Int. J. Mach. Learn. & Cyber.
DOI 10.1007/s13042-012-0113-9

ORIGINAL ARTICLE

Bisociation networks analysis for business process

Hongmei He · Zengchang Qin

Received: 5 May 2011 / Accepted: 7 June 2012

Abstract Bisociation Network (BisoNet) is a novel approach for creative information discovery, and it can be projected to many real application domains. Bisociation of business processes onto a network is one of such applications. In this paper, we investigate business processes on the BisoNet, and develop a directed graph model to map the relations between business process flows. Based on the BisoNet model, we analyze the real-world data provided by a call service centre. The network-based statistics show that some special process steps could be key steps that greatly affect the performance of the service, and could result in a case unsolved. The network is simplified through constructing the network with shortest path of each process flow, and the simplified network may represent an optimal process pattern. This may provide a reference to the business organization for improving the quality of their service.

Keywords Bisociation networks · Business process · Directed graph model

1 Introduction

The concept of association is one of the key issues in information retrieval and data mining, which employ associations by similarity or co-occurrence to discover relevant information that is not related in obvious associative ways, in particular, information that is related across different contexts. This kind of context-crossing association are is needed in real-world practice and other innovative domains. The conventional association-based approach is to use measures of similarity or co-occurrence to calculate association [1, 2]. The innovational approach should be able to help users to uncover, select, re-shuffle, combine and synthesize already existing facts, ideas, faculties and skills into a new whole with novel features and properties [3].

The term, bisociation was coined by Koestler [3] to distinguish the type of metaphoric thinking that leads to the acts of great creativity from familiar associative thinking. Bisociation means to join unrelated, often conflicting information in a new way. In such a new way, we may discover certain undetectable information. The models of thinking allow the mixing of conceptual categories and contexts that are normally separated. The functional basis for these models is a mechanism that is named as bisociation. Bisociation refers to the mixture in one human mind of concepts from two contexts of categories of objects that are normally considered separate by the literate processes of the mind. The overall aim of bisociation is to develop and validate a computational methodology, which facilitates bisociative information discovery. Berthold et al. [4] outlined the approach for network-based information access and exploration, navigation methods such as BisoNet can facilitate the discovery of links across diverse domains.

Bisociative relationships can only be discovered on the basis of a sufficiently large and sufficiently diverse (different contexts) underlying corpus of information. A graphical theoretic approach [5, 6] can provide the best framework to accommodate both dimensions of complexity—type diversity as well as volume size. Besides the

H. He
Intelligent System Lab, University of Bristol,
Bristol BS8 1UB, UK
e-mail: h.he@bristol.ac.uk

Z. Qin (✉)
Intelligent Computing and Machine Learning Lab,
School of Automation Science and Electrical Engineering,
Beihang University, Beijing, China
e-mail: zengchang.qin@gmail.com; zcqin@buaa.edu.cn

Published online: 29 June 2012

fundamental graph processing methods, interactive methods to aggregate, project, query and visualize (sub) structures of very large graphs (e.g., [7–9]) have been proposed based on the current usage context. Especially, the visualization element is of crucial importance, so that the user can effectively explore and navigate promising relations in order to allow for truly bisociative information discovery. This requires the presentation of (partial) graph structures together with content information of the linked items [10, 11].

Bisociation Network (BisoNet) can be used in many real application domains. For example, Kötter et al. [12] used BisoNets to assist the discovery of unexpected associations across different domains; Mozetič et al. [13] presented SEGS+Biomine, a bisociation discovery system for exploratory gene analytics. Bisociation of business processes is one of application scenarios that haven't been well studied for the reason that it is not easy to manipulate a large size of dataset. Also the large dataset usually looks disorderly and unsystematic. Bisociation network presents a good way to deal with such a dataset. The key is to find the mapping between business domain and the network, so that we can analyze business processes based on the network. In this paper, we investigate the bisociation of business processes on a network, and develop a directed graph to map the relations between business process flows. Based on the BisoNet model, we analyze the real-world data provided by a call service centre. From the BisoNet representation of the real-world data, we try to find some potentials to improve the service. NisoNet algorithms to assist process analysis are developed and implemented as functional nodes in KNIME [14], which is a user-friendly and comprehensive open-source platform for data integration, processing, analysis, and exploration.

