Bisociative Discovery in Business Process Models

Trevor Martin[1,2] and Hongmei He[1,3]

[1] Artificial Intelligence Group, University of Bristol BS8 1UB UK
firstname.lastname@bristol.ac.uk
[2] BT Innovate and Design, Adastral Park, Ipswich IP5 3RE, UK
[3] Current Address : School of Computing and Intelligent Systems,
University of Ulster, Magee Campus, BT48 7JL, UK
h.he@ulster.ac.uk

Abstract. Bisociative knowledge discovery - finding useful, previously unknown links between concepts - is a vital tool in unlocking the economic and social value of the vast range of networked data and services that is now available. An important application for bisociative knowledge discovery is business process analysis, where bisociation could lead to improvements in one domain being disseminated to other domains. We identify two forms of bisociation, based on structural similarity, that are applicable to business processes, and present examples using real-world data to show how bisociative reasoning can be applied.

1 Introduction

Business Intelligence has been defined as a broad category of "applications and technologies for gathering, storing, analyzing, and providing access to data to help users and automated systems make fact-driven business decisions[1] " Business process analysis is a subfield, arising from the need for companies to learn more about how their processes operate in the real world. According to Andersen and Fagerhaug [1], a business process is "a logical series of related transactions that converts input to results or output" In particular, business process analysis involves aspects such as

- discovery of process models, based on event logs
- process conformance (do event logs follow prescribed paths through process models)
- process analysis (e.g. are there significant bottlenecks, can process instances be grouped on the basis of different paths through the process model)
- extension of process models, in cases where the actual process execution is not properly reflected in the model.

A business process can be represented naturally as a graph, and hence business processes form suitable inputs for the bisociation operations described in [2] – particularly

[1] This definition was taken from www.oracle.com/us/solutions/sap/database/sapbocmtsizing-352636.pdf; there are many similarly phrased descriptions on the web.

M.R. Berthold (Ed.): Bisociative Knowledge Discovery, LNAI 7250, pp. 452–471, 2012.

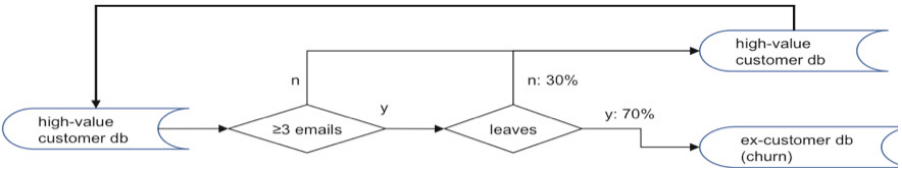

Fig. 1. Simplified process diagram showing 70% of high value customers leave after sending 3 or more emails to a support centre

the notion of graph (structural) similarity. There are, however, some features which distinguish process graphs from most of the other (document- and graph- based) demonstrators mentioned in [3]. In particular, the sequential nature of most business processes is fundamental - there is a specific order required for the steps within a process. In contrast, measures of similarity which depend on counting the number of occurrences (or co-occurrences) of words, nodes, etc. do not require a specific order of occurrence. Additionally, the BisoNet representation assumes that numerical edge labels reflect the probability or strength of a link. In process graphs, edges can be labelled by the time taken to move from one process stage to the next. Notwithstanding these differences, the Bison framework has been used to generate useful suggestions for domain experts, showing its versatility and potential.

As discussed in [4], creative knowledge discovery can be distinguished from "standard" knowledge discovery by defining the latter as the search for explanatory and/or predictive patterns and rules in large volume data within a specific domain. For example, a knowledge discovery process might examine an ISP (internet service provider)'s customer database and determine that people who have a high monthly spend and who send more than three emails to the support centre in a single month are very likely to change to a different provider in the following month. Such knowledge is implicit within the data but is useful in predicting and understanding behaviour. Figure 1 illustrates this as a summary process diagram (the actual set of process instances would be a more complex graph).

By contrast, creative knowledge discovery is more concerned with "thinking the unthought-of" and looking for new links, new perspectives, etc. Such links are often found by drawing parallels between different domains and looking to see how well those parallels hold - for example, compare the ISP example mentioned above to a hotel chain finding that regular guests who report dissatisfaction with two or more stays often cease to be regular guests unless they are tempted back by special treatment (such as complimentary room upgrades), as illustrated in Fig. 2. This is a simple illustration of similar problems (losing customers) in different domains. There is a structural similarity, and a solution in one domain (complimentary upgrades) could inspire a solution in the second (e.g. a higher download allowance at the same price). Of course, such analogies may break down when probed too far but they often provide the creative insight necessary to spark a new solution through a new way of looking at a problem. In many cases, this inspiration is referred to as "serendipity", or accidental discovery.

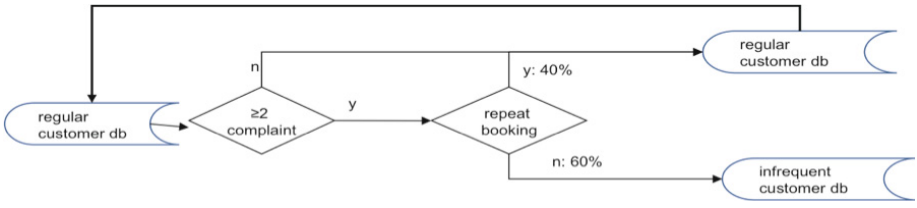

Fig. 2. (a) Simplified process showing 60% of regular customers do not re-book after making 2 or more complaints

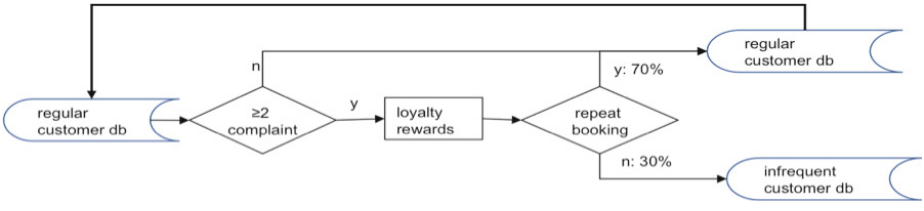

Fig. 2. (b) Updated process introducing loyalty rewards after which only 30% of regular customers do not re-book after making 2 or more complaints

The core of the Bison project is the automation of creativity, in this sense of making novel connections between previously unrelated concepts. For networks representing business processes, we have investigated two possible modes of bisociation:

(i) structural bisociation between two process networks from different domains, with respect to a specific mapping between node types in the two networks. The processes in one domain are assumed to need improvement. We look for sections of the two networks where there is high similarity between most of the nodes and links, with a small segment exhibiting low similarity. The operation of bisociation swaps the (mapped) low similarity segments, as illustrated in the ISP/hotel chain example above.

