Engineering Applications of Artificial Intelligence 45 (2015) 103–118

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

A new framework for mining frequent interaction patterns from meeting databases

Anna Fariha [a], Chowdhury Farhan Ahmed [b,*], Carson K. Leung [c], Md. Samiullah [a], Suraiya Pervin [a], Longbing Cao [d]

[a] Department of Computer Science and Engineering, University of Dhaka, Bangladesh
[b] ICube Laboratory, University of Strasbourg, France
[c] Department of Computer Science, University of Manitoba, Canada
[d] Faculty of Engineering and Information Technology, University of Technology Sydney, Australia

ARTICLE INFO

Article history:
Received 18 December 2014
Received in revised form
13 June 2015
Accepted 22 June 2015
Available online 16 July 2015

Keywords:
Data mining
Frequent patterns
Directed acyclic graphs
Human interaction
Modelling meetings

ABSTRACT

Meetings play an important role in workplace dynamics in modern life since their atomic components represent the interactions among human beings. Semantic knowledge can be acquired by discovering interaction patterns from these meetings. A recent method represents meeting interactions using tree data structure and mines interaction patterns from it. However, such a tree based method may not be able to capture all kinds of *triggering relations* among interactions and distinguish same interaction from different participants of different ranks. Hence, it is not suitable to find all interaction patterns such as those about correlated interactions. In this paper, we propose a new framework for mining interaction patterns from meetings using an alternative data structure, namely, *weighted interaction flow directed acyclic graph* (*WIFDAG*). Specifically, a WIFDAG captures both temporal and triggering relations among interactions in meetings. Additionally, to distinguish participants from different ranks, we assign *weights* to nodes in the WIFDAGs. Moreover, we also propose an algorithm called *WDAGMeet* for mining weighted frequent interaction patterns from meetings represented by the proposed framework. Extensive experimental results are shown to signify the effectiveness of the proposed framework and the mining algorithm built on that framework for mining frequent interaction patterns from meetings.

1. Introduction

Data mining extracts *frequent patterns* and their relationships by analysing large database aiming towards discovering implicit, previously unknown, and potentially valuable information or knowledge. Frequent substructures are a form of frequent patterns, which can be represented using graphs. Trees and directed acyclic graphs (DAGs) are special kinds of graphs, where tree is undirected acyclic graph and DAG is directed acyclic graph and both *tree mining* and *directed acyclic graph mining* play very important role in the field of structured data mining.

Meetings and human interactions are integral parts of workplace dynamics for communicating among the participating members. During a meeting, several kinds of human interactions may occur. Examples include: proposing an idea, positively or negatively reacting to a proposal, acceptance or rejection of a proposal, etc. To gather significant information regarding the success rate of the decision made in a meeting, one can mine patterns from human interactions occurred in the meeting. Meeting mining refers to the process of finding frequent interaction flow patterns occurred in meetings aiming towards discovering higher level semantic knowledge for further understanding and interpretation of human interactions and relationships among interactions. Meeting interaction mining framework can also facilitate in the fields of traffic control system for congestion detection, stock market analysis, social network group behaviour detection, telecommunication network fraud activity detection, etc.

Frequent pattern mining is helpful in finding frequently occurring patterns such as *interaction patterns* from meetings. The discovered interaction patterns help in estimating the effectiveness of decisions made, designating whether a meeting discussion is fruitful, comparing two meeting discussions using interaction flow as a key feature, mining association rules among interactions and indexing meetings for further ease of access in database. Yu et al. (2009) proposed a multimodal approach for interaction recognition; they (Yu et al., 2012) also used a tree based mining

* Corresponding author. Mobile: +33 629 271 568.
E-mail addresses: anna@cse.univdhaka.edu (A. Fariha), cfahmed@unistra.fr, farhan@cse.univdhaka.edu (C.F. Ahmed), kleung@cs.umanitoba.ca (C.K. Leung), samiullah@cse.univdhaka.edu (Md. Samiullah), suraiyacse@gmail.com (S. Pervin), longbing.cao@uts.edu.au (L. Cao).

http://dx.doi.org/10.1016/j.engappai.2015.06.019
0952-1976/

method to discover frequent patterns from human interactions occurred in meetings. Such a method focuses mostly on capturing direct parent–child relations. However, there are other triggering relations in meetings as illustrated in Example 1, which reflects the shortcomings of the existing tree based meeting mining method (Yu et al., 2012) in an extremely significant scenario.

Example 1. Let us consider a scenario about a meeting of four persons (e.g. professor *A*, assistant professor *B*, and two lecturers *C* & *D*) with different weights/ranks. At the beginning of the meeting, *B* proposes an idea which triggers three interactions: (i) *C* expresses negative opinion towards the proposed idea, (ii) *C* asks *D* for opinion regarding the idea, and (iii) *A* expresses positive opinion towards the idea. Now, the interaction of *C*'s request for *D*'s opinion triggers a single interaction performed by *D*. Although the response of *D* is triggered by *C*'s request of opinion, such a response is generally influenced by *A*'s positive opinion. To elaborate, *D* may initially feel negatively regarding *B*'s proposed idea. But, after listening to *A*'s positive comment, *B* may change his mind and lean towards a neutral or even positive opinion. Fig. 1 shows a sample meeting scenario and interaction triggering relations.

Example 1 reveals the following facts: (i) An interaction can be triggered or influenced by *multiple interactions* and (ii) the extent of influence can be significantly dependent on the *weight/rank* of the person triggering that interaction.

However, the aforementioned tree based method (Yu et al., 2012) is limited to some extent and has the following shortcomings: (i) It does *not capture all kinds of triggering relations* that can occur within a meeting. Specially the triggering relations where an interaction partially triggers another interaction previously triggered by some other interaction. For example, in Example 1, the triggering relation between *A*'s positive opinion and *D*'s positive opinion is not captured in the tree based method (Yu et al., 2012). (ii) It does *not consider the ranks* of participants. (iii) The algorithm produces *less significant* frequent patterns while *missing* some important frequent patterns. This lacking motivated us to work on overcoming the problems of existing tree based meeting mining method. Existing DAG mining algorithms do not consider weights of the nodes in DAGs, let alone mining *weighted* frequent interaction patterns. Observing that (i) DAGs and trees are both specializations of graphs and (ii) trees may not capture all triggering relations and person's weight/rank, we explore the usage of DAGs (as alternatives to trees) for modelling meetings. As interactions occur in meetings flow in only one direction with respect to time (i.e. no cycle), DAG would be a logical choice for modelling meetings.

The key contributions of this paper are as follows:

1. A novel DAG based mining framework for modelling and mining interactions occurred in meetings is proposed that captures two kinds of relations: (i) temporal relations and (ii) triggering relations (cf. trees capture temporal relations but not all triggering relations).
2. The feature of node *weight* is incorporated in the framework to preserve the rank information of participants and thus making the representation more realistic.
3. *WDAGMeet* algorithm for *W*eighted *DAG* based *Meet*ing mining in the new proposed framework is devised by integrating DAG based mining (Werth and Dreweke, 2008) and interaction pattern mining with weighted frequent pattern mining (Ahmed et al., 2009, 2012).
4. WDAGMeet algorithm maintains the anti-monotone property by exploiting a new strategy and prunes a large number of candidate patterns efficiently.
5. Elaborate descriptions with application in real-life scenarios and advantages of WDAGMeet algorithm over existing tree based meeting mining algorithms are provided.
6. An extensive performance study is conducted with different variations of our proposed approach in order to show its significance, efficiency, correctness and effectiveness.

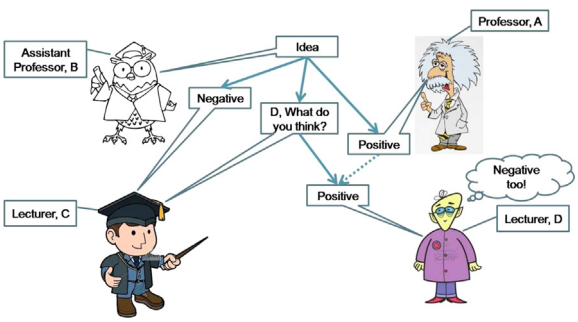


Fig. 1. A motivating scenario.

The rest of the paper is organized as follows: Section 2 describes related works and provides necessary background knowledge. Section 3 introduces our proposed framework to represent meetings using weighted directed acyclic graphs. Section 4 presents the *WDAGMeet* algorithm and simulation of the algorithm. Section 5 focuses on the performance of our proposed algorithm and comparison with the existing algorithm. Section 6 discusses how the proposed meeting mining framework can facilitate in many other real life fields and how it can be applied there. Finally, Section 7 includes ideas about future scope of improvement and concludes the paper.

2. Related work

Transactional data such as sales, cost, inventory, etc. can be mined to discover hidden patterns (Alavi and Hashemi, 2015; Duong et al., 2014; Han et al., 2004). For acquiring further interesting and realistic patterns, weighted frequent pattern (Ahmed et al., 2012; Yun et al., 2012), weighted periodic pattern (Yang et al., 2014), high utility pattern (Ahmed et al., 2009) and data stream (Yun et al., 2014) mining methods were developed, but most of them are limited to transactional databases or streaming databases. *Directed Acyclic Graphs* (DAGs) are useful data structure to represent special kind of data, like procedural abstraction, code compaction, event sequences, etc. Chen et al. (2004) performed one of the earliest works on mining DAG patterns from DAG databases. Campagna and Pagh (2010) worked on finding frequent *traces* among all the paths in the DAGs, by limiting it with a maximum length. Termier et al. (2007) presented as the first algorithm to mine closed frequent embedded sub-DAGs. Werth and Dreweke (2008) designed and implemented a DAG miner for mining DAGs from DAG databases and used it for procedural abstraction. They also demonstrated the quantitative effects of DAG mining in program size reduction. But, none of them has developed any method based on weighted DAG mining to mine interaction patterns from meetings. Although Geng et al. (2009) and Lee and Yun (2012) mined weighted frequent subgraphs from graph databases, to the best of our knowledge, there exists no weighted DAG mining algorithm in the existing literature and algorithms like the one in Werth and Dreweke (2008) prepared for specialized graph (DAG), outperform general purpose graph mining algorithms that work for all sort of graphs.

To acquire the semantic information from a meeting, researchers extracted the meeting contents and represented them in a machine readable format. Kolár et al. (2010) used dialogue act

segmentation and dialogue act classification using simple lexical and prosodic knowledge sources in automatic meeting applications. Zhang et al. (2006) expressed group actions as a two layer process by hidden Markov model (HMM) framework. Waibel et al. (1998) presented a meeting browser using automatic speech transcription, dialogue summarization and the extraction of verbal and non-verbal cues that describe the dynamics of human interaction.

Nijholt et al. (2006) and Otsuka et al. (2007) considered turn-taking, gaze behaviour, head gestures, and utterances for detecting interactions. Yu et al. (2009) proposed a multimodal approach and used support vector machine (SVM), Bayesian Network, Naïve Bayes, and decision tree for interaction recognition to infer human semantic interactions in meeting discussions. Kipp (2001) proposed a widely used video annotation tool that allows user-defined hierarchical layers and attributes.

Several research works were proposed to discover patterns in human behaviours. Magnusson (2000) contributed in discovering hidden time patterns in behaviour. Later, Morita et al. (2005) proposed a pattern mining method for interpretation of interaction by extracting simultaneously occurring patterns of primitive actions such as gaze and speech. Tomobe and Nagao (2006) worked to extract discussion flow for knowledge discovery from human activities in meetings. Yu et al. (2012) proposed a tree based mining method to discover frequent patterns from human interactions occurred in meetings where tree was used to model interaction flow in meetings. But their approach excludes some triggering relations among interactions and also discards participant ranks.