2 Bisociation of two domains

Two orthogonal planes can be used to represent related crossing domains. Figure 1 depicts the bisociation of two different domains. M_1 is the process of thought governed by habitual rules of the game, e.g. the approaches of business process analysis. M_2 is the matrix of associations related to the other domain (e.g network analysis). m_2 represents the actual train of thought which effects the connection.

"Business process" can be defined as "a chain of logical connected, repetitive activities that utilizes the organization's resources to refine an object for the purpose of achieving specified and measurable results or products for internal or external customers" [15]. Process analysis is an approach that helps managers improves the performance of their business activities. It can be a milestone in continuous improvement [16]. Usually, process analysis includes 6 tasks [17]:

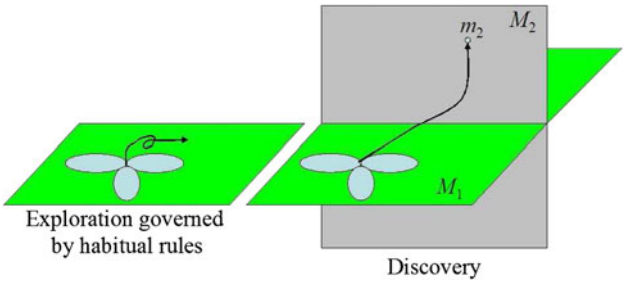

Fig. 1 The Bisociation

- Define the process boundaries that mark the entry points of process inputs and the exit points of process outputs.
- Construct a process flow diagram that illustrates various process activities and their relationships.
- Determine capacities of each step in process, and calculate other measures of interest.
- Identify the bottleneck, that is, the step has the smallest capacity.
- Evaluate further limitations in order to quantify the impact of the bottleneck.
- Use the analysis to make operating decision and improve the process.

A network can be used to express various relationships between objects in the real world. How to implement the tasks under the framework of BisoNet? Namely, we need to find the point m_2 in Fig. 1 to represents the actual train of thought which effects the connection. For the given business case set, the boundaries could be case status, "Open", or "Closed". "Open" means a case is not solved, and "closed" means the case is solved. Every case has a starting point and ending point. Obviously, a network can be used to represent the relationship between process flows. The bisociation of business processes to a network is to create a mapping between business processes and the network. Namely, the network should represent the relations between process flows. Both nodes and edges contain references to their underlying content, explaining their semantics and pointing to their origin.

Imagine that in a pub or restaurant, waiters present service to customers. For example, waitress *A* asks "Would you like to have any drink?"; Waitress *B* asks "Would you like to have a cup of tea or coffee?" Who will get more orders? Both questions could have two answers: for *A*'s question, yes or no; for *B*'s question, tea or coffee. Obviously, *B* will win more orders, as most people will follow the question. This has been proved by experiments. Hence, for the business data from a call centre, each question can be viewed a process step. The quality of a question could affect the performance of a service. The sequence of questions in each case forms a process flow. Therefore, a

bisociation network can directly illustrate the relationships between question sequences for all cases, where the nodes represent the questions and edges represent the orders of questions.

3 Directed graph model to map the relations between process flows

In order to clearly show the relations between process flows, we create a set of test data with 8 process flows, each of which is comprised of a sequence of questions. Some questions appear in different flows. For example, q_{22} and q_{23} appear in both case 2 and case 3; case 6 and case 7 have the same sequence of the first 5 questions.

1. q_{00}, q_{01}, q_{02}, q_{23}, q_{24}, q_{25}, q_{06}, q_{07}, q_{08}, q_{29}, q_{0a}, q_{0b}, q_{0c}, q_{2d}, q_{2e};
2. q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{25}, q_{26}, q_{17}, q_{18}, q_{19}, q_{1a};
3. q_{20}, q_{21}, q_{22}, q_{23}, q_{24}, q_{25}, q_{26}, q_{27}, q_{28}, q_{29}, q_{2a}, q_{2b}, q_{2c}, q_{2d}, q_{2e};
4. q_{30}, q_{31}, q_{22}, q_{23}, q_{34}, q_{35}, q_{36}, q_{37}, q_{38}, q_{29}, q_{2a};
5. q_{00}, q_{01}, q_{02}, q_{43}, q_{44}, q_{45}, q_{46}, q_{37}, q_{48}, q_{49}, q_{4a}, q_{4b}, q_{4c}, q_{4d}, q_{4e};
6. q_{50}, q_{51}, q_{52}, q_{53}, q_{54}, q_{45}, q_{46}, q_{37}, q_{48}, q_{49}, q_{4a}, q_{4b}, q_{4c};
7. q_{50}, q_{51}, q_{52}, q_{53}, q_{54}, q_{65}, q_{66}, q_{67}, q_{68}, q_{69}, q_{6a}, q_{6b}, q_{6c}, q_{6d}, q_{6e};
8. q_{70}, q_{71}, q_{72}, q_{73}, q_{74}, q_{50}, q_{51}, q_{52}, q_{70}, q_{79}, q_{7a}, q_{7b}.

