchers working on Social Network Analysis (SNA) related research tasks would be helpful to a new researcher who has just started working on an SNA-related research task.

In addition, some tasks, such as business processes or workflows, may be executed repeatedly. The frequently referenced knowledge used to perform the tasks can be identified and shared with group members to enhance their performance through knowledge reuse, cooperation and sharing based on the GKF. In summary, the GKF can be used to support the performance of tasks, enhance organizational learning, and facilitate knowledge sharing and reuse in knowledge-intensive environments.

6. Conclusion and future work

In this paper, we have proposed algorithms for mining group-based knowledge flows (GKFs). Workers performing similar (relevant) tasks or cooperative tasks generally have similar task-related information needs, and can form a group to facilitate knowledge reuse, cooperation and sharing. To discover the GKF of a group of workers, we design algorithms that can analyze the workers' information needs expressed in their KFs to generate a GKF model. The model represents the information needs, the direction of knowledge flows, and possible paths for referencing task-relevant knowledge for a group of workers with similar task needs when they perform similar or cooperative tasks. Based on the model, we can identify representative paths as common behavior patterns for the group. Thus, the patterns can be regarded as accessing and learning references of task-related knowledge to help group members accomplish their tasks. The GKF model can be used to identify task-related knowledge topics and their flows as important knowledge resources to fulfill workers' task needs and promote knowledge sharing among group members. Finally, we implement a prototype system to demonstrate the efficacy of the proposed algorithms. Our system not only derives the KF for a group of workers, but also visualizes the mining results for further analysis.

In this work, we focus on proposing a theoretical model and algorithms for deriving GKFs. A prototype system is implemented to demonstrate the efficacy of the proposed approach. In our future work, we will conduct further evaluations using empirical data collected from organizations. We will also develop a recommendation method based on the GKF, so that workers can reuse, cooperate and share their knowledge with other group members to accomplish their tasks. Moreover, different working groups in an organization may provide knowledge support for one another. To facilitate knowledge sharing in a group or among groups, we will investigate recommendation methods that provide task knowledge to workers and groups proactively. The effectiveness of a recommendation method depends to a large extent on how much workers trust one another. This factor is important because the level of trust may determine whether or not a worker is willing to share knowledge with others. Through group recommendation methods, task-related knowledge can be shared effectively to enhance the efficiency of all knowledge workers.

Acknowledgement

This research was supported by the National Science Council of the Taiwan under grant NSC 96-2416-H-009-007-MY3.