To overcome the shortcomings in the existing approaches, we propose a new framework and an algorithm for mining frequent interaction patterns from meeting databases. Moreover, our substantively new and different contributions beyond our preliminary conference version (Fariha et al., 2013) include the proposal of a new framework to define the representation model and mining backbone more accurately, enhanced motivation, revised and optimized mining algorithm, scope of real-life applications, rigorous analysis of the mining algorithm with step by step simulation, and extensive analysis of performance on synthetic and real-life databases.

3. A novel framework for mining meeting databases

The first step of any mining process is to represent the raw data in machine readable format using such data structure that preserves the basic structure of the data. In this section, we propose a novel framework that contains the process towards a complete weighted directed acyclic graph based representation of the meetings and the idea of representing triggering relation among several interactions using directed edges.

3.1. Semantic classes of human interactions

In a meeting session, there can be thousands of sentences and interactions. If we take all human interactions as distinct elements, then the mining process will be worthless. As semantically similar sentences often sound different from person to person, it is necessary to classify all human interactions into some major classes. Some general classes of human interactions (Yu et al., 2012) are taken into account for this framework as stated as follows:

1. **PRO:** Propose an idea.
2. **ASK:** Ask for opinion regarding a proposal.
3. **POS:** Positive opinion towards a proposal.
4. **NEG:** Negative opinion towards a proposal.
5. **ACK:** Acknowledgement of another interaction.
6. **COM:** Comment on another interaction.
7. **REQ:** Request information regarding an issue.
8. **ACC:** Accept the proposed idea.
9. **REJ:** Reject the proposed idea.

When building a model to represent the interaction flow occurred in meetings, we assign a class label to each interaction with a labelling function, $l : V \rightarrow L$, where V denotes the vertex set, each vertex stands for one single interaction, and $L = \{PRO, ASK, POS, NEG, ACK, COM, REQ, ACC, REJ\}$. Table 1 illustrates the idea of labelling each interaction, occurred in a faculty meeting of a university, with specific labels.

3.2. Weight assignment of human interactions

In a meeting, various participants can express similar interactions, but not all of them should be counted equally important. If we only assign a semantic class label to each interaction, the information regarding the person who initiated the interaction is lost. As a result, the **ACC** of a chairman and the **ACC** of a junior lecturer will be indistinguishable. To further specify the rank of the person who initiated the interaction, we incorporate the idea of assigning a weight to each interaction with a value ranging from 1 to n, inclusive. Standard value of n ranges from 3 to 5, because having too many distinct weights may lead to poor number of frequent patterns.

A response from a person having a heavier weight usually strongly influences the decision making process than that from a person of a lighter weight. Table 1 shows the result of assigning weights to each interaction to signify the rank of the person causing that interaction. Here, for our running example based on university faculty hierarchy, we fix $n = 5$.

3.3. Triggering relations among human interactions in a meeting

From the perspective of spontaneity, interactions can be categorized into two types:

1. *Triggering interaction:* Interactions that cause some other interactions to occur. A triggering interaction can be a spontaneous interaction or another triggered interaction.
2. *Triggered interaction (reactive interaction):* Interactions that occur in the reaction of some other interaction.

Fig. 2 shows the graphical representation of interaction flow showing labelling of interactions, weight assignment of each labelled interactions and triggering relations on the meeting session of Table 1. The solid arrows denote *direct triggering relations*, i.e. the triggering interactions that caused immediate triggering of the triggered interactions. The dashed arrows denote

Table 1
Labelling each interaction with a semantic class label and weight assignment.

Designation	Weight	Interaction	Label
Lecturer *A*	1	"I think we should start open credit system."	**PRO**
Professor *B*	4	"Yes! That's a good idea."	**POS**
Professor *C*	4	"But lot's of overheads to be considered."	**NEG**
Chairman *D*	5	"What do you think regarding this?"	**ASK**
Assoc. Prof. E	3	"Most of the schools are applying this."	**COM**
Assoc. Prof. F	3	"Yes! We should go for it too."	**POS**
Lecturer *G*	1	"Like professor C said, it is very complex for our school."	**NEG**
Chairman *D*	5	"Okay! We will start it from next semester!"	**ACC**

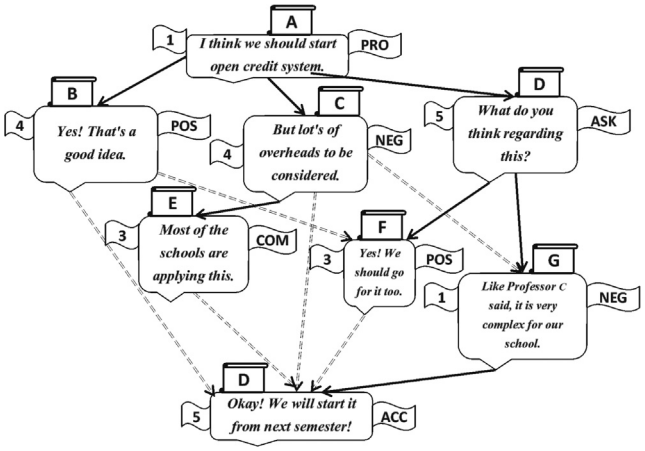


Fig. 2. Graphical representation of interaction flow occurred in a meeting of Table 1 showing labelling of interactions, weight assignment and triggering relations.

indirect triggering relations, i.e. triggering interactions that influenced the triggered interactions in some way, but not directly. With approach of Yu et al. (2012), only the edges of solid arrows would be considered, missing important relations such as *C*'s *NEG* opinion triggers *G*'s *NEG* opinion. Here *G* answers to *D* hence the solid arrow between *D* and *G*, but in fact *G* is also influenced by *C* (dashed arrow *C*–*G*) and this cannot be captured by a tree but is worth of interest.

3.4. Weighted DAG based interaction flow model

From Example 1, it is clear that a tree cannot hold all kinds of triggering relations, because, in a tree, one node can have at most one parent. This constraint implies that an interaction can be triggered by at most one interaction. However, many situations show that one interaction can be triggered or influenced by more than one interaction. Each triggering interaction can contribute partially to the triggering event of the triggered interaction. As (i) interactions only flow in one direction with respect to time, and (ii) no future interaction can trigger any past interaction, no cycle can ever occur in the triggering relationship among several interactions. As a result, DAG is a logical and appropriate choice for representing interaction flow in meetings. Moreover, to signify the rank of the participants of a meeting, the notion of assigning weights to each node of DAG is also integrated in the framework, i.e. *Weighted DAG*.

Example 2. Each weighted and labelled node in the weighted DAG in Fig. 3 denotes an instance of human interactions occurred in a meeting. In the figure, the ***PRO*−*1*** node (the number after the interaction label denotes the weight of the person causing that interaction) reflects a triggering interaction, which represents a lecturer proposing an idea spontaneously. The remaining nodes (***POS*−*4***, ***NEG*−*4***, ***ASK*−*5***, ***COM*−*3***, ***POS*−*3***, ***NEG*−*1***, ***ACC*−*5***) reflect triggered interactions, which occur in response to the triggering interactions. Directed edges between nodes indicate triggering relations between the nodes, and the arrows point from the triggering interaction to the triggered one. Consequently, we generate a weighted DAG based interaction flow diagram for modelling meetings.

Besides those triggering relations, our proposed weighted DAG based representation represents the temporal relations by topological levels. Here, a collection of vertices of a DAG are said to be at the same topological level if and only if no vertex within that topological level is ancestor of any other vertex lying in the same topological level and all the ancestors of the vertices belong to previous topological level. Nodes of a certain topological level appear temporally before nodes of the next/lower level. Within the same topological level, the node on the left appears temporally before the node on the right.

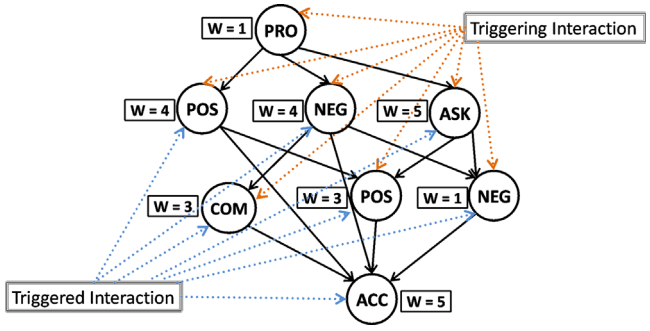


Fig. 3. Weighted DAG based representation of interaction flow of meeting database in Table 1 showing both triggering relations and temporal relations.

Example 3. Fig. 3 shows a sample session of a faculty meeting, in which a lecturer *A* (weight=1) proposes an idea. Triggered by *A*'s proposed idea, a professor *B* (weight=4) expresses her positive opinion and then another professor *C* (weight=4) expresses his negative opinion towards the proposal. After that, the chairman *D* (weight=5) asks others' opinions. Then an associate professor *E* (weight=3) comments on *C*'s negative opinion. In response to *D*'s asking, an associate professor *F* (weight=3) expresses his positive opinion, which was also inspired by *B*'s positive opinion. After that, another lecturer *G* (weight=1) expresses her negative opinion in response to *D*'s asking, which was also inspired by the previous negative opinion of professor *C*. Finally, based on two negative opinions from professor *C* and lecturer *G* as well as two positive opinions from professor *B* and associate professor *F*, chairman *D* (weight=5) accepts *A*'s idea, biased to the interaction performed by persons of higher rank. Note that, the weighted DAG in Fig. 3 captures not only single triggering relations but also interactions triggered by multiple triggering interactions.

Fig. 3 also shows the weighted DAG based representation of interaction flow of meeting database in Table 1. It is also clear that the indirect triggering relations in Fig. 2 are now incorporated as directed edges between nodes in this representation. Note that representing multiple triggering relations to a single triggered interaction was previously impossible to represent using tree based representation.

We define the basic key structures to represent meetings in this work in Definitions 1–3. Definitions 4 and 12 are basic definitions for frequency counting of WIFDAGs. Definitions 5–9 define canonical encoding of a WIFDAG, that will be useful for enumeration. Definitions 10, 11, 13–18 allow to compute frequent patterns while taking weight into account.

Definition 1 (*Session*). A session is a unit of a meeting that begins with a spontaneous interaction and concludes with an interaction that is not followed by any reactive interaction (Yu et al., 2012). One single meeting may consist of several sessions.

Definition 2 (*Weighted Interaction Flow DAG, WIFDAG*). Interaction flow within interactions of a meeting session can be represented by a weighted and labelled DAG $D=(V_d, E_d, L_d, W_d)$, where $V_d=\{v_1, v_2, \ldots, v_n\}$, a set of n vertices and $E_d=\{e_1, e_2, \ldots, e_m\}$, a set of m directed edges. Labelling function $L_d: V_d \rightarrow L$ is used to assign each node v_i to a class label $l(v_i)$, where $l(v_i) \in L=\{PRO, ASK, POS, NEG, ACK, COM, REQ, ACC, REJ\}$. Weight function W_d associates each node with a weight $W_d(v_i)$,

$0 \le W(v_i) \le$ MaxWeight, where MaxWeight denotes maximum possible weight of a participant, that carries information regarding the (absolute or relative) rank of a participant who initiates an interaction in a meeting. Each edge is a directed connection between two vertices, i.e. $e_k = \{(v_i, v_j) | 1 \le i, j \le n; v_i, v_j \in V_d\}$. Here, v_i denotes the source/origin of the directed edge and v_j denotes the destination of that edge. An edge from v_i to v_j implies that v_i (directly or indirectly) triggers v_j. All DAGs are connected acyclic graphs and no two DAGs, representing sessions of the same meeting, are connected to each other. This *W*eighted *I*nteraction *F*low *DAG* is called *WIFDAG*.

Definition 3 (*Topological level of a WIFDAG*). Given a WIFDAG, $G(V_g, E_g, L_g, W_g)$, topological level i is defined by a set of vertices, V_i, where $\forall e_j(u_j, v_j) \in E_g, v_j \in V_i \rightarrow u_j \in V_k \wedge k < i$ and $\forall v_m \in V_i, \exists u_m \in V_{i-1} \wedge e_m(u_m, v_m) \in E_g$.