A directed graph model is developed to map the relations between process flows. In a directed graph $G(V, E)$, V is the vertex (or node) set and E is the edge set. Any vertex v in the graph represents a question q in process flow, and any edge $e(v_i, v_j)$ represents the order relation of two questions q_i and q_j in the graph G. Namely, q_j follows q_i. According to the model, we can create a directed graph for a set of process flows. Figure 2 depicts the relations of process flows in the test data. We can see there some cycles in the green process flow. From the process step 67, go through several vertices, then back to vertex 67, and then go to vertex 72. Namely, vertices, 67, 68, 69, 70, 71, 52, 53, 54, 67 form a cycle. Can such cycles be avoided in real-world process flow through improving quality of service?

We can easily know how many different questions to be asked in the data set, and each question uniquely corresponds to a node in the graph. Therefore, given a data set, we can easily know the size of the graph that is constructed with the data set, and it is easy to implement the construction algorithm (see Algorithm 1). We hold an adjacent list of all nodes in this graph, each node holds two lists, *outDegreeList* and *inDegreeList*. We scan all cases, when q_j follows q_i in a process flow, node v_j is added into the *outDegreeLsit* of node v_i, and node v_i is added into the *inDegreeList* of node v_j. Most importantly, with this kind of directed graph, it is convenient for the calculation of two graphs, such as common edges, common nodes, different edges, different nodes, if we unify the indices of questions in the whole data set.

There are more than eight paths from the start node to the terminal node in the graph shown in Fig. 2. Namely, there exist some paths that do not exist in the real-world process flows. For example, the path of $\{q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{25}, q_{06}, q_{07}, q_{08}, q_{29}, q_{2a}\}$ appears in the graph, but this case does not exist in the real data. Therefore, additional information may be needed to associate to the process flows for differing the process flows. In Fig. 2, we use different colors or dashed line to indicate different process flows. However, on the other hand, this might present a potential for process improvement. In the implementation of the graph, each node can be associated with a list of case IDs. As each case has a unique ID, therefore, we can traverse the graph to get all cases.

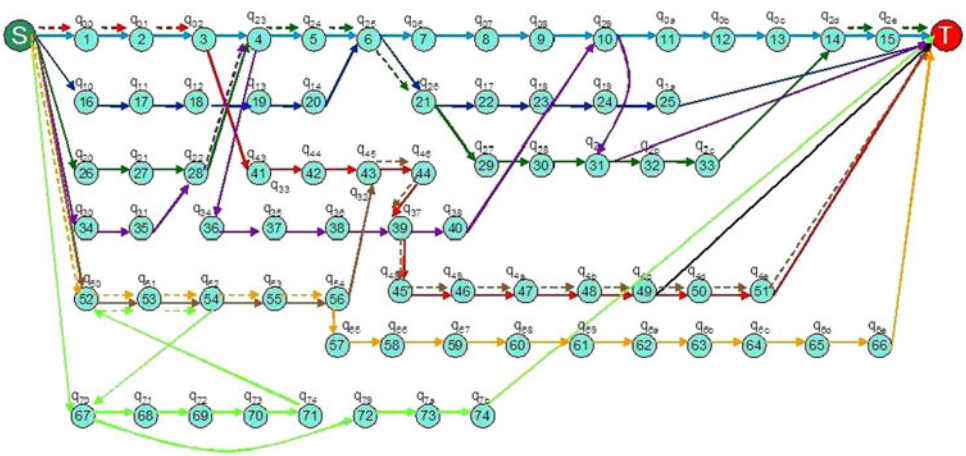

Fig. 2 The directed graph for the test data

Algorithm 1 CreateDG(D, S)

```
1: V = createVSet(START, S, STOP);
2: for (each process flow f_i in D) do
3:    t = START;
4:    for (each step p_j) do
5:       h = getIndex(p_j, S)+1;
6:       if (h ∉ V.t.outDegreeList) then
7:          insert(h, V.t.outDegreeList);
8:          insert(t, V.h.inDegreeList);
9:          t = h;
10:         j = j + 1;
11:      end if
12:   end for
13:   if (STOP ∉ outDegreeList(p_final)) then
14:      insert(STOP, V.t.outDegreeList);
15:      insert(t, V.STOP.inDegreeList);
16:   end if
17:   i = i + 1;
18: end for
```