(ii) conceptual bisociation, which is a form of structural bisociation, requiring one process network and a generalisation / specialisation hierarchy on the node types. Similar process graphs for use in bisociative combination can be generated using the generalisation/ specialisation hierarchy. The origin of this approach is explained below.

Sherwood [5] proposed a systematic method, in which a situation or artefact is represented as an object with multiple attributes, and the consequences of changing attributes, removing constraints, etc are progressively explored. For example, given an old style reel-to-reel tape recorder as starting point, Sherwood's approach is to list some of its essential attributes, substitute plausible alternatives for a number of these attributes, and evaluate the resulting conceptual design or solution. Table 1 shows how this could have led to the Sony Walkman in the late 70s [5]. Again, with the benefit of hindsight the reader should be able to see that by changing magnetic tape to a hard disk and by also considering the way music is purchased and distributed, the same method could (retrospectively, at least) lead one to invent the iPod. Of course,

having the vision to choose new attributes and the knowledge and foresight to evaluate the result is the hard part - and the creative steps are usually only obvious with hindsight.

This systematic approach is ideally suited to handling data which is held in an object-attribute-value format, with a taxonomy defined on the domain of (at least) one attribute. This provides a means of changing/generalising attribute values, so that "sensible" changes can be made (e.g. *mains electricity, battery* are possible values for a *power* attribute). Representing an object O as a set of attribute-value pairs

$$\{(a_i, v_i) | \textit{attribute } a_i \textit{ of object } O \textit{ has value } v_i\}$$

we generate a new "design"

$$O^* = \{(a_i, T(v_i))\}$$

by changing one or more values using *T*, a non-deterministic transformation of a value to another value from the same taxonomy. Given sufficient time, this would simply enumerate all possible combinations of attribute values. We can reduce the search space by looking at the solution to an analogous problem in a different domain, as in the structural bisociation method, so that analogies can be found. This requires tools for taxonomy matching e.g. [6], for converting the data into object-attribute-value form (or extending the number of attributes), and for detecting structure arising from the attribute patterns. The latter two are covered in the next section.

Table 1. Attributes of two music players (taken from [4])

Conventional tape recorder	Sony Walkman
big	small
clumsy	neat
records	does not record
plays back	plays back
uses magnetic tape	uses magnetic tape
tape is on reels	tape is in cassette
speakers in cabinet	speakers in headphones
mains electricity	battery

2 Tools Used for Pre-processing Data

Two sets of process data were examined, as described in section 3. In order to identify taxonomic elations within the data, we used fuzzy formal concept analysis. One dataset contained short text sequences at each process step, and fuzzy grammars were used to extract key features from the text, prior to the concept analysis. For completeness, both methods (fuzzy grammars and fuzzy formal concept analysis) are briefly described in this section.

2.1 Fuzzy Grammars

A text fragment is a sequence of symbols, and it is common to use shallow processing (e.g. presence / absence of keywords) to find attributes of the text. This operation can

be viewed as a way to label sub-sequences of symbols with different tags indicating the nature of the text fragment. For example, we could have a schema for the process step "arranging to call back a customer", including attributes such as the time, which party requested the call-back, reason for the call-back, etc. It is not necessary to extract every attribute from the text, and it is possible that information may not be recognised due to unexpected ways of expressing the information or abbreviations, mis-spelling etc. The latter case can be handled by extending the matching process to include fuzzy matches, that is sequences of symbols that almost conform to the pattern and are sufficiently close to be recognisable as examples of the pattern.

It is often not possible to define simple patterns (such as regular expressions) which can reliably identify the information structure. The key contribution of the fuzzy grammar approach is the definition and use of approximate grammars, where a degree of support is calculated for the matching process between an approximate grammar and a sequence of symbols that may not precisely conform to the grammar.

For example, the following fragments are all examples where a call back has been arranged:

spoke with Michelle the cust partner need a call after 2hrs.
cust need a call tomorrow.
cust is going to think about charge and call back if needs to.
eu is going to call back in a minute on a different phone

(*cust* is an abbreviation for *customer,* and *eu* is an abbreviation for *"end user"* i.e. customer). It is difficult to anticipate all possible forms (including abbreviations) in a regular expression. Full details of the fuzzy grammar approach are given in [7, 8]

2.2 Fuzzy Formal Concept Analysis

Formal concept analysis (FCA) [9, 10] is a way of extracting hidden structure (specifically, lattice-based structure) from a dataset that is represented in object-attribute-value form. In its simplest form, FCA considers a binary-valued relation on a set of objects *O* and attributes *A*

$$R \subseteq O \times A$$

The structure *(O, A, R)* is a formal context. Given *X*, a subset of objects and *Y*, a subset of the attributes,

$$X \subseteq O$$
$$Y \subseteq A$$

the operators $\uparrow$ and $\downarrow$ are defined as follows:

$$X^{\uparrow} = \{y \in Y \mid \forall x \in X : (x, y) \in R\} \quad (1)$$

$$Y^{\downarrow} = \{x \in X \mid \forall y \in Y : (x, y) \in R\} \quad (2)$$

Any pair (*X, Y*) such that $X^{\uparrow}$=Y and $Y^{\downarrow}$=X is a formal concept.

Table 2. A simple formal context

	a1	**a2**	**a3**
o1	1	0	1
o2	1	1	0
o3	0	0	1

For example, Table 2 shows the relation between three objects *o1, o2, o3* and attributes *a1, a2* and *a3*. The resulting concepts are

({o2}, {a1, a2})
({o1}, {a1, a3})
({o1, o2}, {a1})
({o1, o3}, {a3})

i.e. the object *o2* is the only one to have both attributes *a1* and *a2*, objects *o1* and *o2* are the only objects to have attribute *a1* etc. In larger tables, this is less obvious to inspection.

A partial order, ≤ , is defined on concepts such that

$$(X1, Y1) \leq (X2, Y2)$$

means $X1 \subseteq X2$ and $Y2 \subseteq Y1$ i.e. the higher concept contains more objects and fewer conditions (attributes that must be true) than the lower concept. This gives rise to a lattice, enabling us to discover relations between concepts - for example, in Fig 3 we see that attributes *a2* and *a3* in Table 2 are mutually exclusive, since no object has both attributes i.e. the least upper bound is equal to the top element of the lattice and the greatest lower bound is equal to the bottom. Each node drawn as a large circle represents an object (or objects), and each object has all the attributes attached to its node and all higher nodes linked directly or indirectly to it. The software draws a node with a black lower half if it represents an object (or set of objects), and with a blue upper half if it is the highest node corresponding to an attribute; this convention allows the diagram to be simplified by omitting labels.