Definition 4 (*Meeting WIFDAG database DB*). Meeting WIFDAG database, denoted by *DB*, consists of several WIFDAGs. $DB = \{G_1, G_2, G_3, \ldots, G_n\}$, where each G_i stands for a specific WIFDAG, representing a meeting session. Size of *DB* is denoted by $|DB|$ and stands for the number of WIFDAGs that *DB* contains.

Definition 5 (*Induced sub-WIFDAG and super-WIFDAG*). Consider two WIFDAGs $S = (V_s, E_s, L_s, W_s)$ and $T = (V_t, E_t, L_t, W_t)$. S is an induced *sub-WIFDAG* or sub-WIFDAG of T, if there exists an injective homomorphism $\psi : S \rightarrow T$ such as: (i) for two vertices $v_s \in V_s$ and $v_t \in V_t$ such that $\psi(v_s) = v_t$, $L_s(v_s) = L_t(v_t)$ and $W_s(v_s) = W_t(v_t)$; (ii) for each directed edge, $(u_s, v_s) \in E_s$, there exists a directed edge from $\psi(u_s)$ to $\psi(v_s)$. Conversely, T is a *super-WIFDAG* of S.

Definition 6 (*Node insertion/traversal signature in WIFDAG*). Insertion/traversal of each node n in a WIFDAG $D = (V_d, E_d, L_d, W_d)$ can be expressed with a 7-tuple $(NE, L_1, W_1, I_1, L_2, W_2, I_2)$, where $NE = 0$ for node insertion. $L_1 = 0, W_1 = 0, I_1 = 0$ for node insertion. $L_2 =$ index of $(L(n)), W_2 = W_d(n)$ and $I_2 =$ index of n. Index of label denotes the position of that label in L, the set of all possible labels, starting from 1; index of n denotes the insertion order of n, starting from 1. Earlier the node is inserted or traversed, less the index will be. For example, if a node is inserted first in a WIFDAG, it has a label *NEG* with label index 4 and the person who caused this interaction is assigned a weight of 3, then the signature for this node will be $(0, 0, 0, 0, 4, 3, 1)$. Fig. 4 denotes the indices for each class label $l \in L$.

Definition 7 (*Edge insertion/traversal signature in WIFDAG*). Insertion/traversal of each edge $e = (v_i, v_j)$ in a WIFDAG $D = (V_d, E_d, L_d, W_d)$ can also be expressed with a 7-tuple $(NE, L_1, W_1, I_1, L_2, W_2, I_2)$, where $NE = 1$ for edge insertion. $L_1 =$ index of $(L(v_i)), W_1 = W_d(v_i)$ and $I_1 =$ index of v_i, $L_2 =$ index of $(L(v_j)), W_2 = W_d(v_j)$ and $I_2 =$ index of v_j. Node and edge insertion signatures of the WIFDAG of Fig. 5 can be found, sorted according to the time stamp they were inserted, at the bottom of the figure. ith element of an insertion signature S is denoted by $S(i)$.

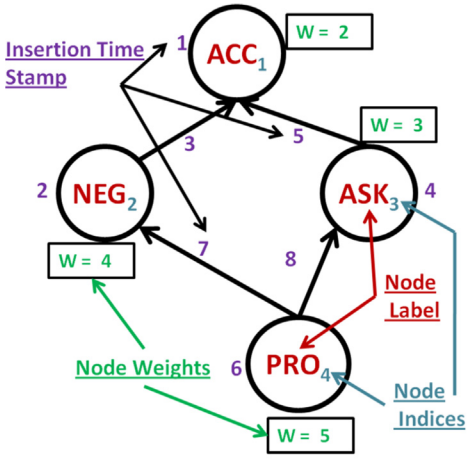
Label	Index
PRO	1
ASK	2
POS	3
NEG	4
ACK	5
COM	6
REQ	7
ACC	8
REJ	9

Fig. 4. Label index table.

Definition 8 (*Canonical description of WIFDAG*). Canonical description of WIFDAG states the order of insertion or traversal of nodes and edges to a WIFDAG. It is a list of ordered insertion signatures of nodes and edges according to insertion time stamp. Fig. 5 illustrates the idea of canonical description precisely. Insertion signature at time stamp i of a canonical description C can be denoted using $IS(C, i)$.

Definition 9 (*Canonical WIFDAG fragment*). A WIFDAG fragment is a canonical WIFDAG fragment if there is no insertion order of nodes and edges with a bigger canonical description. All other WIFDAG fragments, isomorphic to the canonical WIFDAG fragment, are called *duplicate WIFDAG fragments*. A canonical description C is bigger than another canonical description C' if there exists a time stamp i such that $IS(C, i) > IS(C', i)$ and $IS(C, k) = IS(C', k)$ for all $k < i$. An insertion signature S is bigger than another insertion signature S' if there exists a number i such that $S(i) > S'(i)$ and $S(k) = S'(k)$ for all $k < i$. For example, in Fig. 5, let us name the WIFDAG shown as G, where the node labelled **ACC** (insertion time stamp 1) was inserted before the node labelled **NEG** (insertion time stamp 2). If the later node (**NEG**) was inserted earlier, we can produce another isomorphic WIFDAG, G', and the canonical description of G' is (0,0,0,0,4,4,1), (0,0,0,0,8,2,2), (1,4,4,1,8,2,2), (0,0,0,0,2,3,3), (1,2,3,3,8,2,2), (0,0,0,0,1,5,4), (1,1,5,4,4,4,1), (1,1,5,4,2,3,3). We can see that, $IS(G, 1) > IS(G', 1)$, because, $IS(G, 1)(5) > IS(G', 1)(5)$, as $8 > 4$. It can be shown that, among all other possible node and edge insertion order to generate the same WIFDAG shown in Fig. 5, the one shown in the figure holds the biggest canonical description and hence, is the canonical WIFDAG fragment.

Definition 10 (*Canonical form of WIFDAG*). For a WIFDAG fragment, D, there exists another WIFDAG fragment, I_D such that, I_D is isomorphic to D and I_D is a canonical WIFDAG fragment. The canonical description of I_D is the canonical form of D. The canonical form is unique for all duplicate isomorphic WIFDAGs. If the canonical description of a WIFDAG fragment is not equal to



Canonical Description :
(0,0,0,0,8,2,1), (0,0,0,0,4,4,2), (1,4,4,2,8,2,1), (0,0,0,0,2,3,3), (1,2,3,3,8,2,1), (0,0,0,0,1,5,4), (1,1,5,4,4,4,2), (1,1,5,4,2,3,3)

Fig. 5. Canonical description of a sample WIFDAG.

its canonical form, then we discard that duplicate WIFDAG fragment for further consideration.

Definition 11 (*Absolute weight of a WIFDAG, Weight*). For a WIFDAG, $D=(V_d, E_d, L_d, W_d)$, the absolute weight or weight of D is defined as follows:

$$\text{Weight}(D)=\frac{\sum_{v\in V_d} W_d(v)}{|V_d|} \quad (1)$$

Definition 12 (*Normalized weight of a WIFDAG, N-Weight*). As same absolute weight can possess different significance in different scenarios, we use normalized weight of a WIFDAG, N-Weight to transform the absolute weight of that WIFDAG to a real value ranging from 0 to 1, inclusive. For normalization, we define MaxWeight as the maximum possible weight for an individual node for that scenario. For a WIFDAG, $D=(V_d, E_d, L_d, W_d)$, the N-Weight of D is defined as follows:

$$\text{N–Weight}(D)=\frac{\text{Weight}(D)}{\text{MaxWeight}} \quad (2)$$

Fig. 6 illustrates the calculation of absolute and normalized weight of a WIFDAG.

Definition 13 (*Support Sup*). Given (i) a sub-WIFDAG D' and (ii) a database DB, the support, Sup of D' (usually expressed in percentage) is defined by the following equation:

$$\text{Sup}(D')=\frac{\#\ \text{Super–WIFDAGs of } D'}{|DB|} \quad (3)$$

Definition 14 (*Weighted support, W-Sup*). Given a sub-WIFDAG D', the weighted support, W-Sup of D' is defined by the following equation:

$$\text{W–Sup}(D')=\text{Sup}(D')\times\text{N–Weight}(D') \quad (4)$$

Weighted support is also usually expressed in percentage.

Definition 15 (*Minimum weighted support, Min-WSup*). For mining a WIFDAG database, a specific threshold is defined for weighted support and is called minimum weighted support, Min-WSup. It is worthy to note that, as highest normalized weight of a WIFDAG can be at most 1, some WIFDAG fragments might be frequently having low normalized weight. These WIFDAG fragments result into lower weighted support than its original support. As a result, Min-WSup must be defined a bit lower than minimum support to get the desired result in frequent patterns. We can calculate Min-WSup by multiplying minimum support with average N-Weight of WIFDAGs in the database.

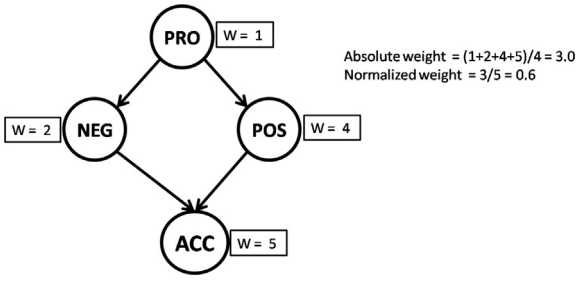


Fig. 6. Calculation of absolute weight and normalized weight of a WIFDAG.

Definition 16 (*Frequent WIFDAG patterns*). WIFDAGs, having weighted support greater than or equal to the user-specific minimum weighted support threshold Min-WSup are considered *frequent WIFDAG patterns*.

Definition 17 (*Semi-frequent WIFDAG patterns*). Given (i) a WIFDAG fragment D, (ii) maximum possible number of nodes in a frequent WIFDAG N, (iii) minimum weighted support Min-WSup, *maximum possible weighted support*, MPWS(D) is defined as follows:

$$\text{MPWS}(D)=\frac{\text{N–Weight}(D)\times|D|+(N-|D|)}{N}\times\text{Sup}(D) \quad (5)$$

D is called *semi-frequent WIFDAG pattern* if the following conditions hold:

$$\text{W–Sup}(D)<\text{Min–WSup}. \quad (6)$$

$$\text{MPWS}(D)\geq\text{Min–WSup}. \quad (7)$$

Definition 18 (*Infrequent WIFDAG patterns*). Given (i) a WIFDAG fragment D and (ii) minimum weighted support Min-WSup, D is called *infrequent WIFDAG pattern* if the following two conditions hold:

$$\text{W–Sup}(D)<\text{Min–WSup}. \quad (8)$$

$$\text{MPWS}(D)<\text{Min–WSup}. \quad (9)$$

Property 1. *WIFDAG can represent indirect triggering relations where tree-based method fails.*

Discussion. As tree does not allow multiple parent of a node, if we consider that tree can represent multiple triggering relations, we have to assume indirect triggering relations are propagated through levels, i.e. a node indirectly triggers all of its descendants except its direct children (a node triggers its children directly). Example 4 provides a scenario that explains the failure of tree-based method (Yu et al., 2012) in distinguishing different cases of multiple triggering relations and how using WIFDAG can overcome this.

Example 4. Consider a small meeting snapshot in Fig. 7(a). Tree based method will represent the meeting as in Fig. 7(b) and WIFDAG based proposed framework will represent as in Fig. 7(c). Now let us consider another small meeting snapshot in Fig. 8(a). Tree based method will represent the meeting as in Fig. 8(b) and WIFDAG based proposed framework will represent as in Fig. 7(c).