Algorithm 1 CreateDG(D, S)

4 Computing based on the BisoNet

4.1 The shortest path

Here, the shortest path is a path from the start node to the terminate node in the graph that represents a single process flow. In real-world process data, there are a lot of similar questions. In the other words, questions may be repeated in different styles in a process flow. Fuzzy grammar technology provide a good means of extracting patterns of those similar questions. Martin et. al [18] proposed a fuzzy grammar-based approach combining with incremental learning to extract structured data from sections of unstructured text. We use this fuzzy pattern recognition technology [19, 20] to produce fuzzy-tagged question patterns, and use them to replace the practical questions. For example, we have the following questions:

q_1: "What does the customer need help with?"
q_2: "What is the customers problem?"
q_3: "What problem is customer experiencing with on-demand?"
q_4: "Which part of the wizard does the customer need help with?"
q_5: "What issue is the customer experiencing?"

All of these questions (e.g., q_1, q_2, q_3, q_4, q_5), which have similar semantics, are expressed as one tag (e.g., "<g1FindCustomerProblem />"). In the graph, all of these questions are mapped to the node that represents their tag. Hence, these nodes represent multiple questions make the process flows in the graph linked each other, as these questions could appear in different cases. There even could be some repeated question tags in one process flow. Repeated questions in a process flow form cycles in the unit graph that is comprised of the nodes in the process flow. On the other hand, this might indicate that we could change the question style to reduce the redundant process steps. Therefore, we developed a heuristic algorithm to get the shortest path sp (the possible shortest process flow).

For each process flow, which is a sequence of fuzzy tagged question patterns, the first step is usually a question pattern that is to find customer problem. Such kind of questions is the key of solving a problem. Therefore, we set a pointer to scan the process flow after the start node. For each step, we try to find the repeated tags in the remaining sequence after the step. Record the final repeated tag, and the pointer is set to the next step of the final repeated tag. If there is no repeated tag, then record the current tag, and the pointer is set to the next step of the current step. This procedure is repeated until the point is at the terminate node. Algorithm 2 presents the pseudo code for getting the shortest path for each process flow.

Algorithm 2 ShortestPath(D)

```
1: for (each process flow f_i ∈ D) do
2:    initialize(sp_i);
3:    j=0;
4:    while (j < sizeOf(f_i)) do
5:       iter = j;
6:       for (k = j + 1 to sizeOf(f_i)) do
7:          if (p_k = p_j) then
8:             iter = j;
9:          end if
10:      end for
11:      add(p_iter, sp_i);
12:      j = iter+1;
13:   end while
14: end for
```

Algorithm 2 ShortestPath(D)

4.2 Visualization of a network

In the directed graph model, the node represents a process step (a fuzzy-tagged question pattern), and an edge indicates the relation of two process steps. In order to visualize a directed graph, we need to have coordinate, label and tip information that presents the details of the fuzzy-tagged pattern for each node in the graph. Algorithm 3 presents the pseudocode for creating a graph drawing. Due to the limitation of screen size, we can only partially display the graph. Therefore, user is allowed to select process flows in the data table. Assume we have selected the process flows, from which we can index all items that appear in the selected process flows as described in previous section. The graph drawing is placed in a panel with the size of (w, h). The selected process flows are saved in the set F. Initialization of a vertex includes setting the label of the vertex, label color, label font, coordinates in the panel. All nodes of a process flow are naturally placed in one line with even

distance. However, different process flows may share some nodes each other. This makes the layout of the graph drawing have many crossings. Therefore, we develop a GUI to allow user to drag each node in the layout, so that the graph drawing is more readable.