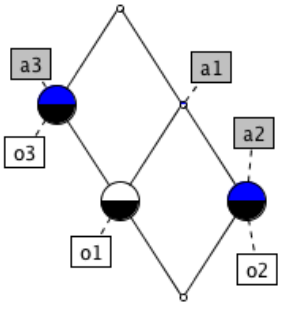

Fig. 3. Concept lattice corresponding to the formal context in Table 2. The lattice is drawn using the *conexp* tool (conexp.sourceforge.net).

2.2.1 Conceptual Scaling

For attributes which take a range of values (rather than true / false as above), the idea of "conceptual scaling" is introduced [11]. This transforms a many-valued attribute (e.g. a number) into a symbolic attribute - for

example, an attribute such as "*time in seconds*", given an integer or real value between 0 and 200 could be transformed to attributes "*timeLessthan50*", *"timeFrom50to99"*, etc. These derived attributes have true/false values and can thus be treated within the framework described above.

2.2.2 Fuzzy FCA

Clearly the idea of conceptual scaling is ideally suited to a fuzzy treatment, which reduces artefacts introduced by having to draw crisp lines between the categories. The idea of a binary-valued relation is easily generalised to a fuzzy relation, in which an object can have an attribute to a degree. Instead of defining the relation

$$R \subseteq O \times A$$

as

$$R : O \times A \rightarrow \{0,1\}$$

we define it as a fuzzy relation

$$R : O \times A \rightarrow [0,1]$$

where each tuple of R, $(o,a) \in R$ has a membership value in [0, 1].

We define a fuzzy formal concept as a pair X, Y where X is a fuzzy set of objects and Y is a crisp set of attributes such that $X^{\uparrow}$=Y and $Y^{\downarrow}$=X where

$$X^{\uparrow} = \left\{ y \in Y \mid \forall x \in X : \mu_R(x,y) \geq \mu_X(x) \right\} \tag{3}$$

$$Y^{\downarrow} = \left\{ x / \mu_X(x) \middle| \mu_X(x) = \min_{y \in Y} \left(\mu_R(x,y) \right) \right\} \tag{4}$$

It is also possible to define crisp sets of objects and fuzzy sets of attributes, using a dual of these operators.

Our approach is related to that of Belohlavek (e.g. [12], [13]) but differs in some important practical and philosophical respects. By restricting ourselves to crisp attribute sets (intensions), we simplify the calculation of the closures but (more importantly) we follow Zadeh's original motivation for fuzzy sets - modelling predicates for which there is no clear boundary between membership and non-membership. This notion is based on a universe of discourse, and a generalisation of the characteristic function corresponding to the set defined by a predicate, reflecting the fuzziness in the predicate. The extension of the predicate is fuzzy but the underlying universe is not - for example, the set of *small* dice values could be 1 and 2 with full membership, and 3 with intermediate membership. From the set of possible values, {1, 2, 3, 4, 5, 6} we identify a fuzzy subset of *small* values. Given the value 1, we can say that it definitely has the attribute *small*, whereas given the value 3 we can say that it only has the attribute *small* to an intermediate degree. If we are told that a single dice roll has resulted in a *small* value, we can model it as a possibility

distribution using the same membership function. We have a crisp event (small dice roll) with a fuzzy attribute (value displayed on the dice).

In contrast, methods based on residuated implication allow both intension and extension to be fuzzy.

Methods based on the alpha-cut are essentially crisp, once the choice of a threshold is made; changing the threshold is equivalent to defining a different conceptual scaling.

3 Process Data

Access to a number of process datasets was provided by an industrial partner, BT Innovation and Design. The datasets were taken from real operations, and were anonymised by removal of obvious personal details; in order to ensure commercial and customer confidentiality, the datasets were not taken offsite. Two datasets were selected for study:

1. Repair Data - a dataset of approximately 55000 process instances, stored in an XML format. Each process instance represented a single case of fault analysis and repair, and contained data on the tasks carried out in that case, the length of time taken for each task, the location, etc. Process and task names were structured but not necessarily understandable - for example, task names (or WorkFlowModelElement, using the XML tag) mostly consisted of a 5 character code (e.g. UK106) representing the centre at which the task was carried out, followed by a three character identifier (e.g. TS2) representing the actual task. Process instances varied in length from 3 up to 440 (including start and end) e.g.

start BB450GAB end

Figure 4 shows the distribution of path lengths. Over 30 centre identifiers were included in the data, representing a wide range of repair functions within the company. Python and Unix scripts, plus custom java modules were used with KNIME to convert the data into BisoNet form.

2. Call-Centre Data - a dataset of call-centre interactions, related to different business units within the company. Each process instance involved a number of questions designed to elicit information (customer details, problem symptoms, etc) and find a solution (including an immediate fix, additional tests, or appointment for an engineer to visit). These questions were a mixture of scripted and free-form text. Each step in the process had a unique identifier; additional data included an identifier for each process instance, the customer and call-centre agent, date/time and duration of the step, and information about the handling department and ultimate resolution of the problem. The data was recorded automatically, with scripted questions provided by the system and unscripted questions plus responses entered by the call centre agent. The free-form nature of unscripted questions and the number of abbreviations, mis-spellings and grammatical shortcuts taken when these questions are typed added an additional complication to the dataset.