In Fig. 8(a), lecturer *C* opposes only the idea of "Robotics" lab, he does not oppose the idea of a "New Lab" (interaction by professor *A*). Hence, lecturer *C* only expresses ***NEG*** opinion in response to the ***POS***

a

Timestamp	Designation	Weight	Interaction	Label
1	Professor A	4	Let's start a new Lab	PRO
2	Lecturer B	1	Yes, the lab should be about robotics.	POS
3	Lecturer C	1	Robotics is not good, I think bioinformatics is a good field of study now-a-days.	NEG

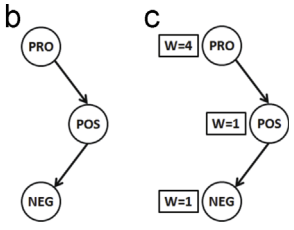


Fig. 7. Representing a meeting snapshot using both tree and WIFDAG. (a) A sample meeting. (b) Representation of meeting in (a) using tree. (c) Representation of meeting in (a) using WIFDAG.

a

Timestamp	Designation	Weight	Interaction	Label
1	Professor A	4	Let's start a new Lab	PRO
2	Lecturer B	1	Yes, the lab should be about robotics.	POS
3	Lecturer C	1	No, I don't think any new lab will be good for robotics, there should just be a few courses about robotics.	NEG

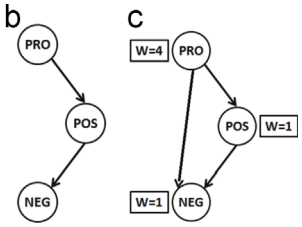


Fig. 8. Representing another meeting snapshot using both tree and WIFDAG. (a) A sample meeting. (b) Representation of meeting in (a) using tree. (c) Representation of meeting in (a) using WIFDAG.

opinion of lecturer *B*. Please note that, lecturer *C* was not influenced by professor *A* in the meeting in Fig. 7(a), that is why both the tree in Fig. 7(b) and the WIFDAG in Fig. 7(c) representations are similar for that meeting. Also note that, the ***PRO−4*** node does not trigger ***NEG−1*** node indirectly. If tree representation was interpreted in a way such that a node indirectly triggers all of its descendants, then the interpretation would have been wrong, because there is no indirect triggering relationship from ***PRO−4*** node to ***NEG−1*** node.

On the other hand, let us have a look at Fig. 8(a). In this case, lecturer *C* opposes the idea of "New Robotics Lab". Here, lecturer *C* opposes both the ideas of professor *A* ("New Lab") and lecturer *B* ("Robotics Lab"). Hence, lecturer *C* is influenced by both professor *A* and lecturer *B*. As tree does not support multiple parents, it would represent this meeting the same way it represented the meeting in Fig. 7(a), which obviously illustrates the limitation of tree models to distinguish between cases where there is an indirect relation (for later scenario in Fig. 8(a)) and where there is no indirect triggering relation (for former scenario in Fig. 7(a)). But if we represent the meeting of Fig. 8(a) using WIFDAG (shown in Fig. 8(c)), both ***PRO−4*** and ***POS−1*** nodes have triggering relations to ***NEG−1*** node, which leads to the correct interpretation.

It is clear that, although the two meetings in Fig. 7(a) and Fig. 8(b) are different, their tree based representations are same while WIFDAG based representations differ. So it is clearly shown that the tree fails to distinguish between certain scenarios where WIFDAG succeeds. In summary, trees cannot distinguish between cases where a predecessor node indirectly triggers a node and where it does not, but, a WIFDAG can distinguish these two cases by explicitly adding triggering edges where necessary.

Property 2. *The definition of W-Sup includes more interesting semi-frequent patterns.*

Proof. The definition of W-Sup allows us to discover frequent WIFDAG fragments containing nodes that may not occur too frequently but are associated with heavy weights. This property of weighted support allows us to mine more significant WIFDAG fragments where excluding less significant WIFDAG fragments. Example 5 explains the reason to choose weighted support of a WIFDAG fragment rather than support for the mining process to find frequent WIFDAG fragments. □

Example 5. Consider a sample WIFDAG database *DB*, consisting of 7 WIFDAGs, $DB = \{D_1, D_2, \ldots, D_7\}$ shown in Fig. 9. Here, Min-WSup=36%. In the first three sessions, associate professor *A* (weight=4) proposes ideas (**PRO**), and each of them are rejected (**REJ**) by professor *B* (weight=5). In the last 4 sessions, lecturer *C* (weight=1) makes comments (**COM**), and each of them triggers lecturer *F*'s (weight=1) comments (**COM**). Here, the frequency of the pattern "*A* proposes an idea, which is rejected by *B*" is 3; the frequency of another pattern "*C* makes a comment, which is commented by *F*" is 4. Between them, the first pattern is more interesting than the second one because interactions between persons of higher rank (i.e. heavier weights) are usually more important and useful in analysing decision-making meetings even when the frequency of these interactions is not too high. Although **PRO-4** node is not frequent, but it has MPWS > Min–WSup. As a result, it is considered for further expansion and marked as a semi-frequent pattern. This pattern later generates a frequent pattern. On the other hand **COM-1** node has MPWS < Min–WSup and is marked as an infrequent pattern. We can safely discard it for further consideration, because, it cannot result into a frequent pattern even when it is expanded by the maximum weighted node in any future iteration. The calculation of support, weighted support and finding frequent WIFDAG fragments is shown in Fig. 9.

Definition 19 (*Mining frequent interaction patterns from meeting WIFDAG database*). Given (i) a meeting WIFDAG database *DB*, consisting of WIFDAGs of sessions of several meetings, (ii) a user-specific minimum weighted support threshold, Min-WSup, the problem of mining frequent interaction patterns is to discover from *DB* every frequent WIFDAG fragments, i.e. every sub-WIFDAG D' such that W–Sup(D') ≥ Min–WSup. In this paper, we have used the terms "frequent interaction pattern" and "interaction pattern" interchangeably.

The provided novel framework in this section serves as the structure for developing algorithms for mining frequent interaction patterns from meeting databases.

4. WDAGMeet – an algorithm for mining frequent interaction patterns from meetings

For mining frequent interaction patterns from *W*eighted *DAG* based *Meet*ing databases, a new algorithm *WDAGMeet* is proposed in this section.

4.1. Expansion rules

WDAGMeet algorithm discovers frequent interaction patterns in the form of frequent sub-WIFDAGs from WIFDAG *DB* by first generating a set of all frequent nodes in *DB*. It then expands these nodes (i.e. singleton sub-WIFDAGs) using the four expansion rules topologically. These expansion rules are described in detail below.

New root: New root expansion rule can only be applied if the current WIFDAG fragment has nodes inserted only in its first topological level. This rule is applied recursively by inserting new root nodes, with no edge attached to it, to the current (level 1) topological level of the WIFDAG fragment. Duplicates are avoided by defining an order according to insertion signature.

New level: A new topological level is introduced with the insertion of a new node into that level and the insertion of an edge from a node in the previous topological level. The insertion of the edge is necessary to ensure that a new topological level is introduced. Insertion signature order constraint is also applied in new level expansion rule. Fig. 10(a) explains the new level expansion rule by introducing a new level to a WIFDAG that was created using new root expansion rule.

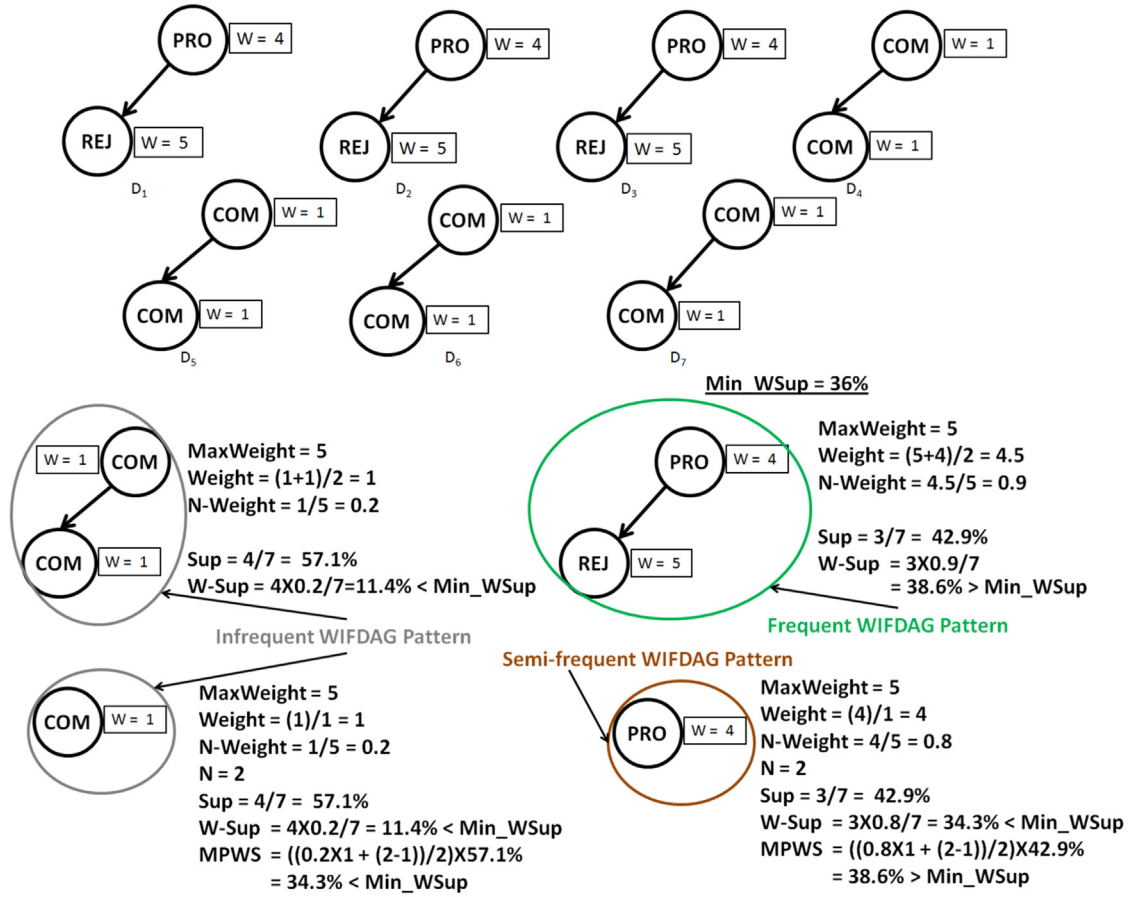


Fig. 9. Significance of the property of weighted support.

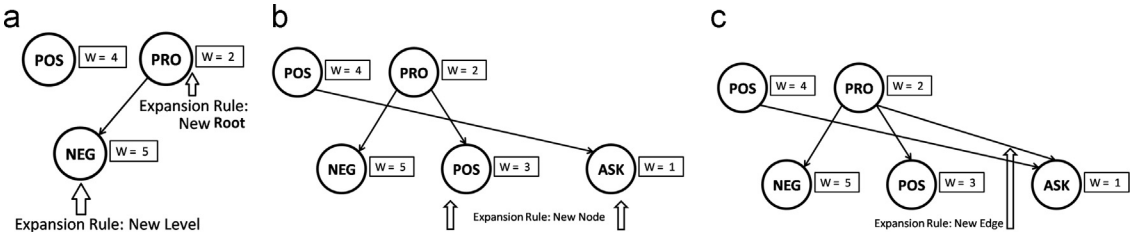


Fig. 10. Expansion rules. (a) New root and new level expansion rule. (b) New node expansion rule. (c) New edge expansion rule.

New node: This expansion rule allows the insertion of a new node to the current topological level. For the same topological level, nodes must be inserted (along with a connecting edge from previous topological level) according to insertion signature order constraint, i.e. the node that will have the highest insertion signature, after inserting as a new node, among all the candidate nodes, will be picked first. Fig. 10(b) explains the new node expansion rule with the insertion of two new nodes in the same topological level to the WIFDAG fragment of Fig. 10(a).