Algorithm 3 CreateGraphDrawing(F, $itemList$, w, h)

```
1: EmptyOf(visitedNodes);
2: Initialize(graph);
3: Initialize(START);
4: add(START, graph);
5: Initialize(STOP);
6: add(STOP, graph);
7: for (each process flow f_i selected in the data table) do
8:    t = START;
9:    for (each process step p_j) do
10:      if (p_j ∉ visitedNodes) then
11:         v = createNewNode(p_j);
12:         add(v, visitedNodes);
13:         index = getIndex(v, visitedNodes);
14:         setCoordinate(v.x,v.y);
15:         setLabel("q"+index,v.lab);
16:         setColor(v.Lab, BLUE);
17:         add(v, graph);
18:         add(e(t,v), graph);
19:      else
20:         index = getIndex(p_j, visitedNodes);
21:         v = vSet(index);
22:         if ((e(t,h) ∉ graph) then
23:            add(e(t,v),graph);
24:            if (e(h,t) ∈ graph) then
25:               setColor(e(t,v),RED);
26:               setColor(e(v,t),RED);
27:            end if
28:         end if
29:      end if
30:      t=v;
31:   end for
32:   if (e(v(p_final),STOP) ∉ graph) then
33:      add(e(v(p_final),STOP),graph);
34:   end if
35: end for
```

Algorithm 3 CreateGraphDrawing(F, $itemList$, w, h)

5 Business process analysis based on the BisoNet

5.1 Statistics based on the BisoNet

In the real-world data, each case has a unique ID for unique customer. The values of userResponseIDs increase as the time when communication goes through for each case. The number of total cases is 2,571. Totally 1,035 agents involved in the service. There are 22 departments in record. The number of different questions is 5,450. There are 2,204 different responses. This indicates that different questions could have a same response. For most cases, the number of contacts is 2, 3 or 4. Table 1 presents the summary of process steps, the number of agents, the number of contact times, duration and the number of department for all cases in the data set. We use fuzzy technology to tag all questions. The number of fuzzy tagged questions is 3,312. Therefore, the size of the network constructed with the sequences of fuzzy tags is about 60 % of the original network that is directly constructed with the sequences of questions.

Table 1 Summary for all cases in the data set

	#steps	Duration	#agents	ContactTimes	#departments
Max	98	9,939	7	7	2
Min	2	11	1	2	1

For each case, there are three important attributes: case status, the number of agents who are involved in the service for the case, and duration of each process step in the case. In order to observe the crossing domains, we partition all cases based on the three attributes, final case status (open or closed), duration sum of the case (short, middle or long), and number of agents (small, middle or large). Table 2 shows the distribution of each partition in the whole data set, where, s-agents, m-agents and l-agents represent small, middle and large number of agents respectively, and s-dur, m-dur and l-dur represent short, middle and long durations, respectively.

It can be seen that most cases are with small number of agents and short duration. Namely, the duration and agent number are not directly relevant to the final status of a case. Therefore, most important thing is to improve the effectiveness of the service – increasing the rate of closed cases.

5.2 Special question tags

We make statistics of repeated question tags in each process flow. The first column in the Table 3 lists the special question tags that are repeated at least once in many cases, and the second column in the table lists how many cases where those special question tags are repeated. These questions may greatly affect the performance of the service. Obviously, the initial question tag "<g1FindCustomerProblem />" in a process flow is repeated in most cases (2,108/2,571), the second most frequently repeated question tag is "<g1SystemTest />". Improving these questions could reduce the repeats, and thus improve the performance of the serve.

We partition all cases into two parts: Open and Closed, and repeat the experiments on the two partitions of cases respectively. The third and fourth columns in the Table 3 list the numbers of cases in the open and closed cases, where those questions are repeated. Obviously, all those special questions are frequently repeated in open cases.

Table 2 Partition distributions based on the three attributes

Partitions	Number of cases	Distribution (%)
Open	1,170	45.5
Closed	1,401	54.5
s-Agents	2,289	89
m-Agents	270	10.5
l-Agents	12	0.5
s-dur	2,512	97.7
m-dur	58	2.26
l-dur	1	0.04

Table 3 Special questions

Questions	#Cases	#Open Cases	#Closed Cases
<g1FindCustomerProblem / >	2,108	2,035	73
<g1SystemTest / >	1,840	1,761	79
<g1FindProblemDetails / >	1,385	1,321	64
<g1FollowKM / >	931	875	56
<g1ProblemFeature / >	912	872	40
<g1CallbackQ / >	441	436	5
<g1CustErrorCodeReport / >	436	430	6
<g1AdviseCustomer / >	306	268	38
<g1LineSpeed / >	288	270	18
<g1Reboot / >	286	277	9
<g1FollowKM / > <g1FindProblemDetails / >	280	259	21
<g1FindProblemDetails/ > <g1ProblemFeature / >	229	201	28
<g1Reboot / > <g1FollowKM / >	146	142	4
<g1CircuitSyncTest / >	122	115	7
<g1Signal / >	109	91	18
<g1MCI / >	102	84	18