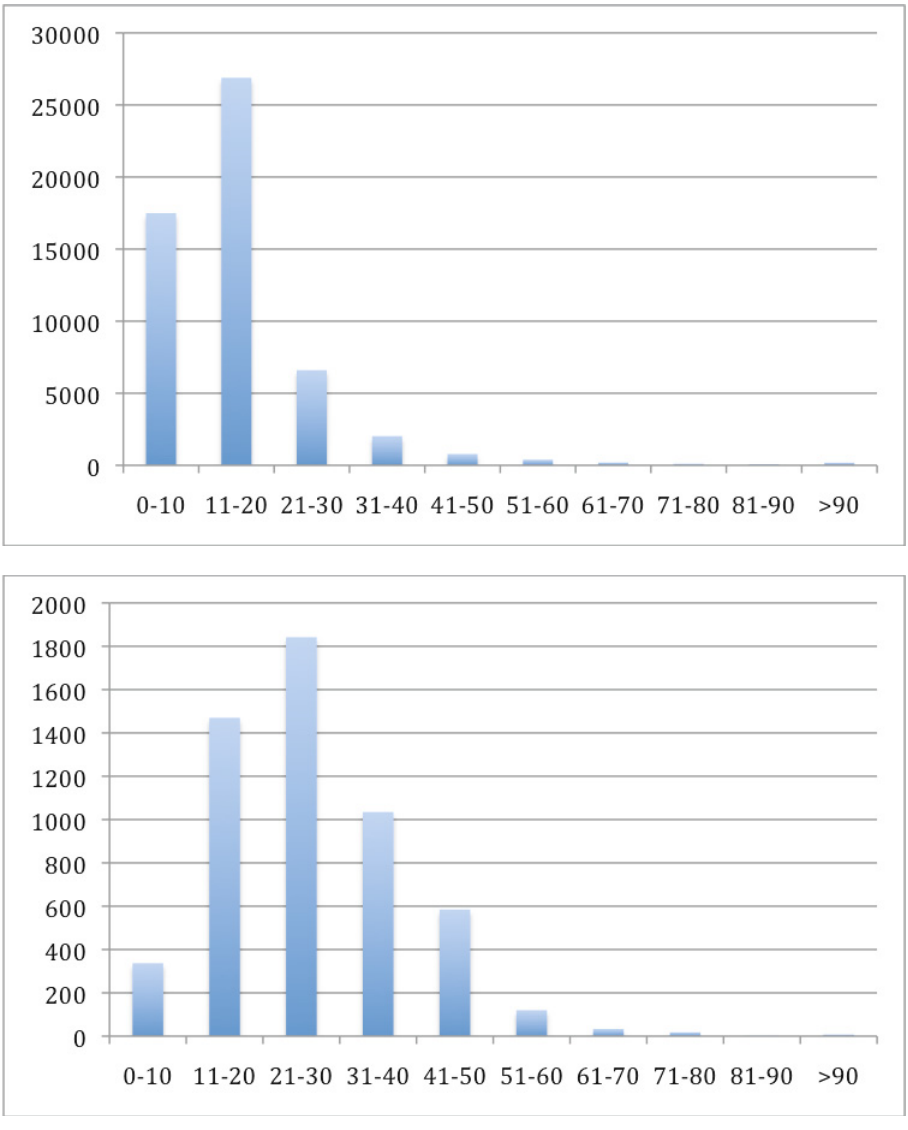

Fig. 4. Distribution of path lengths in dataset 1 (top) and dataset 2 (bottom)

The dataset consisted of around 5500 process instances and a total of over 65000 steps. The process data was in the form of a series of questions (and answers) plus time taken, and identifiers for caller and call-centre agent, date/time and other data. The complete set of attributes (with brief description) was

CASE_ID	a unique identifier for this process instance
USER_RESPONSE_ID	unique identifier for this process step

AGENT	call centre agent
CONTACT_CREATED	timestamp for one (or more) steps
CUSTOMER	identifier for customer
QUESTION	text of scripted or unscripted question / other notes
RESPONSE	text of answer / summary of test result
DURATION	System-generated time taken by this process step
EXIT_POINT1, 2, 3	internal data
CASESTATUS	boolean indicating whether process has finished
DEPARTMENT	name of dept that handled this process

Figure 4 shows the distribution of path lengths (note that dataset 1 contains approximately 10 times as many instances as dataset 2). Table 4 shows a small part of an interaction; the "question" field was used to record scripted questions and notes made by the agent. Each sequence of questions as a process instance, represented as a directed graph.

Because there was so much flexibility in the question/answer patterns, we pre-processed the text to extract key features, using fuzzy grammar tagging [7] to add attributes. This went beyond a simple keyword-recognition approach (which was found to be inadequate) and was able to cope with the non-standard language employed in the questions. Table 4 shows examples of the tags added; these were used as node labels in the directed graphs.

Subsequent to the tagging, a combination of Unix scripts and customised Java / KNIME workflows were used to convert the data into Bisonet form.

Table 3. Example of call centre interaction (a single process instance)

Question	Response	Duration
What is the call about?	New fault	11
What type of fault?	No incoming calls	30
Is the Customer calling from home?	Yes, using line with the fault	4
Is this an early life Customer?	No	3
Are there any Open orders or billing restrictions (including issues from high value accounts) on the account which could be causing this problem?	No problems	4
Ask the Customer if they are reporting an issue with a BB talk hub phone, a VOIP phone or an internet phone.	No	5
Is there an open SR with a valid test within the last 30 minutes?	No Open SR	2
Start the test to check settings on the asset in OneView, use the Incoming calls option.	OK	34
…	…	…
System Test Line Test	Green Test Result	3
Have all line/equipment checks been completed for this problem?	Yes	2
cust will do more checking	Progress Saved	147

Table 4. Tags applicable to example shown in Table 3

Question	Fuzzy Tag(s)
What is the call about?	<g1FindCustomerProblem />
What type of fault?	<g1FindProblemDetails />
Is the Customer calling from home?	<g1FindProblemDetails />
Is this an early life Customer?	<g1FindProblemDetails />
Are there any Open orders or billing restrictions (including issues from high value accounts) on the account which could be causing this problem?	<g1FindAccountDetails />
Ask the Customer if they are reporting an issue with a BB talk hub phone, a VOIP phone or an internet phone.	<g1ProblemFeature />
Is there an open SR with a valid test within the last 30 minutes?	<g1SystemTest />
Start the test to check settings on the asset in OneView, use the Incoming calls option.	<g1SystemTest />
…	…
System Test Line Test	<g1CheckTestResult />
Have all line/equipment checks been completed for this problem?	<unknown />
cust will do more checking	<g1EndCall />

The process instances were derived from different call centres, dealing with different business areas (business and consumer phone services, broadband, cable broadcasting, etc). This was indicated to some extent by the "Department" field, and we took this as a starting point for finding different Bison domains within the data. Departments whose processes involved similar sequences of questions were grouped together using fuzzy FCA; we also divided each of these domains into good and bad process instances. The characteristics of a good process instance are

- it does not involve multiple interactions,
- it does not contain loops, and
- it is completed in a reasonably short time.

4 Bisociative Knowledge Discovery in Business Processes

4.1 Illustrative Example

We first provide a simple illustration to show how the fuzzy FCA approach can aid in conceptual bisociation for creative knowledge discovery. The data used to create the examples shown in Figs 1 and 2 leads to the concept lattices shown in Figs 5 and 6. Note that this is a "toy" example and the similarity between lattices is obvious in this case. We have developed methods which facilitate this matching by comparing lattices [14] and by finding fuzzy association confidences in fuzzy class hierarchies [15].