New edge: This expansion rule allows inserting a new edge from a previously inserted node to the most recently inserted node. The destination node of this edge must be the latest inserted node. This expansion rule is lack of the property of not generating duplicate fragments. Fig. 10(c) explains the new edge expansion rule with the insertion of a new edge to the latest inserted node from Fig. 10(b).

4.2. The mining process of WDAGMeet algorithm

WDAGMeet first identifies frequent and semi-frequent weighted nodes from the given WIFDAG database. After constructing the initial frequent node set, the expansion rules are applied recursively in depth first order to generate new frequent WIFDAG patterns.

The pseudo code provided in Algorithm 1 illustrates the steps of WDAGMeet mining algorithm. Note that, WDAGMeet algorithm only inserts frequent *connected* WIFDAGs to the resulting frequent pattern set. The disconnected fragments are used as input to expansion rules for further processing. The algorithm repeatedly generates new candidate patterns using above expansion process until any new expansion is impossible. Also note that, the non-canonical fragments are pruned from the candidate set and are not checked for the possibility of them being frequent or semi-frequent to avoid generating duplicate frequent patterns.

```
Input  : (1) Meeting database of WIFDAGs, DB (2) User defined minimum weighted support threshold,
         Min-WSup
Output: A set of frequent WIFDAG interaction patterns, F
Modules:
(1) CanonicalForm(f ) - Returns the canonical form of fragment f
(2) CanoncialDescription(f ) - Returns the canonical description of fragment f
(3) IsInfrequent(DB, f , Min-WSup) - Returns whether fragment f is infrequent in DB w.r.t Min-WSup
(4) IsFrequent(DB, f , Min-WSup) - Returns whether fragment f is infrequent in DB w.r.t Min-WSup
(5) IsSemiFrequent(DB, f , Min-WSup) - Returns whether fragment f is semi frequent in DB w.r.t Min-WSup
(6) IsConnected(f ) - Returns whether fragment f is connected WIFDAG or not
begin
    F ← all the frequent weighted nodes in all the WIFDAGs in DB
    SF ← all the semi-frequent weighted nodes in all the WIFDAGs in DB
    tempPatterns ← F
    tempPatterns ← tempPatterns ∪ SF
    while tempPatterns ≠ ∅ do
        cnd ← ∅
        currentFP ← ∅
        currentSemiFP ← ∅
        foreach f ∈ tempPatterns do
            cnd ← cnd ∪ expandWithRoots(DB, f)
            cnd ← cnd ∪ expandWithNewLevel(DB, f)
            cnd ← cnd ∪ expandWithNewNode(DB, f)
            cnd ← cnd ∪ expandWithNewEdge(DB, f)
        end
        foreach f ∈ cnd do
            if CanonicalForm(f) ≠ CanoncialDescription(f) then
                cnd = cnd − f
            end
            else if IsInfrequent(DB, f, Min-WSup) then
                cnd = cnd − f
            end
            else if IsFrequent(DB, f, Min-WSup) then
                if IsConnected(f)) then
                    F = F ∪ f
                end
                currentFP ← currentFP ∪ f
            end
            else if IsSemiFrequent((DB, f, Min-WSup) then
                if IsConnected(f)) then
                    SF = SF ∪ f
                end
                currentSemiFP ← currentSemiFP ∪ f
            end
        end
        tempPatterns ← currentFP ∪ currentSemiFP
    end
end
```

Algorithm 1. WDAGMeet mining algorithm.

One important observation on the WDAGMeet algorithm is that, when the patterns are expanded, it adds not only frequent nodes but also semi-frequent nodes. The reason is that, the expansion rules do not satisfy the anti-monotone property: A pattern *f* may not be frequent because of low average-weight of the nodes contained in it, but connecting some nodes (of heavier weight and high support) can make *f* frequent. To overcome this problem we have introduced the idea of semi-frequent WIFDAG patterns and kept a clear distinction between the semi-frequent and infrequent patterns. We have discarded the infrequent patterns from further expansion as an infrequent pattern can never be frequent (see Lemma 1). We have kept into consideration only the frequent and semi-frequent patterns for further expansion to generate new candidate patterns.

Another important observation is, while expanding a node *A*, with another node *B*, we can check if it is possible for *B* to occur after *A* (*A* triggers *B*) according to the *WIFDAG DB*. If this expansion is not possible, we can skip expanding *A* with *B*.

Lemma 1. *Any infrequent WIFDAG fragment in any step of the WDAGMeet algorithm can never result into a frequent WIFDAG fragment with applying expansion rules.*

Proof. By the definition of infrequent WIFDAG patterns in Definition 18, the conditions in Eqs. (8) and (9) must hold. For an infrequent WIFDAG pattern *f*, by Eq. (8), *f* is not frequent, as

W–Sup(f) < Min–WSup (without applying any expansion rule). Suppose we have added new P nodes, where $P+|f| \le N$ (N denotes an assumption about the maximum number of nodes in any frequent WIFDAG), to f using expansion rules and resulted into a new WIFDAG, f'. In best case, all of the newly inserted nodes can carry maximum weight and have normalized weight = 1. Moreover, in best case, after inserting P new nodes, f' can have support as much as f, i.e. Sup(f') ≤ Sup(f). This condition holds due to the anti-monotone property, that a fragment must have support less than or equal to any of its sub-fragment in general graph mining (unweighted). Now, we calculate the weighted support of f' as follows:

$$\text{W–Sup}(f') = \frac{\text{N–Weight}(f) \times |f| + P \times 1}{P+|f|} \times \text{Sup}(f) \quad (10)$$

From Eq. (9), we can say that

$$\frac{\text{N–Weight}(f) \times |f| + (N-|f|)}{N} \times \text{Sup}(f) < \text{Min–WSup}. \quad (11)$$

Since, $P+|f| \le N$, so, $P \le N-|f|$. So from Eq. (11) we can say that

$$\frac{\text{N–Weight}(f) \times |f| + P \times 1}{P+|f|} \times \text{Sup}(f) < \text{Min–WSup}. \quad (12)$$

From Eqs. (10) and (12),

$$\text{W–Sup}(f') < \text{Min–WSup}. \quad (13)$$

So, f' can never be frequent. □

4.3. Illustrative example

In this section, we will perform WDAGMeet mining algorithm on a sample meeting. Let us consider some snapshots of meeting sessions of Table 2 occurred in a faculty meeting. The weight of each interaction is assigned according to the rank of the person causing that interaction. In this example, the chairman is assigned a weight of 5, professor – 4, associate professor – 3 and lecturer – 1. So, MaxWeight = 5.

Seven persons participated in these meeting sessions; chairman, two professors, two associate professors and two lecturers. We will denote C for chairman, P_i for ith professor, AP_i for ith associate professor and L_i for ith lecturer. The corresponding WIFDAG representation of these sessions are shown in Fig. 11. The triggering relations are taken from the intention of interactions. Note that, this representation includes many indirect triggering relations, those were absent in tree based representation.

This representation also reflects the weights of each participant of the meeting. For mining frequent WIFDAG patterns from the meeting database in Fig. 11, we have defined minimum support = 60%. As Min-WSup must be less than minimum support, for mining significant patterns efficiently, we have multiplied the minimum support with average normalized weight of nodes (0.6) occurred in the meeting database to derive Min-WSup. So, calculation of Min-WSup (in percentage) is shown in the following equation:

$$\text{Min–WSup} = \text{Min–Sup} \times \text{AvgNormalizedWeight} = 60\% \times 0.6 = 36\% \quad (14)$$

Since number of graphs in the database = 3, so Min-WSup Count = 3 × 36% = 1.08. We have also assumed that maximum number of nodes in a frequent graph can be as much as 6. We have used label index of each label according to the order *PRO*, *ASK*, *POS*, *NEG*, *COM*, *ACC*, *REJ* for generating insertion signature, i.e. *PRO* has label index 7 and *REJ* has 1.

The first step of the mining process in WDAGMeet algorithm, i.e. mining frequent nodes, is shown in Fig. 13. Their support, normalized weight, minimum support and maximum possible W-Sup are also shown. We have compared the weighted support and maximum possible weighted support with Min-WSup and determined whether the node is frequent/semi-frequent/infrequent using the modules IsFrequent(), IsSemiFrequent() and IsInfrequent(). We can see that 6 nodes are frequent and 2 nodes are semi-frequent. As Lemma 1 shows, infrequent nodes can be safely discarded in further expansion rules.

Next, we proceed to apply expansion rules on the found single node WIFDAG patterns. We have taken the frequent pattern containing the node labelled *POS* having weight = 4. New root expansion and new level expansion of *POS*−4 are shown in Figs. 14 and 15, respectively. 4 temporary frequent patterns and 1 temporary semi-frequent pattern are found in the new root expansion, where 2 frequent patterns are found in new level expansion.

Similarly all other single nodes those are frequent or semi frequent, will be expanded using two expansion rules – new root and new level. Next, we have shown three expansion rules on a fragment consisting of two nodes, one as root (*POS*−4), and another node triggered by the root, residing in the second topological level (*POS*−3). The new root expansion rule is not applicable to this fragment. Other three rules, new level, new node and new edge expansion rules are applicable to the fragment. The result of all possible new node expansions is shown in Fig. 12. The result of applying new level expansion that finds a frequent pattern is shown in Fig. 16(a). The result of applying the new edge expansion on the frequent WIFDAG found in Fig. 16(a) is shown in Fig. 16(b) and another frequent pattern is found after this expansion. The step by step generation of the frequent WIFDAG found in Fig. 16(b) by applying various expansion rules is shown in Fig. 17. While generating new patterns, we have checked for duplicity using the modules CanonicalForm() and CanoncialDescription().

Fig. 18 shows how new WIFDAG patterns are generated by applying various expansion rules on the existing frequent/semi-frequent WIFDAG patterns. Note that, in Fig. 18, the last pattern is a frequent pattern, which was not mined in previously shown tree based meeting mining. Thus, it is clear that, mining in WDAGMeet algorithm results into some interesting frequent patterns that tree based meeting mining used to miss.

Table 2
Snapshots of three meeting sessions.

Session 1				Session 2				Session 3			
Time stamp	Desig.	Weight	Int. label	Time stamp	Desig.	Weight	Int. label	Time stamp	Desig.	Weight	Int. label
1	L_1	1	**PRO**	9	L_1	1	**PRO**	17	AP_1	3	**PRO**
2	P_1	4	**POS**	10	P_1	4	**POS**	18	AP_2	3	**POS**
3	P_2	4	**NEG**	11	AP_1	3	**POS**	19	L_2	1	**POS**
4	C	5	**ASK**	12	$P2$	4	**NEG**	20	L_1	1	**POS**
5	AP_2	3	**COM**	13	C	5	**ASK**	21	C	5	**NEG**
6	AP_1	3	**POS**	14	L_2	1	**NEG**	22	C	5	**ASK**
7	L_2	1	**NEG**	15	AP_2	3	**COM**	23	P_1	4	**NEG**
8	C	5	**ACC**	16	C	5	**ACC**	24	C	5	**REJ**

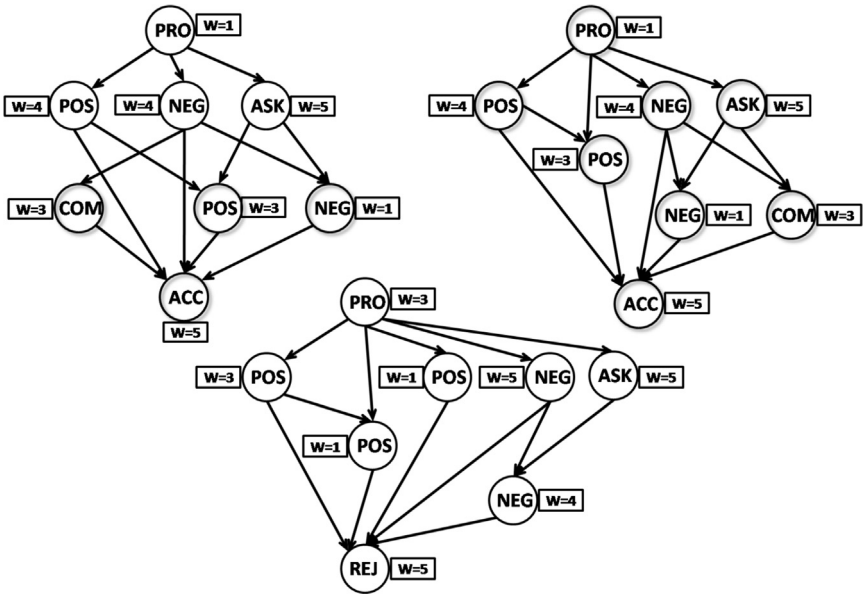


Fig. 11. WIFDAG based representation of meeting sessions of Table 2.