5.3 Simplifying the network with the shortest paths

The shortest path of a case is obtained through removing all nodes in a cycle, but keep the repeated node and those nodes that are not in a cycle. Here, we should notice that each node in the graph represents the fuzzy tag of a question in a process flow, and those questions, which have similar semantic meanings but have logic order on time axis, are labeled with different fuzzy tags. We can simplify the network through using the shortest paths. Figure 3 displays the network that represents the three selected process flows. In the visualization window, the network of the three selected process flows is depicted. 'S' represents the start point of a process flow, 'T' represents the terminating point of the process flow. The yellow nodes indicate the node is repeated successively in process flows.

The three selected workflows are:

$F_1 : q_2, q_{13}, q_0, q_0, q_2, q_7, q_2, q_0, q_{50}, q_{58}, q_{52}, q_7, q_{51}, q_{53}, q_{50}, q_{58}, q_{59}.$

$F_2 : q_2, q_{13}, q_2, q_0, q_0, q_{53}, q_2, q_8, q_2, q_{60}.$

$F_3 : q_2, q_{13}, q_2, q_0, q_0, q_{53}, q_2, q_7, q_7, q_8, q_{67}, q_{65}, q_{64}, q_{24}, q_{66}, q_{63}, q_{89}, q_9, q_{61}, q_{24}, q_{62}.$

It can be seen that there are the same sequence of questions in the three process flows. In the real scenario, agents should avoid repeating a question as possible as they can. The process flow F_1 starts with q_2. It can be found that the last q_2 is followed by q_0, the last q_0 is followed by q_{50}, the last q_{50} is followed by q_{58}, and the last q_{58} is followed by q_{59}, and then the process flow is closed. Therefore, the shortest path of the process flow F_1 is q_2, q_0, q_{50}, q_{58}, q_{59}.

The process flow F_2 starts from q_2 as well, the last q_2 is followed by q_{60}, and then the process flow is closed. Therefore, the shortest path of process flow F_2 is q_2, q_{60}.

The process flow F_3 starts with q_2 too, the last q_2 is q_7, the last q_7 is followed by q_8, the last q_8 is followed by q_{67}, the last q_{67} is followed by q_{65}, the last q_{65} is followed by q_{64}, the last q_{64} is followed by q_{24}, the last q_{24} is followed by q_{62}, and then the process flow is closed. Therefore, the shortest path of the process flow F_3 is: q_2, q_7, q_8, q_{67}, q_{65}, q_{64}, q_{24}, q_{62}.

Figure 4 displays the network that represents the shortest paths of the three selected process flows in Fig. 3. Through the simplified graph, we can further observe these special question nodes and their follows.

It should be noted that the simplified business process flows depend on the fuzzy tags of questions, produced with fuzzy grammar technology, as the tags represent the question patterns in the business process flows. Assume the fuzzy tags can exactly represent the question patterns in business process flows, then we have the following theorem:

Theorem 1 The shortest path for the process flow of a case forms the optimal solution of the case.

Proof

1. Given a process flow, in the form of $S,\ldots, p_{i\text{-}1}, p_i,\ldots, p_j, p_{j+1}\ldots, T$, where only p_i and p_j have the same fuzzy-tag p. There are some question tags between two questions p_i and p_j. These questions with the question tag p form a cycle in the network that represent the process flow.
2. There exists a process $p_{i\text{-}1}$ before the process p_i, and a process p_{j+1} after the process p_j. Due to p_i and p_j have the same fuzzy tag p. Namely, node p represents the two processes p_i and p_j. Therefore, the shortest path of the process flow should be $S,\ldots, p_{i\text{-}1}, p, p_{j+1},\ldots, T$ for the process flow, as the cycle is removed.