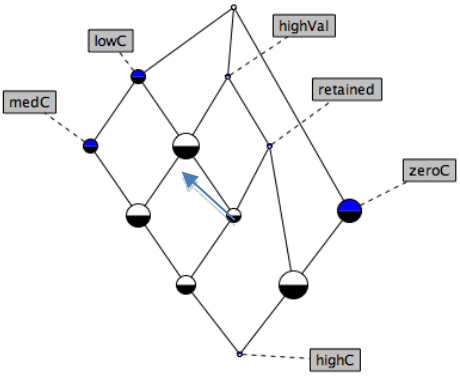

Fig. 5. Concept lattice corresponding to the process shown in Fig 1. The arrow indicates nodes used in calculating the association confidence (key performance indicator)

The attribute *highVal* indicates that a customer is a member of the set of high valued customers at the beginning of a specified period, time point t_0; the attribute *retained* indicates membership at the end of the period, time point t_1, and *zeroC, lowC, medC, highC* show the number of complaints (respectively, zero, low, medium, high) made in the period. Note that every object with membership in *medC* is also in *lowC* because of the overlapping nature of these sets. In this dataset, there are no customers who have made a high number of complaints, so the concept labelled *highC* is at the bottom of the lattice with no elements,.

Figure 5 shows the concept lattice for the ISP example of Fig. 1. The arrow indicates the association rule between the set of high value customers who complained a non-zero (*low* or *medium)* number of times and the subset who also satisfy the *retained* attribute. In this case, the confidence is 40% and this forms a key performance indicator for the process.

Figure 6 shows concept lattices corresponding to the hotel example of Fig. 2. The introduction of the *reward* attribute makes a major difference to the key performance indicator, raising it from 30% to 70%. Because the lattice is isomorphic to Fig. 5, the automated creative knowledge discovery process suggests that introduction of "something like" a rewards programme could also benefit the ISP in retaining high value customers Although the parallels are obvious here, practical examples require considerable search to find the best match between lattices. Work in this area is ongoing, outside the Bison project.

4.2 Business Process Example - Definition of Domains

Our second application looks for structural bisociations, and we start by defining domains. In both cases (datasets 1 and 2), data was gathered during a specific time interval, and was not balanced across different business units. Since the business units are (effectively) anonymised, the first step was to group processes from different units into domains for bisociation.

Because the range of problem areas is large (domestic and businesses customers using a wide range of services such as standard telephone, voice over IP, all aspects of broadband connections - including wireless - and TV), it is valid to regard different centres as different domains. At the same time, there is significant overlap between some centres - for example, a centre dealing with domestic broadband in one region is effectively identical to a centre dealing with domestic broadband in another region. The first stage of analysis in both cases was to identify similarities between centres; this was achieved using fuzzy formal concept analysis [9, 15] In dataset 1, we extracted relative frequencies of task-codes associated with the various centres, converted the frequencies to fuzzy memberships using a mass assignment approach [16] and used the task- code/membership pairs as attributes for FCA. The result (Fig 7, displayed using software adapted from conexp.sourceforge.net) shows that some centres are very similar (for example, UK450, GT450, WA450 near the top of the diagram), that there is a generalisation hierarchy (the UK450, GT450, WA450 cluster is more general than BB450, in terms of tasks performed), and that there are dissimilarities (e.g. UK107, UK106 near the bottom left have no overlap). The opinion of a domain expert was that these groupings were a realistic reflection of the functions.

In dataset 2, we used the fuzzy tags assigned by fuzzy grammar analysis as attributes, leading to the concept lattice in Fig 7(b). Here, it is possible to assess the groupings by inspection of the centre names - for example, it is not surprising to see the strong connection between centres dealing with businesses (six connected nodes on right hand side), with vision products (three nodes on left), etc.

4.3 Bisociations

There are a number of indicators for “good” and “bad” process execution. Reaching a satisfactory end point in a relatively short time, with no unnecessary loops is an ideal situation; cases which require repeated work, suffer from long delays and/or incorrect execution paths are not ideal.

Multiple Domains in a Single Dataset

Having defined different domains within each dataset, we looked for possible overlapping concepts between the domains. We first combined all process instances within a domain by adding universal start-process and finish-process nodes, and combining common paths from / to these universal nodes (using a modification of the DAWG algorithm in [17]).

In dataset 2, we used the fuzzy tags assigned by fuzzy grammar analysis as attributes, Three variants were initially produced for each set of instances. The first retained loops, but unrolled them so that each vertex had indegree and outdegree of 1 (other than the start-process and finish-process nodes). Second and third variants were produced, in which a node representing the loop (including the source/sink node of the loop) was given a derived identifier or given the same arbitrary identifier as all loops. Figure 8 shows an example of a single process with a loop from dataset 1.

Bisociations were sought by looking for structural similarity between domains. This was interpreted as finding a consistent mapping from the set of nodes in one graph to the set of nodes in the second graph, such that paths (i.e. process instances) are preserved (NB timing data for process steps was ignored here). For two domains (V_1, E_1) and (V_2, E_2) we search for a mapping

$$f : V_1 \rightarrow V_2$$

such that for each process instance from domain 1

$P_{1i} = (v_{i1}, v_{i2}, \ldots, v_{in})$ where each $v_{1i} \in V_1$

there is a corresponding process

$$P_{2k} = f(P_{1j}) = (f(v_{j1}), f(v_{j2}), \ldots, f(v_{jn}))$$

in domain 2.

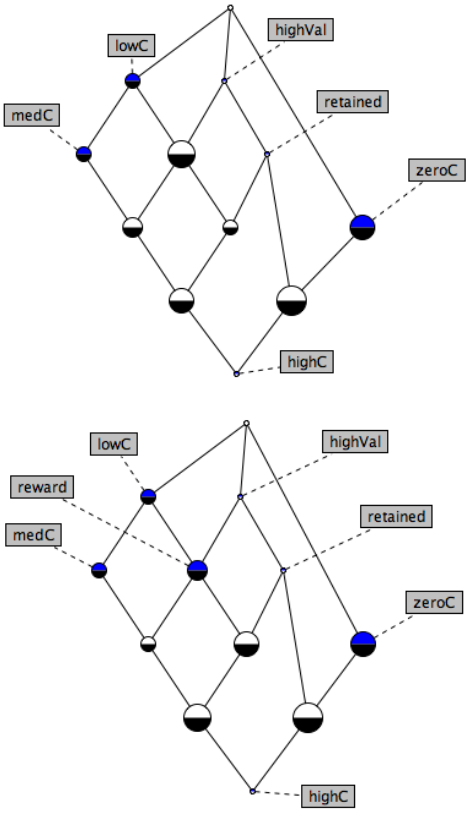

Fig. 6. Concept lattices corresponding to the processes shown in Fig 2. The key performance indicator is not shown, but improves from 40% to 70% . The similarity to Fig 5 is clear, and the suggestion to add an attribute corresponding to "reward" is obvious, once the parallel between contexts is seen.