WIFDAG	Support	Normalized Weight	Weighted Support	Maximum Possible W-Sup	Frequent?	Semi-Frequent?	Infrequent ?
W=4 POS, W=3 POS, W=3 POS	0	0.67	0	0	No	No	Yes
W=4 POS, W=3 POS, W=1 NEG	0	0.73	0	0	No	No	Yes
W=4 POS, W=3 POS, W=1 NEG	0	0.53	0	0	No	No	Yes
W=4 POS, W=3 POS, W=3 COM	0	0.67	0	0	No	No	Yes
W=4 POS, W=3 POS, W=5 ACC	**2**	**0.8**	**1.6**	**0**	**Yes**	**No**	**No**

Fig. 12. New node expansion.

WIFDAG	Support	Normalized Weight	Weighted Support	Maximum Possible W-Sup	Frequent?	Semi-Frequent?	Infrequent ?
W=1 PRO	**2**	**0.2**	**0.4**	**1.73**	**No**	**Yes**	**No**
W=3 PRO	1	0.6	0.6	0.93	No	No	Yes
W=4 POS	**2**	**0.8**	**1.6**	**X**	**Yes**	**No**	**No**
W=3 POS	**3**	**0.6**	**1.8**	**X**	**Yes**	**No**	**No**
W=1 POS	1	0.2	0.2	0.87	No	No	Yes
W=5 NEG	1	1.0	1.0	1	No	No	Yes
W=4 NEG	**3**	**0.8**	**2.4**	**X**	**Yes**	**No**	**No**
W=1 NEG	**2**	**0.2**	**0.4**	**1.73**	**No**	**Yes**	**No**
W=5 ASK	**3**	**1.0**	**3.0**	**X**	**Yes**	**No**	**No**
W=3 COM	**2**	**0.6**	**1.2**	**X**	**Yes**	**No**	**No**
W=5 ACC	**2**	**1.0**	**2.0**	**X**	**Yes**	**No**	**No**
W=5 REJ	1	1.0	1.0	1	No	No	Yes

Fig. 13. Single node frequency table along with their support, normalized weight, weighted support, maximum possible W-Sup and whether they are frequent, semi-frequent or infrequent.

WIFDAG	Support	Normalized Weight	Weighted Support	Maximum Possible W-Sup	Frequent?	Semi-Frequent?	Infrequent ?
W=4 POS, W=4 POS	0	0.8	0	0	No	No	Yes
W=4 POS, W=3 POS	**2**	**0.7**	**1.4**	**X**	**Yes**	**No**	**No**
W=4 POS, W=4 NEG	**2**	**0.8**	**1.6**	**X**	**Yes**	**No**	**No**
W=4 POS, W=1 NEG	**2**	**0.5**	**1.0**	**1.67**	**No**	**Yes**	**No**
W=4 POS, W=3 COM	**2**	**0.7**	**1.4**	**X**	**Yes**	**No**	**No**
W=4 POS, W=5 ACC	**2**	**0.9**	**1.8**	**X**	**Yes**	**No**	**No**

Fig. 14. New root expansion on POS-4 node.

WIFDAG	Support	Normalized Weight	Weighted Support	Maximum Possible W-Sup	Frequent?	Semi-Frequent?	Infrequent ?
W=4 POS, W=1 PRO	0	0.5	0	0	No	No	Yes
W=4 POS, W=5 ASK	0	0.9	0	0	No	No	Yes
W=4 POS, W=4 POS	0	0.8	0	0	No	No	Yes
W=4 POS, W=3 POS	**2**	**0.7**	**1.4**	**X**	**Yes**	**No**	**No**
W=4 POS, W=4 NEG	0	0.8	0	0	No	No	Yes
W=4 POS, W=1 NEG	0	0.5	0	0	No	No	Yes
W=4 POS, W=3 COM	0	0.7	0	0	No	No	Yes
W=4 POS, W=5 ACC	**2**	**0.9**	**1.8**	**X**	**Yes**	**No**	**No**

Fig. 15. New level expansion on POS-4 node.

5. Evaluation results

To evaluate the performance of the proposed WDAGMeet mining algorithm, we have performed extensive experiments on synthetic datasets, semi-real dataset and real dataset. Each dataset contains a description of (i) the meeting sessions captured in WIFDAGs, (ii) labelled interactions with their corresponding weights, and (iii) triggering relations (i.e. directed edges of the WIFDAG). We have used a weight function to map each interaction to an integer value ranging from 1 to 5, inclusive. We have used nine interaction labels to label interactions occurred in meeting sessions. An ordered pair, denoting the interaction indices, was used to describe each directed edge in WIFDAG. The former interaction in the ordered pair is the triggering interaction and the later interaction is triggered interaction. Experiments were run using an Intel Core i5 2.50 GHz machine with 2.94 GB of RAM and 32 bit OS (Windows 7). The algorithm was implemented in C++. Used compiler and editor was Microsoft Visual C++ 6.0.

To demonstrate the real-life significances of our approach in different scenarios and to present a fair comparison of the existing tree-based meeting mining approach (Yu et al., 2012), two additional variations, called *WDAGMeet*$^{\#}$ and *WDAGMeet*$'$ have been shown. Discarding the weight feature of interactions in WDAGMeet results into *WDAGMeet*$^{\#}$ and *WDAGMeet*$'$ is another variation of WDAGMeet that is suitable for testing on weighted tree database. Note that, in *WDAGMeet*$'$, new root expansion and new edge expansion are excluded while patterns are expanded, because a tree can neither contain multiple root nor more than one edge connecting a node

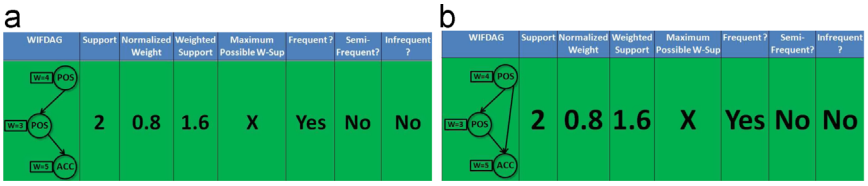


Fig. 16. Results of applying (a) new level expansion and (b) new edge expansion.

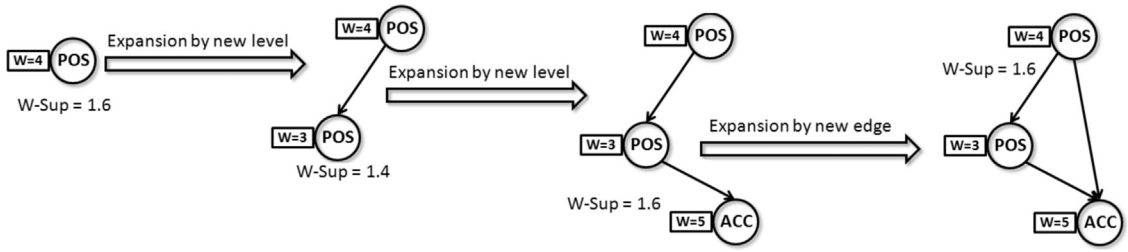


Fig. 17. Step-by-step WIFDAG generation by applying various expansion rules to POS-4 node.

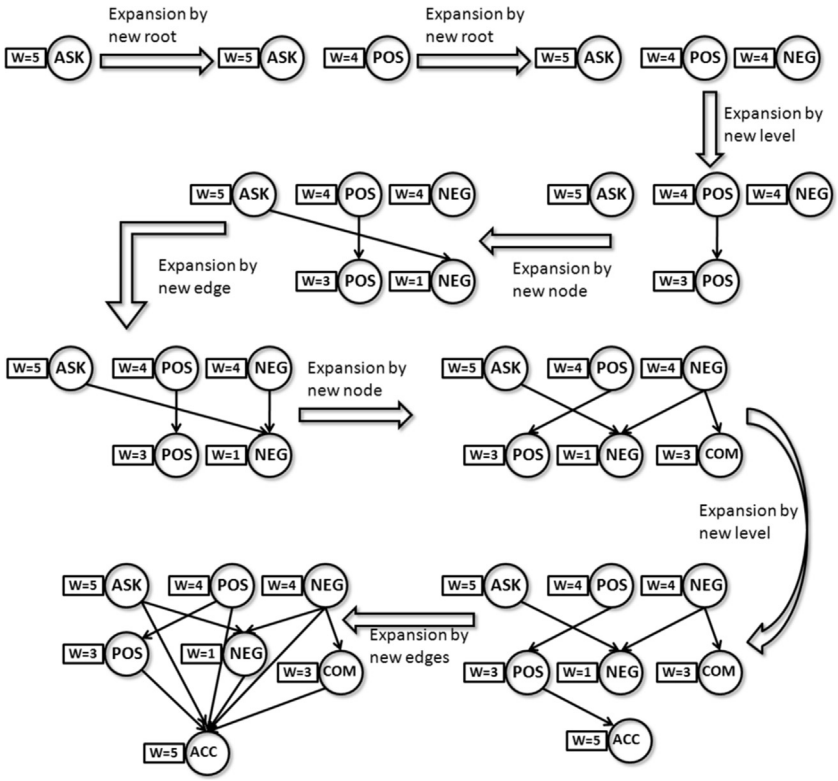


Fig. 18. Step-by-step WIFDAG generation by applying various expansion rules to ASK-5 node.

from nodes residing in earlier topological levels. We have converted the weighted DAG database to weighted tree database by removing all forward and cross edges.

5.1. Performance on synthetic datasets

We have generated synthetic datasets to simulate real meeting scenarios. Table 3 shows the number of frequent patterns mined from a synthetic meeting dataset, *WD10*, consisting of 10 meeting sessions, and elapsed time comparison between WDAGMeet and WDAGMeet#. *WD10* contains 142 vertices, 132 edges in tree based representation and 708 edges in WIFDAG based representation. Table 4 shows similar comparison for tree database representation of another synthetic meeting dataset, *WD37*, containing 37 meeting sessions, performed by WDAGMeet′ and tree based meeting mining method. *WD37* contains 1117 vertices, 1080 edges in tree based representation and 10 672 edges in WIFDAG based representation.

It is clear from Tables 3 and 4 that when node weights are taken into account, number of mined frequent patterns becomes lesser and those patterns are the more interesting patterns. Elapsed time is also lower for weight based version for both WIFDAG and tree databases. Also, observe that, as the number of frequent patterns increases the required time to discover these frequent patterns also increases.