References

[1] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kuhn, M. Sintek, Context-aware, proactive delivery of task-specific information: the KnowMore Project, Information Systems Frontiers 2 (2000) 253–276.
[2] A. Abecker, A. Bernardi, H. Maus, M. Sintek, C. Wenzel, Information supply for business processes: coupling workflow with document analysis and information retrieval, Knowledge-Based Systems 13 (2000) 271–284.
[3] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, Proceedings of the international conference on Management of data (ACM SIGMOD), 1998, pp. 94–105.
[4] R. Agrawal, D. Gunopulos, F. Leymann, G. Boblingen, Mining process models from workflow logs, 6th International Conference on Extending Database Technology (EDBT'98), Valencia, Spain, 1998, pp. 469–483.
[5] A. Anjewierden, R. de Hoog, R. Brussee, L. Efimova, Detecting knowledge flows in weblogs, 13th International Conference on Conceptual Structures (ICCS 2005), 2005, pp. 1–12.
[6] C. Augusto, F. Maria Grazia, P. Silvano, Knowledge-based document retrieval in office environments: the Kabiria system, ACM Transactions on Information Systems 13 (1995) 237–268.
[7] R. Baeza-Yates, B. Ribeiro-Neto, Modern information retrieval, Addison-Wesley, Boston, 1999.
[8] J. Cardoso, M. Lenic, Web Process and Workflow Path Mining Using the Multimethod Approach, International Journal of Business Intelligence and Data Mining 1 (2006) 304–328.
[9] K. Charter, J. Schaeffer, D. Szafron, Sequence alignment using FastLSA, International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS,2000), 2000, pp. 239–245.
[10] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to algorithms, MIT Press, 2001.
[11] A.K.A. de Medeiros, B.F. van Dongen, W.M.P. van der Aalst, A. Weijters, Process mining: extending the a-algorithm to mine short loops, BETA Working Paper Series, WP, 113, 2004.
[12] G. Greco, A. Guzzo, G. Manco, D. Sacca, Mining and reasoning on workflows, IEEE Transactions on Knowledge and Data Engineering 17 (2005) 519–534.
[13] J. Guo, Y. Wang, Context modeling for knowledge flow, IEEE International Conference on Information Reuse and Integration (IRI), 2008, pp. 330–335.
[14] W. He, K.-K. Wei, What drives continued knowledge sharing? An investigation of knowledge-contribution and -seeking beliefs, Decision Support Systems 46 (2009) 826–838.
[15] H. Holz, H. Maus, A. Bernardi, O. Rostanin, A lightweight approach for proactive, task-specific information delivery, Proceedings of the 5th International Conference on Knowledge Management,(I-Know), Graz, Austria, 2005.
[16] S.Y. Hwang, C.P. Wei, W.S. Yang, Discovery of temporal patterns from process instances, Computers in Industry 53 (2004) 345–364.
[17] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Computing Surveys (CSUR) 31 (1999) 264–323.
[18] S.C. Johnson, Hierarchical clustering schemes, Psychometrika 32 (1967) 241–254.
[19] A.B. Kahn, Topological sorting of large networks, Communications of the ACM 5 (1962) 558–562.
[20] L. Kaufman, P.J. Rousseeuw, Finding groups in data: an introduction to cluster analysis, Wiley, New York, 1990.
[21] S. Kim, H. Hwang, E. Suh, A process-based approach to knowledge-flow analysis: a case study of a manufacturing firm, Knowledge and Process Managment 10 (2003) 260–276.
[22] C.-H. Lai, D.-R. Liu, Integrating knowledge flow mining and collaborative filtering to support document recommendation, The Journal of Systems and Software 82 (2009) 2023–2037.
[23] D.R. Liu, I.C. Wu, Collaborative relevance assessment for task-based knowledge support, Decision Support Systems 44 (2008) 524–543.
[24] D.R. Liu, I.C. Wu, K.S. Yang, Task-based K-support system: disseminating and sharing task-relevant knowledge, Expert Systems with Applications 29 (2005) 408–423.
[25] X. Luo, Q. Hu, W. Xu, Z. Yu, Discovery of textual knowledge flow based on the management of knowledge maps, Concurrency and Computation: Practice and Experience 20 (2008) 1791–1806.
[26] I. Nonaka, H. Takeuchi, The knowledge-creating company: how Japanese companies create the dynamics of innovation, Oxford University Press, 1995.
[27] S.G. Oguducu, M.T. Ozsu, Incremental click-stream tree model: learning from new users for web page prediction, Distributed and Parallel Databases 19 (2006) 5–27.
[28] M. Polanyi, The tacit dimension, Doubleday, New York, 1966.
[29] O.M. Rodriguez, A.I. Martinez, J. Favela, A. Vizcaino, M. Piattini, Understanding and supporting knowledge flows in a community of software developers, International Workshop on Groupware (CRIWG 2004), Springer, 2004, pp. 52–66.
[30] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Information Processing and Management 24 (1988) 513–523.
[31] S. Sun, A. Kumar, J. Yen, Merging workflows: a new perspective on connecting business processes, Decision Support Systems 42 (2006) 844–858.
[32] W.M.P. van der Aalst, A. Weijters, Process mining: a research agenda, Computers in Industry 53 (2004) 231–244.
[33] W.M.P. van der Aalst, T. Weijters, L. Maruster, Workflow mining: discovering process models from event logs, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 1128–1142.
[34] B.F. van Dongen, W.M.P. van der Aalst, Multi-phase process mining: building instance graphs, International Conference on Conceptual Modeling (ER 2004), 2004, pp. 362–376.
[35] C.J. Van RijsBergen, Information Retrieval, Butterworths, London, 1979.
[36] J. Wang, K. Gwebu, M. Shanker, M.D. Troutt, An application of agent-based simulation to knowledge sharing, Decision Support Systems 46 (2009) 532–541.

[37] H.J. Wang, J.L. Zhao, L.-J. Zhang, Policy-Driven Process Mapping (PDPM): discovering process models from business policies, Decision Support Systems 48 (2009) 267–281.
[38] I.C. Wu, D.R. Liu, P.C. Chang, Toward incorporating a task-stage identification technique into the long-term document support process, Information Processing and Management 44 (2008) 1649–1672.
[39] H. Zhuge, A knowledge flow model for peer-to-peer team knowledge sharing and management, Expert Systems with Applications 23 (2002) 23–30.
[40] H. Zhuge, Discovery of knowledge flow in science, Communications of the ACM, ACM Press, New York, NY, USA, 2006, pp. 101–107.
[41] H. Zhuge, Knowledge flow network planning and simulation, Decision Support Systems 42 (2006) 571–592.

Dr. Duen-Ren Liu is a professor of the Institute of Information Management at the National Chiao Tung University of Taiwan. He received the BS and MS degrees in Computer Science from the National Taiwan University and his PhD in Computer Science from the University of Minnesota. His research interests include information systems, knowledge engineering and management, workflow systems, electronic commerce and recommender systems.

Chin-Hui Lai is currently a postdoctoral fellow of the Institute of Information Management at the National Chiao Tung University of Taiwan. She received the MS and the PhD degrees in Information Management from the National Chiao Tung University. Her research interests include information systems, recommender systems, knowledge management, and electronic commerce.