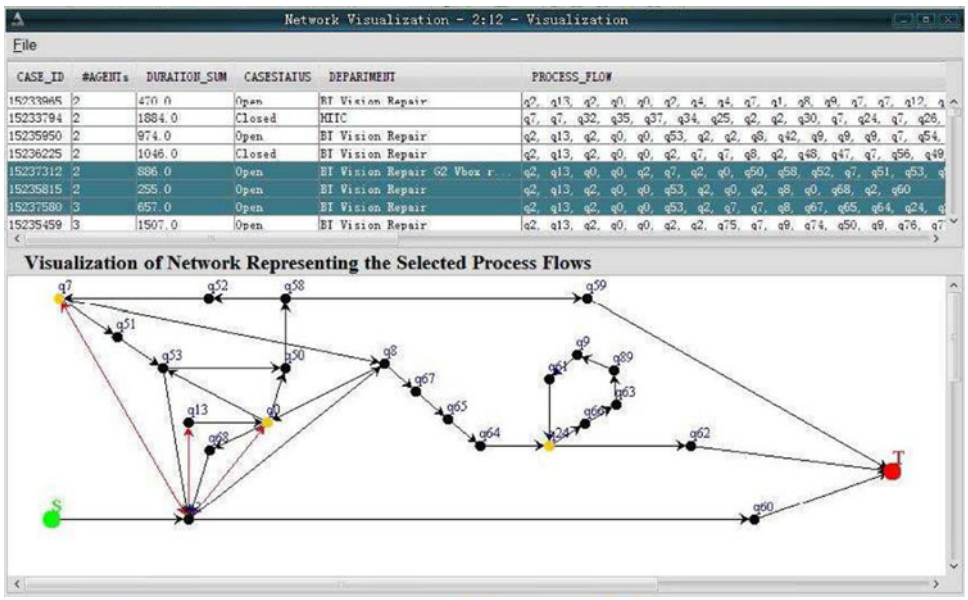

Fig. 3 The viewing panel of the visualization node

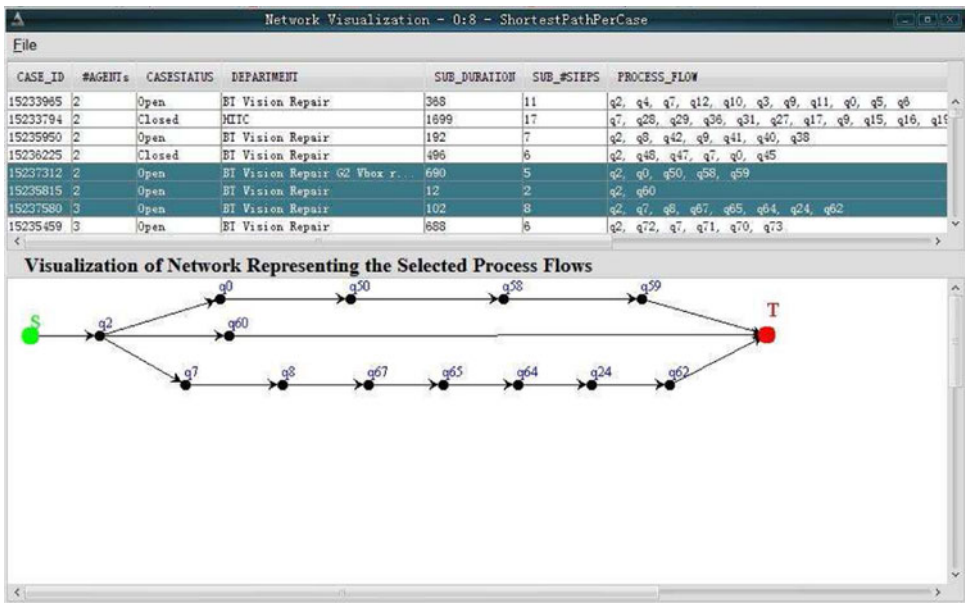

Fig. 4 The viewing panel of the ShortestPathPerCase node

3. As we record the process after p_j, this means that the node p should represent the process p_j. In the real-world data, it means that p_j is the right question to get right direction at the point p_{j+1}.
4. Scanning the whole process flow, we can get the shortest path through using the approach above to eliminate all cycles in the network for the process flow. Namely, every process step is the right question, which leads to right direction, and make the process go further.
5. As we label those questions, which have similar semantic meanings and have logic order on time axis, with different fuzzy tags, the procedure of eliminating cycles will not lose right steps.

Therefore, the shortest path for the process flow of a case forms the optimal solution for the case.