5.2. Performance on a converted real-life dataset

Now we present experimental result based on converted real-life dataset. We have downloaded Hayes Roth dataset from UCI machine learning repository (Bache and Lichman, 2013). This dataset was adjusted by converting the attribute values to suit our required format and generate *HayesRoth**. The dataset contained education, class, hobby, age and marital status information. We have calculated weight of interaction = 1 + (education + class) mod 5, label of interaction = (hobby + age + marital status). Edges (interaction triggering relations) were randomly generated. We have segmented the dataset to produce 14 different meeting sessions. Average number of interaction per session was 9.07. The dataset contains 127 vertices, 113 edges in tree based representation and 294 edges in WIFDAG based representation. Table 5 shows comparison between WDAGMeet and WDAGMeet# based on number of mined frequent patterns and Fig. 19 reflects the required time to mine patterns for both the algorithms. It can be easily observed that WDAGMeet mines lesser number of patterns in much less time than WDAGMeet#.

Table 3
Comparison between WDAGMeet and WDAGMeet# on *WD10* dataset.

Min-Sup (%)	Number of frequent patterns		Elapsed time (s)	
	WDAGMeet	WDAGMeet#	WDAGMeet	WDAGMeet#
90	0	3	0.352	0.762
85	0	3	0.328	0.801
80	1	5	0.973	2.283
75	1	5	0.721	2.314
70	1	6	0.729	4.628
65	2	6	1.304	4.146
60	2	12	2.790	8.414
55	2	12	3.000	8.727
50	7	24	3.524	23.616
45	7	24	7.236	23.425
40	12	63	8.074	128.388
35	12	63	8.136	128.729
30	19	579	111.131	1881.267

Table 4
Comparison between WDAGMeet′ and tree based method on *WD37* dataset.

Min-Sup (%)	Number of frequent patterns		Elapsed time (s)	
	WDAGMeet′	Tree based	WDAGMeet′	Tree based
90	7	9	21.025	46.594
85	7	12	21.325	50.669
80	9	13	21.594	117.591
75	9	13	23.245	118.883
70	9	15	26.511	235.456
65	9	16	30.731	303.972
60	10	19	31.166	413.089
55	11	27	32.314	744.599
50	12	36	32.824	1597.951

5.3. Performance on a real-life meeting dataset

This dataset is collected from academic meeting sessions held in the Department of Computer Science and Engineering, University of Dhaka and is named *CSEDU5* dataset.

All faculty members of the department participated in this meeting for taking a decision regarding the MS thesis marking strategy. We have included 5 meeting sessions and on an average, each session contains 12 interactions. The dataset contains 60 vertices, 55 edges in tree based representation and 91 edges in WIFDAG based representation. Figs. 20 and 21 represent performance comparison between WDAGMeet and WDAGMeet# with respect to number of mined patterns and elapsed time for various minimum supports for the real dataset respectively.

Figs. 22 and 23 represent performance comparison between WDAGMeet′ and tree based method with respect to number of mined patterns and elapsed time for various minimum supports for the real dataset respectively. It is clear from the figures that algorithms that consider weights of interactions outperform the algorithms that discard interaction weights in terms of both number of mined patterns and elapsed time for both DAG and tree databases.

5.4. Comparison between WDAGMeet and the existing tree-based meeting mining method (Yu et al., 2012)

We have already shown some comparisons between WDAGMeet and the existing tree-based meeting mining method in the previous subsections. In this subsection, we thoroughly analyse their performances; discover the reasons and show a direction to select a particular variation of our approach by explaining when it is more useful compared to the existing approach. There are a number of advantages of WDAGMeet over the existing tree based meeting mining.

First, the tree based method misses some important frequent patterns because it does not capture all triggering relations. As illustrated in Fig. 24, only one triggering relation is captured in the tree database for each triggered interaction. For instance, the tree captures the interaction ASK triggered by PRO but misses the one triggered by POS. As such, the tree based method does not generate the pattern POS–ASK–NEG as these three nodes are not directly connected in the tree. In fact, fragments containing siblings or ancestor's siblings of a node might not be connected without the presence of their common ancestor in a tree. Hence, if the common ancestor is not frequent, the tree mining method fails to mine such fragments as a frequent pattern. In contrast, being internally connected with partial triggering relations, WDAGMeet discovers this kind of frequent interaction patterns such as POS–ASK–NEG in the above example. This kind of frequent patterns reveals highly correlated interactions. However, if all the frequent DAG patterns are

Table 5
Comparison between WDAGMeet and WDAGMeet# with respect to number of mined frequent patterns on *HayesRoth** dataset.

Min-Sup (%)	Number of frequent patterns	
	WDAGMeet	WDAGMeet#
90	0	2
85	0	3
80	1	3
75	1	3
70	1	6
65	1	6
60	3	7
55	3	11
50	4	16
45	5	16
40	6	32
35	9	111

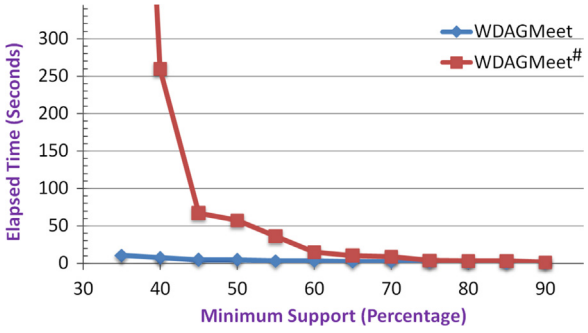


Fig. 19. Comparison between WDAGMeet and WDAGMeet# with respect to elapsed time on *HayesRoth** dataset.

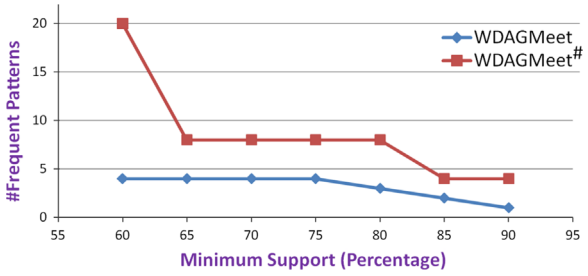


Fig. 20. Comparison between WDAGMeet and WDAGMeet# with respect to number of frequent patterns on *CSEDU5* dataset.

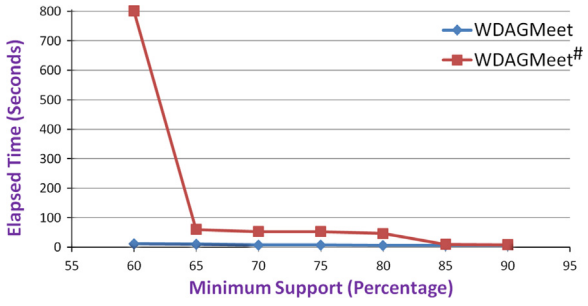


Fig. 21. Comparison between WDAGMeet and WDAGMeet# with respect to elapsed time on *CSEDU5* dataset.

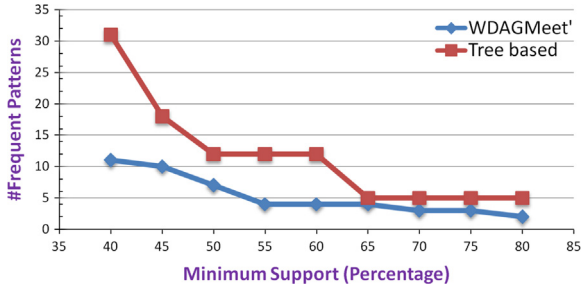


Fig. 22. Comparison between WDAGMeet′ and tree based mining with respect to number of frequent patterns on *CSEDU5* dataset.

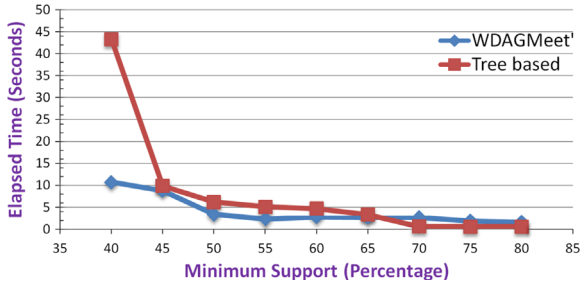


Fig. 23. Comparison between WDAGMeet′ and tree based mining with respect to elapsed time on *CSEDU5* dataset.

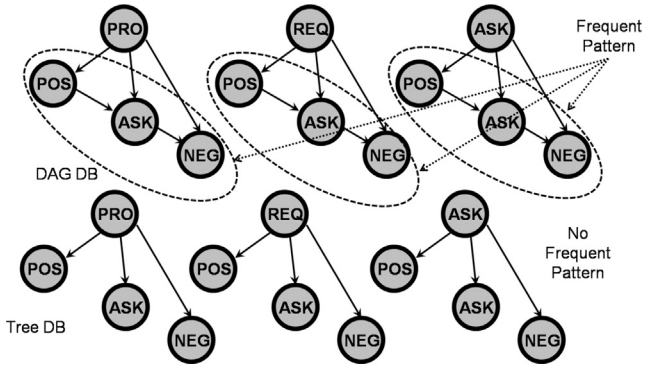


Fig. 24. DAG based vs. tree based representations of meetings.

needed by some applications without regarding the weight, WDAGMeet# can be used as the best solution.

Second, tree based mining gives equal weights to all interactions and does not distinguish two same interactions performed by persons lying in different ranks. As WDAGMeet associates weight to each of the interactions according to the person who initiates that interaction, the problem of ignoring person's rank in a meeting is solved as shown in Example 5. However, weighted tree based version of WDAGMeet, WDAGMeet′, gives the best

performance when database has weight information and interactions are presented in tree format.

Third, tree based mining misses some frequent patterns, as discussed earlier, but it also generate some less interesting frequent patterns. As weights of all nodes are equal in a tree, some interactions, frequently occurred by a least significant person is counted as frequent pattern. But the main goal of discovering frequent pattern is to discover interesting patterns. Although WDAGMeet generates fewer frequent patterns, because it distinguishes same interaction by their different ranked initiators, it does not miss any interesting frequent pattern.

Now we present the comparison among WDAGMeet, WDAGMeet#, WDAGMeet′ and existing tree based meeting mining method (Yu et al., 2012) with respect to number of frequent patterns for various minimum supports on the *CSEDU5* real-life meeting dataset, discussed in Section 5.3, in Fig. 25. It is clear that by integrating the feature of considering weighted nodes in trees, WDAGMeet′ generates fewer patterns than tree based method. But as stated earlier, it misses some correlated patterns. On the other hand, by mining the same meetings by representing them using unweighted DAG database, WDAGMeet# generates much more patterns than tree based method. WDAGMeet# does not miss any correlated pattern, but it does not take into account participant ranks. As a result, it generates some uninteresting patterns.

WDAGMeet contains both the positive factors of WDAGMeet′ and WDAGMeet#. It is reflected in the figure that WDAGMeet generates more patterns than WDAGMeet′ but less patterns than both WDAGMeet# and tree based method. Because WDAGMeet generates all correlated patterns by taking different ranks of persons into account, hence, minimizes the total number of frequent patterns by eliminating uninteresting patterns. In contrast, the tree-based method generates more frequent patterns by discarding the weights of participants. Moreover, it misses some frequent correlated patterns as it does not consider all the triggering interactions. For the sake of comparison, we have converted the minimum support applicable for unweighted database to Min-WSup of WDAGMeet algorithm by performing necessary calculations.

5.5. Scalability test of WDAGMeet

In this section, we present the test results for scale up properties of WDAGMeet algorithm. To test the scalability of WDAGMeet algorithm, we have analysed the algorithm's performance on 10 different datasets.

Size of the datasets varies from 5 KB to 427 KB. Min-WSup varies from 60% to 80%. Number of meeting sessions of these datasets varies from 10 to 1000. Fig. 26 shows the performance of WDAGMeet algorithm on these datasets. Here, *X*-axis denotes number of meeting sessions in a database and *Y*-axis denotes elapsed time in seconds to find WIFDAG patterns using WDAGMeet algorithm. 3 different curves are shown to reflect elapsed time on different sized datasets and different values of Min-WSup. It is clear from the figure that WDAGMeet algorithm takes more time on large datasets and as the Min-WSup decreases, elapsed time also increases. However, from the experimentation, it is clear that WDAGMeet algorithm shows scalability with increasing database size and decreasing Min-WSup values.