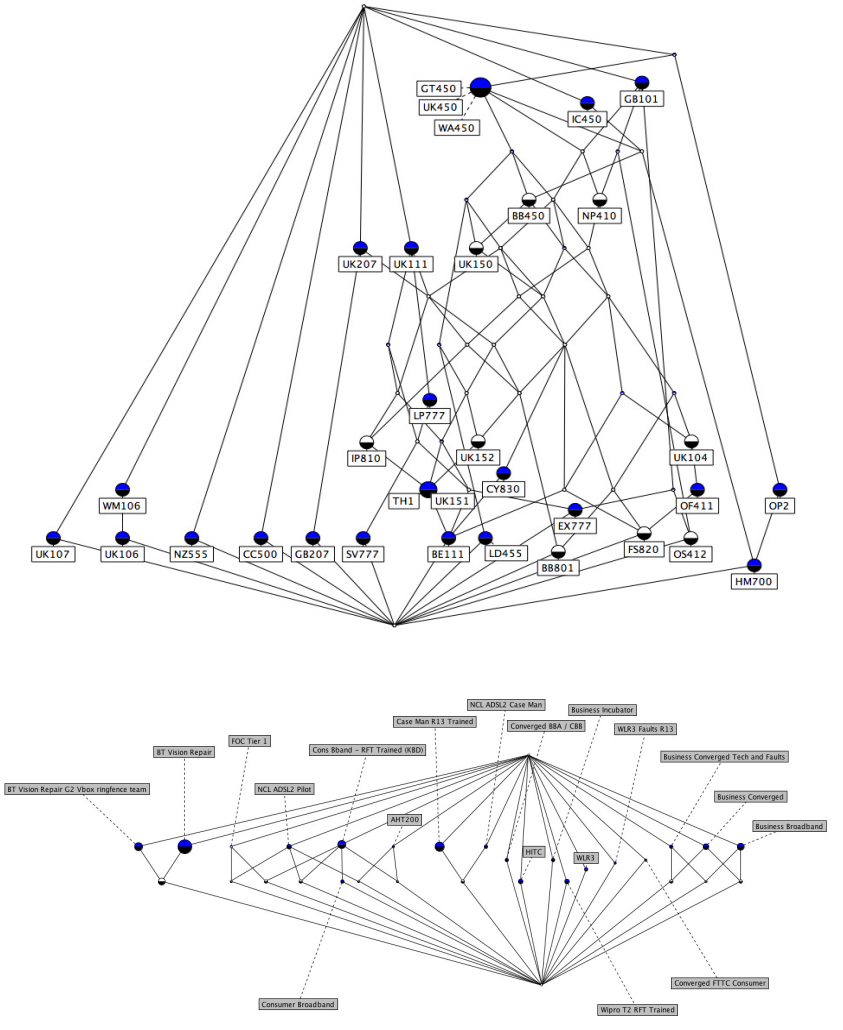

Fig. 7. Fuzzy Formal Concepts used to group different centres into domains for bisociation within dataset 1 (top) and dataset 2 (bottom)

Clearly this is a computationally intensive task, and in general it is not possible to find a consistent mapping that covers all processes. We therefore measured the goodness of a mapping by minimising

$$\frac{1}{N_{P1}}\sum_{i=1}^{N_{P1}}\frac{\min_{j}\left(d\left(f\left(P_{1i}\right),\ P_{2j},\right)\right)}{length\left(P_{1i}\right)} \tag{5}$$

where d is the edit distance between the two sequences, *length* measures the number of steps in a process instance and N_{P1} is the number of process instances in domain 1

and j indexes processes in domain 2. The value of (5) ranges between 0 (every mapped process instance in domain 1 is identical to a process instance in domain 2) and 1 (no overlap at all). Subtracting the value of (5) from 1 gives an indication of the degree of overlap.

A number of heuristics were used to guide the search for a good mapping, based on the frequencies of nodes and node pairs.

Obviously if there is an exact mapping, there is an equivalence between the process domains and the only contribution from bisociative reasoning would be to suggest that improvements in one domain might also be made in the other. In cases where there is a short distance between a process instance and its image in the target domain, bisociative reasoning might suggest process modification - for example if