6 Conclusions

Bisociation network presents a novel framework for data analysis. In this paper we investigate the bisociation of business processes on the network. We developed a directed graph model to map the relations between business process flows. As we use data from the call service centre, we use the fuzzy-tagged question sequence of a case on time axis as a process flow. The statistic data based on the network show that the duration and agent number of a case

seems not related to the final status of the case, and some special questions are the key steps that may result in a case unsolved. With the merit of network, we simplify the network through constructing the directed graph with the shortest paths of all cases, and the simplified network may represent an optimal process pattern. Moreover, the simplified BisoNet provides an intuitive view of the relations between process flows for further analysis. We may further apply the Formal Concept Analysis approach to analyze the properties of the special question patterns, and thus to see what the best question styles are for the service. Hence, we can present some suggestions to the call service centre for improving their service.

Acknowledgments This work is based on the research supported by the EU PF7 BISON project, when the first author worked on the project in the University of Bristol. ZQ also thanks the NECT Program of MOE, China and the SRF for ROCS.

References

1. Jackson AD, Somers KM, Harvey HH (1989) Similarity coefficients: measures of co-occurrence and association or simply measures of occurrence? Am Nat 133(3):436–453
2. Baeza-Yates RA, Ribeiro-Neto B (1999) Modern information retrieval. Addison-Wesley Longman Publishing Co., Boston
3. Koestler A (1964) The act of creation. Hutchinson, London
4. Berthold MR, Dill F, Kötter T, Thiel K (2008) Supporting creativity: towards associative discovery of new insights. In: Advances in knowledge discovery and data mining, the proceedings of Advances in Knowledge Discovery and Data Mining 12th Pacific-Asia Conference, PAKDD 2008 Osaka, Japan, May 20–23
5. Albert R, Barabasi AL (2002) Statistical mechanics of complex networks. Rev Mod Phys 74(1):47C 97
6. Bales ME, Johnson SB (2006) Graph theoretic modeling of large-scale semantic networks. J Biomed Inform 39(4): 451–464
7. Bosman DWJ, Blom E-J, Ogao PJ, Kuipers OP, Roerdink JBTM (2007) MOVE: a multilevel ontology-based visualization and exploration framework for genomic networks. Silico Biol 7:35–59
8. Shen Z, Ma K-L, Eliassi-Rad T (2006) Visual analysis of large heterogeneous saocial networks by semantic and structural abstraction. IEEE Transact Vis Comput Graphics 12(6):1427–1439
9. Tollis IG, Di Battista G, Eades P, Tamassia R (1999) Graph drawing: algorithms for the visualization of graphs. Prentice Hall, Englewood Cliffs
10. Zhong Z, Liu Z, Li C, Guan Y (2011) Event ontology reasoning based on event class influence factors. Int J Mach Learn Cyber. doi:10.1007/s13042-011-0046-8
11. Castillo JJ (2011) A WordNet-based semantic approach to textual entailment and cross-lingual textual entailment. Int J Mach Learn Cyber 2(3):177–189
12. Kötter T, Thiel K, Berthold MR (2010) Domain bridging associations support creativity. In Proceedings of the first international conference on computational creative (ICCC-X), Lisbon, Portugal, pp 7–9
13. Mozetič I, Lavrač N, Podpecan V, Novak PK, Motaln H, Petek M, Gruden K, Toivonen H, Kulovesi K (2010) Bisociative knowledge discovery for microarray data analysis. In Proceedings of the first international conference on computational creative (ICCC-X), Lisbon, Portugal, pp 7–9
14. http://www.knime.org
15. Andersen B (1999) Business improvement toolbox. ASQ Quality Press, Milwaukee
16. Trischler WE (1996) Understanding and applying value added assessment. ASQ Quality Press, Milwaukee
17. http://www.netmba.com/operations/process/analysis
18. Martin TP, Shen Y, Azvine B (2008) Incremental evolution of fuzzy grammar fragments to enhance instance matching and text mining. IEEE Transact Fuzzy Syst 16:1425–1438
19. Wang X-Z, Tsang E, Zhao S-Y, Chen D-G, Yeung D (2007) Learning fuzzy rules from fuzzy examples based on rough set techniques. Inform Sci 177(20):4493–4514
20. Wang X-Z, Dong C-R (2009) Improving generalization of fuzzy if-then rules by maximizing fuzzy entropy. IEEE Trans Fuzzy Syst 17(3):556–567