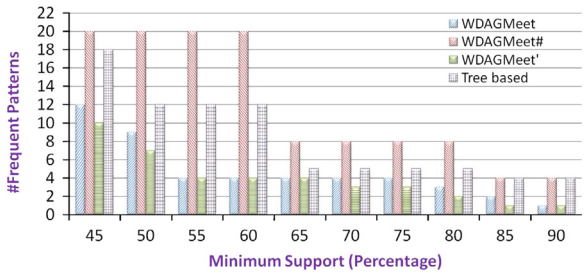


Fig. 25. Comparison among WDAGMeet, WDAGMeet#, WDAGMeet′ and tree based method with respect to number of frequent patterns on *CSEDU5* dataset.

6. Discussion

If we observe the interaction flow patterns of meetings and can mine the relation among meeting contents, decisions taken and the outcome of that decision, it is possible to predict the outcomes of future meetings. We can also predict the correctness of the decisions made in meetings. Some real life applications of WDAGMeet, i.e. meeting mining in the field of meetings are given below:

1. Mined patterns from meetings can serve as features in the classification process of meetings. We can classify a specific meeting as productive/non-productive, successful/unsuccessful according to the classification algorithm trained by mined frequent patterns.
2. Meeting mining can contribute in interpreting semantic knowledge from meeting contents.
3. Mined patterns can be used in discovering the relations among participants. For example, if there exists two groups in an organization, who always oppose one another, it can be easily found out by association rule/correlation mining. Mined patterns can also be used to predict participant behaviours in future meetings.
4. Effectiveness of discussion taken place in meetings and correctness of decisions taken in meetings can also be verified through patterns and rules mined from previous meetings.
5. Nowadays, there is a huge volume of data. All meeting records can be stored, but retrieving useful information from previously occurred meetings becomes almost impossible without any proper indexing. The mined frequent patterns can be used as indexing features for future ease of access to the meeting database.

Many business organizations, medical institutions and other organizations can be greatly benefited from meeting mining in order to verify decisions taken in meetings, calculating future probabilities of a certain event to occur and classify meetings as useful or useless.

Another important application of meeting mining framework can be in the field of traffic control system in highly congested roads in big cities. Traffic polices of each road crossing usually communicate over wireless walkie-talkies. They also deliver information regarding the load of traffic at that specific crossing at a specific time. If we represent those communication instances with WIFDAG nodes as described in the proposed framework in this paper and assign weights to the nodes according to the load of the traffic in that

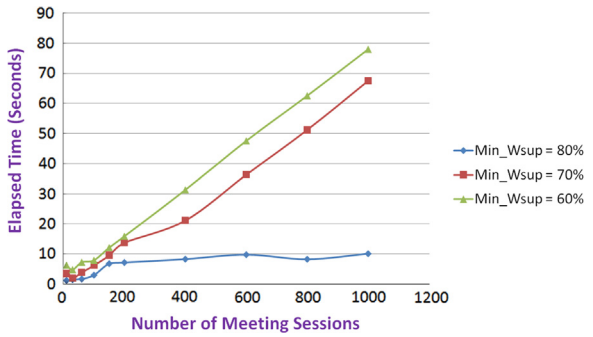


Fig. 26. Scalability test of WDAGMeet algorithm on different datasets of varying sizes and varying Min-WSup.

crossing, we can generate similar weighted DAG based database as meeting WIFDAG databases. By mining frequent traffic patterns from the generated database, we can predict traffic load at certain crossing at certain time and also adjust the traffic control system according to the prediction. In Yun (2007), weighted frequent/interesting patterns have been used to facilitate the traffic control system using transactional databases. But as discussed above, traffic control system communication can be better represented using WIFDAGs than WFPs/WIPs. Therefore, the proposed framework is better than the existing framework (Yun, 2007) to mine more significant knowledge from traffic patterns.

Nowadays social network chats like group chats in Facebook, Google hangout and e-mail conversations are often group conversations. Moreover, twitter provides an option of re-twitting the status of a person. This sort of triggering and triggered interactions produce WIFDAGs. From these group conversations, we can generate WIFDAG patterns in a quite similar fashion to the live meetings. Supervised learning, by considering the mined frequent interaction patterns from those social group conversations as features, can reveal useful information regarding the relation among the participants as well as predicting the importance or usefulness of that group conversation. Moreover, we can also apply unsupervised learning on those group conversations to cluster same type of group conversations in one cluster. This type of clustering often helps to identify cyber criminal groups.

Like social networking, telecommunication patterns and communication frequency between parties can also be represented using the proposed framework. We can assign weights to the WIFDAG nodes according to the communication frequency and duration of communication. Classifying those telecommunication history through frequent interaction pattern mining can help in group behaviour, interesting person and fraud detection.

Bidding in stock market can be considered as interaction. From the bidding information (amount of shares bidden for or value of each share), we can assign weights to each bidding interaction (node in WIFDAG). We can produce WIFDAG database from bidding information generated by various parties and also incorporate triggering and triggered relations among biddings. The proposed framework can represent the problem of stock market. Cao (2010), together with his collaborators (Cao et al., 2012), analysed couple behaviour in stock market using behaviour sequences, but the proposed framework can also represent triggering/triggered relation among the sellers and/or buyers and thus preserve more information in stock market scenario. Mining frequent bidding patterns in stock market can result into many useful business predictions like increment or decrement of value of a certain share or detecting fraud activities.

7. Conclusions

In this paper, we have proposed a new framework to model human interactions in meetings using weighted interaction flow directed acyclic graph, WIFDAG, that can preserve temporal and triggering relations among interactions and participant rank in meetings. Moreover, we have also proposed WIFDAG based frequent interaction pattern mining algorithm, WDAGMeet, that makes use of expansion rules to enumerate all frequent interaction flow patterns in a bottom up manner. Evaluation results show that different variations of our WDAGMeet algorithm can be used more successfully in different real-life scenarios and as a consequence, our approach is significantly better than the existing approach for mining meeting databases. For future extension of WDAGMeet algorithm and the proposed framework, the mined frequent sub-WIFDAGs can be served as foundations to further association rule and correlation mining to predict future effects of meetings. Moreover, integration of other types of human interactions can be done so that resulting mining algorithm will be able to handle other classes of meetings such as medical interviews and business discussions. Integration of dynamic weight assignment by designing a customizable shell based on this framework will make it even more robust and worthwhile.

References

Ahmed, C.F., Tanbeer, S.K., Jeong, B.S., Lee, Y.K., 2009. Efficient tree structures for high utility pattern mining in incremental databases. IEEE Trans. Knowl. Data Eng. 21 (12), 1708–1721.

Ahmed, C.F., Tanbeer, S.K., Jeong, B.S., Lee, Y.K., Choi, H.J., 2012. Single-pass incremental and interactive mining for weighted frequent patterns. Expert Syst. Appl. 39 (9), 7976–7994.

Alavi, F., Hashemi, S., 2015. DFP-SEPSF: a dynamic frequent pattern tree to mine strong emerging patterns in streamwise features. Eng. Appl. Artif. Intell. 37, 54–70.

Bache, K., Lichman, M., 2013. UCI Machine Learning Repository. ⟨http://archive.ics.uci.edu/ml/⟩.

Campagna, A., Pagh, R., 2010. On finding frequent patterns in event sequences. In: ICDM, pp. 755–760.

Cao, L., 2010. In-depth behavior understanding and use: the behavior informatics approach. Inf. Sci. 180 (17), 3067–3085.

Cao, L., Ou, Y., Yu, P.S., 2012. Coupled behavior analysis with applications. IEEE Trans. Knowl. Data Eng. 24 (8), 1378–1392.

Chen, Y.-L., Kao, H.-P., Ko, M.-T., 2004. Mining DAG patterns from DAG databases. In: WAIM, pp. 579–588.

Duong, H.V., Truong, T.C., Vo, B., 2014. An efficient method for mining frequent itemsets with double constraints. Eng. Appl. Artif. Intell. 27, 148–154.

Fariha, A., Ahmed, C.F., Leung, C.K.-S., Abdullah, S.M., Cao, L., 2013. Mining frequent patterns from human interactions in meetings using directed acyclic graphs. In PAKDD, pp. 38–49.

Geng, R., Xu, W., Dong, X., 2009. Efficient mining of interesting weighted patterns from directed graph traversals. Integr. Comput.-Aided Eng. 16 (1), 21–49.

Han, J., Pei, J., Yin, Y., Mao, R., 2004. Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min. Knowl. Discov. 8 (1), 53–87.

Kipp, M., 2001. Anvil—a generic annotation tool for multimodal dialogue. In: Eurospeech, pp. 1367–1370.

Kolár, J., Liu, Y., Shriberg, E., 2010. Speaker adaptation of language and prosodic models for automatic dialog act segmentation of speech. Speech Commun. 52 (3), 236–245.

Lee, G., Yun, U., 2012. Mining weighted frequent sub-graphs with weight and support affinities. In: MIWAI, pp. 224–235.

Magnusson, M.S., 2000. Discovering hidden time patterns in behavior: T-patterns and their detection. Beh. Res. Meth. Instr. Comp. 32 (1), 93-110.

Morita, T., Hirano, Y., Sumi, Y., Kajita, S., Mase, K., 2005. A pattern mining method for interpretation of interaction. In: ICMI, pp. 267–273.

Nijholt, A., Rienks, R., Zwiers, J., Reidsma, D., 2006. Online and off-line visualization of meeting information and meeting support. Vis. Comput. 22 (12), 965–976.

Otsuka, K., Sawada, H., Yamato, J., 2007. Automatic inference of cross-modal nonverbal interactions in multiparty conversations: “who responds to whom, when, and how?” from gaze, head gestures, and utterances. In: ICMI, pp. 255–262.

Termier, A., Tamada, Y., Numata, K., Imoto, S., Washio, T., Higuchi, T., 2007. Digdag, a first algorithm to mine closed frequent embedded sub-dags. In: MLG.

Tomobe, H., Nagao, K., 2006. Discussion ontology: knowledge discovery from human activities in meetings. In: JSAI, pp. 33–41.

Waibel, A., Bett, M., Finke, M., Stiefelhagen, R., 1998. Meeting browser: tracking and summarizing meetings. In: DARPA Broadcast News Transcription and Understanding Workshop.

Werth, T., Dreweke, A., Wörlein, M., Fischer, I., Philippsen, M., 2008. Dagma: Mining directed acyclic graphs. In: IADIS ECDM, pp. 11–18.

Yang, K., Hong, T., Lan, G., Chen, Y., 2014. A two-phase approach for mining weighted partial periodic patterns. Eng. Appl. Artif. Intell. 30, 225–234.

Yu, Z., Yu, Z., Ko, Y., Zhou, X., Nakamura, Y., 2009. Inferring human interactions in meetings: a multimodal approach. In: UIC, pp. 14–24.

Yu, Z., Yu, Z., Zhou, X., Becker, C., Nakamura, Y., 2012. Tree-based mining for discovering patterns of human interaction in meetings. IEEE Trans. Knowl. Data Eng. 24 (4), 759–768.

Yun, U., 2007. Efficient mining of weighted interesting patterns with a strong weight and/or support affinity. Inf. Sci. 177 (17), 3477–3499.

Yun, U., Shin, H., Ryu, K.H., Yoon, E., 2012. An efficient mining algorithm for maximal weighted frequent patterns in transactional databases. Knowl.-Based Syst. 33, 53–64.

Yun, U., Lee, G., Ryu, K.H., 2014. Mining maximal frequent patterns by considering weight conditions over data streams. Knowl.-Based Syst. 55, 49–65.

Zhang, D., Gatica-Perez, D., Bengio, S., McCowan, I., 2006. Modeling individual and group actions in meetings with layered hmms. IEEE Trans. Multimed. 8 (3), 509–520.