$$d\left(f\left(P_{1i}\right),\ P_{2k}\right)=1$$

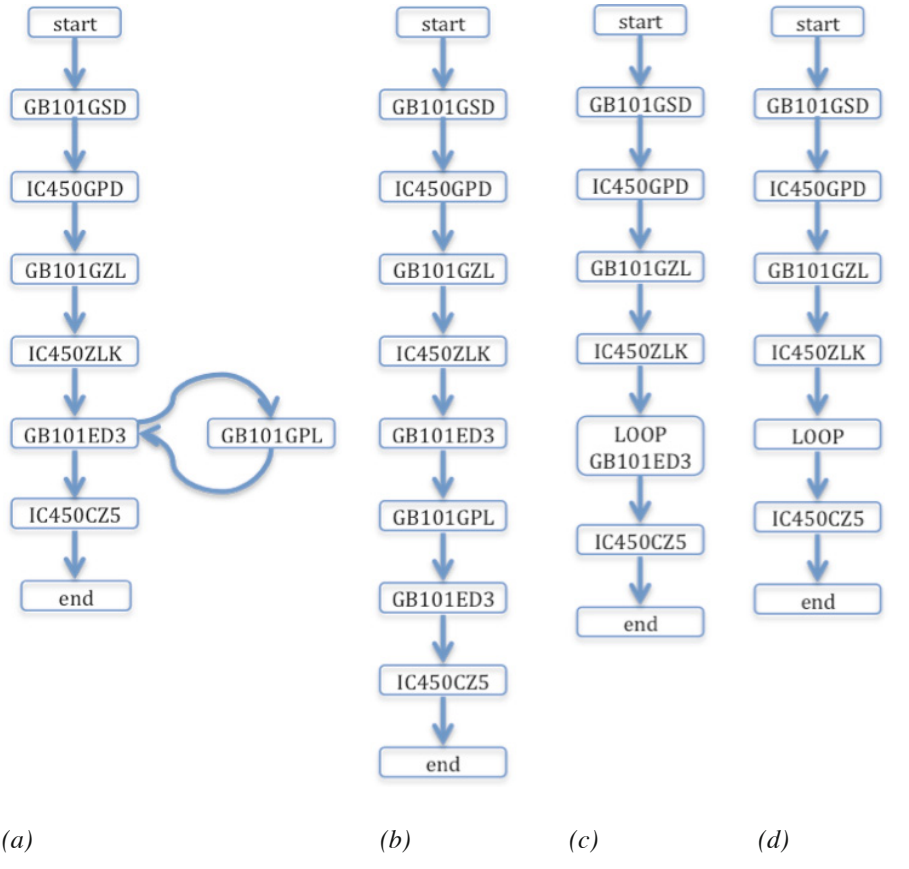

Fig. 8. Different options for treating loops in processes
(a) original process with a loop
(b) unrolled loop
(c) all loop nodes replaced by a single node, named by its start/end node
(d) all loop nodes replaced by an anonymous *loop* node

for some process P_{2k} then there is one node where the processes differ. Bisociation would suggest replacing this node by the inverse image of its counterpart in D_2. That is, if

$$P_{2k} = \left(f\left(v_{j1}\right), f\left(v_{j2}\right), \ldots, v^*_{kl}, \ \ldots\ , f\left(v_{jn}\right)\right)$$

then we should change the first process to

$$P_{1j} = \left(v_{j1}, v_{j2}, \ \ldots, f^{-1}\left(v^*_{kl}\right), \ \ldots\ , v_{jn}\right)$$

This is a limited interpretation of bisociation, and - in the cases studied here - meets with little success, not least, because of the difficulty in finding a possible *f* which gives a reasonable mapping between process domains. Examination of the most frequently occurring substitutions and substitutions applied to pairs did not lead to any significant insight.

Greater success in finding mappings occurred when anonymised loops were considered. In part this is due to the reduced size of the problem. A possible additional explanation is that there is an underlying similarity between the different process domains, and that the loops represent parts of the process that should not be carried out at all or that could be carried out independently (i.e. in parallel with the rest of the process, where this is semantically feasible). Evidence for this view arises from the observation that there is a (roughly) 50% reduction in the number of execution paths within a domain graph if we treat anonymised loops as identical irrespective of their position in the sequence.

This effect was seen in both datasets. An example of a partial mapping between the BT Vision domain and the BT Business domain (both from dataset 2) is shown in Table 5.

Another successful outcome arose from examining sequences of events in dataset 2 where domain experts had noticed anomalous event durations. In these call centre interactions, there were sequences of operations with very short duration (defined as 0 -2 seconds). This represents the time taken to ask a question and gather an answer, and is not a plausible duration - expert opinion was that it represented questions that were skipped by call centre agents, i.e. the related information was gathered at another point in the interaction.

$$D_1 = \left(V_1, E_1\right)$$
$$D_2 = \left(V_2, E_2\right)$$

We identified all sequences of more than 2 operations with short duration and then replaced each sequence with a single node indicating the sequence and whether or not the time was short or “normal”. Thus if the sequence

$$\ldots\ a\text{ - }(0)\text{ -> }b\text{ - }(1)\text{ -> }c\text{ - }(0)\text{ -> }\ldots$$

was found, then all sequences *a-b-c* were replaced by a single node *ABC-short* or *ABC-normal*. The two domains for bisociation were defined as (i) processes containing one or more nodes denoting a short sequence and (ii) processes containing

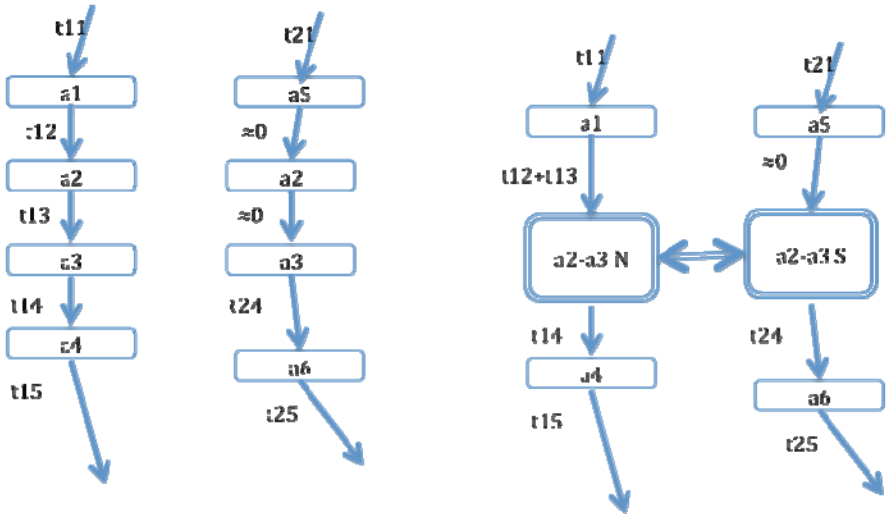

Fig. 9. Schematic illustration of bisociation between short duration sequences and normal duration sequences. The two graphs on the left both contain the sequence *-a2-a3-* , in the first case with "normal" duration and in the second with abnormally short duration. The sequences are concatenated to *a2-a3-normal (N)* and *-short (S),* and the durations of adjacent nodes are compared - see Fig. 10.

one or more nodes denoting a normal sequence. The replacement nodes were treated as bridging concepts (e.g. *ABC-short* in one domain was matched with *ABC-normal* in the second domain). Process time was examined in the joined graphs, since it was key to the bridging concepts, and we found that there was a significant increase in process step time for the immediate predecessors / successors of the bridging nodes (see Fig. 10). This suggests that although questions were skipped, the related information was gathered during preceding or succeeding questions. In turn, this means that the sequences could be moved e.g. they could be asked whilst waiting for another part of the process to complete. Such delays can happen when tests are run on the line, for instance, but further work would be required to test the feasibility of the suggestion.

The final example of bisociation was reached by comparing all of dataset 1 with all of dataset 2. Within each dataset, all processes were combined into a single large graph (with universal start-process and finish-process nodes). Based on the previous investigations, we chose as bridging concepts the loops in dataset 1 and the short-duration sequences in dataset 2. These were used to derive a mapping between nodes from domain 1 and domain 2, and the overlap in process graphs arising from the mapping was estimated by (5). Note that we used relative frequencies of process paths, since there are approximately 10 times more process instances in dataset 1 than in dataset 2. The resultant mapping between domains was deemed to be relatively high quality, since it led to high similarity between the mapped domain 1 and domain 2.

Table 5. Example of mapping between domains

domain 1 tag	domain 2 tag
g1Migration	*g2ProblemFeature*
g1EndCall	*g2EndCall*
g1Signal	*g2FindProblemDetails*
g1FindCustomerProblem	*g2ProblemFeature*
g1SystemTest	*g2SystemTest*
g1FollowKM	*g2FindProblemDetail*

Total overlap in process graphs : 56.4%

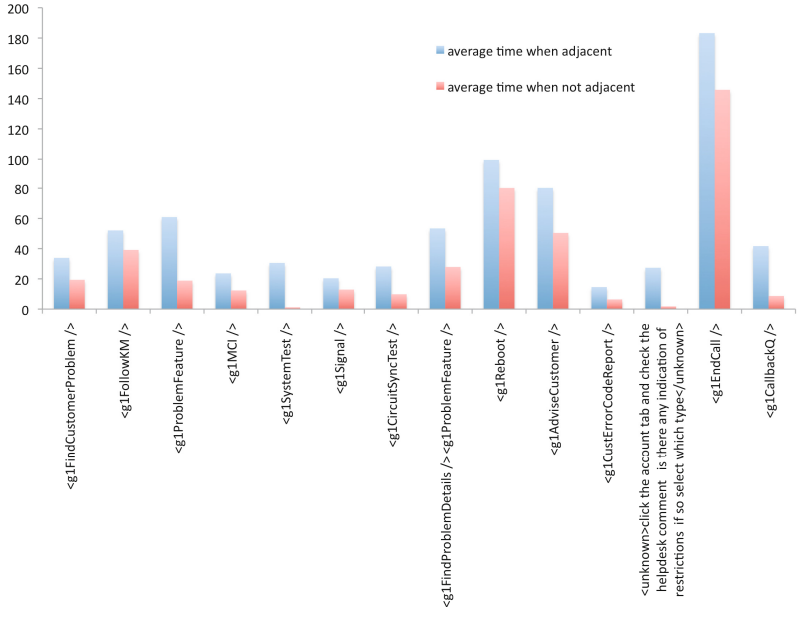

Fig. 10. Comparison of process durations for nodes adjacent to abnormally short duration sequences. The most frequent nodes are shown. Left (blue) column denotes the average duration when adjacent to an abnormal sequence, the right (red) column shows the average duration when not adjacent to an abnormal sequence. The difference may be due to additional information being gathered in adjacent nodes

5 Summary

Application of bisociation analysis to the task of creative process engineering has generated novel insight into the underlying data and into possible improvements - in particular, by suggesting parts of processes that could be performed at different points in the process sequence. The results of this study are sufficiently encouraging to warrant further investigation. Areas for future work include better presentation and visualisation of results, particularly with large data sets, the need to handle matching in edges as well as within the node structure, and issues relating to the non-static nature of process data (relevant links that may emerge and change with further data).

Acknowledgment. This work was partly funded by BT Innovate and Design and by the FP7 BISON (Bisociation Networks for Creative Information Discovery) project, number 211898

References

[1] Andersen, B., Fagerhaug, T.: Advantages and disadvantages of using predefined process models. Strategic Manufacturing: IFIP WG5 (2001)
[2] Kotter, T., Berthold, M.R.: From Information Networks to Bisociative Information Networks. In: Berthold, M.R. (ed.) Bisociative Knowledge Discovery. LNCS (LNAI), vol. 7250, pp. 33–50. Springer, Heidelberg (2012)
[3] Berthold, M.R. (ed.): Bisociative Knowledge Discovery. LNCS (LNAI), vol. 7250. Springer, Heidelberg (2012)
[4] Berthold, M.R. (ed.): Bisociative Knowledge Discovery. LNCS (LNAI), vol. 7250. Springer, Heidelberg (2012)
[5] Sherwood, D.: Koestler's Law: The Act of Discovering Creativity-And How to Apply It in Your Law Practice. Law Practice 32 (2006)
[6] Martin, T.P., Shen, Y.: Fuzzy Association Rules to Summarise Multiple Taxonomies in Large Databases. In: Laurent, A., Lesot, M.-J. (eds.) Scalable Fuzzy Algorithms for Data Management and Analysis: Methods and Design, pp. 273–301. IGI-Global (2009)
[7] Martin, T.P., Shen, Y., Azvine, B.: Incremental Evolution of Fuzzy Grammar Fragments to Enhance Instance Matching and Text Mining. IEEE Transactions on Fuzzy Systems 16, 1425–1438 (2008)
[8] Sharef, N.M., Martin, T.P.: Incremental Evolving Fuzzy Grammar for Semi-structured Text Representation. Evolving Systems (2011) (to appear)
[9] Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer (1998)
[10] Priss, U.: Formal Concept Analysis in Information Science. Annual Review of Information Science and Technology 40, 521–543 (2006)
[11] Prediger, S.: Logical Scaling in Formal Concept Analysis. In: Delugach, H.S., Keeler, M.A., Searle, L., Lukose, D., Sowa, J.F. (eds.) ICCS 1997. LNCS, vol. 1257, pp. 332–341. Springer, Heidelberg (1997)
[12] Belohlavek, R.: Fuzzy Relational Systems. Springer (2002)
[13] Belohlavek, R., Sklenar, V., Zacpal, J.: Crisply Generated Fuzzy Concepts. In: Albrecht, A.A., Jung, H., Mehlhorn, K. (eds.) Parallel Algorithms and Architectures. LNCS, vol. 269, pp. 269–284. Springer, Heidelberg (1987)
[14] Martin, T.P., Majidian, A.: Dynamic Fuzzy Concept Hierarchies (2011) (to appear)
[15] Martin, T., Shen, Y., Majidian, A.: Discovery of time-varying relations using fuzzy formal concept analysis and associations. International Journal of Intelligent Systems 25, 1217–1248 (2010)
[16] Baldwin, J.F.: The Management of Fuzzy and Probabilistic Uncertainties for Knowledge Based Systems. In: Shapiro, S.A. (ed.) Encyclopedia of AI, 2nd edn., pp. 528–537. John Wiley (1992)
[17] Sgarbas, K.N., Fakotakis, N.D., Kokkinakis, G.K.: Optimal Insertion in Deterministic DAWGs. Theoretical Computer Science 301, 103–117 (2003)