Pattern Recognition 45 (2012) 2633–2644

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

Clustering with proximity knowledge and relational knowledge

Daniel Graves [a,*], Joost Noppen [b], Witold Pedrycz [a]

[a] Department of Electrical and Computer Engineering, 9107-116th Street, University of Alberta, Edmonton, Alberta, Canada T6G 2V4
[b] Computing Department, InfoLab21, South Drive, Lancaster University, Lancaster LA1 4WA, UK

ARTICLE INFO

Article history:
Received 23 March 2011
Received in revised form
18 November 2011
Accepted 20 December 2011
Available online 4 January 2012

Keywords:
Relational clustering
Fuzzy clustering
Proximity
Knowledge representation
Software requirements

ABSTRACT

In this article, a proximity fuzzy framework for clustering relational data is presented, where the relationships between the entities of the data are given in terms of proximity values. We offer a comprehensive and in-depth comparison of our clustering framework with proximity relational knowledge to clustering with distance relational knowledge, such as the well known relational Fuzzy C-Means (FCM). We conclude that proximity can provide a richer description of the relationships among the data and this offers a significant advantage when realizing clustering. We further motivate clustering relational proximity data and provide both synthetic and real-world experiments to demonstrate both the usefulness and advantage offered by clustering proximity data. Finally, a case study of relational clustering is introduced where we apply proximity fuzzy clustering to the problem of clustering a set of trees derived from software requirements engineering. The relationships between trees are based on the degree of closeness in both the location of the nodes in the trees and the semantics associated with the type of connections between the nodes.

1. Introduction

Relational clustering problems are increasingly encountered in a number of different applications, cf. [1–13]. It is common to identify clustering problems where data are vectors positioned in a d-dimensional feature space $X=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N\}\subset\Re^d$. An area that is gaining interest is relational clustering, where the entities (i.e., objects) to be clustered are not vectors themselves but rather are described in terms of relational knowledge between objects. Following similar notation as presented in [2], we denote these objects by $X=\{O_1,O_2,\ \ldots,\ O_N\}$, which may or may not have a numerical representation in a feature space. Hathaway, et al., introduced relational knowledge coming in the form of a fuzzy binary relation called $\rho(\mathbf{x}_i,\mathbf{x}_j)$, i.e. $\rho: X\times X\mapsto[0,1]$. There are many interesting applications of relational clustering as well, cf. [2,4–6,8,14–20]. Clustering with proximity information, however, offers a number of benefits over clustering with other forms of relational knowledge including the more common distance-based knowledge. While there are many forms of relational knowledge to choose from, it is important to understand the advantages and challenges of each form of domain knowledge to select the best one for a given problem. This work is largely motivated to that end to gain a better understanding of the advantages and disadvantages of proximity and distance knowledge in the context of clustering.

Let us consider an illustrative example to highlight the essence of the study. Suppose we have a data set with three clusters as shown in Fig. 1.

In order to cluster this data set with commonly encountered algorithms such as Fuzzy C-Means (FCM) or k-means, we would need to choose a suitable distance measure (i.e. a way to measure the relationship between the objects (points in this example)). A simple measure that is most often used is the Euclidean distance. Given the rather complicated geometry of this data set, it is not surprising that FCM equipped with the Euclidean distance is unable to discover a structure of data, see Fig. 2.

The boundaries of the clusters are shown in Fig. 2 (note that those are determined by finding all points where the membership values of the two neighboring clusters are equal). Let us express the relationship between these objects using proximity rather than distance, where proximity describes the degree of closeness of the objects. In the example, these objects are points on a two dimensional grid; therefore, we can quantify the degree of closeness using a Gaussian membership function (see Table 1 for the formula). The width parameter σ^2 offers some flexibility of the construct. When using a proximity-based relational clustering algorithm to cluster the data, its "real" structure has been revealed as can be seen in Fig. 3.

When $\sigma^2=0.7$, we obtain two spherical clusters and a third cluster that is formed by the rest of the data. The flexibility offered by the width parameter is visualized in Fig. 3(c)–(e): when the values of σ^2 increase, the two spherical clusters become larger and "over-emphasized." The other values of sigma show the flexilibity offered by adjusting sigma. In Fig. 3(c)–(e), the two spherical clusters become larger and over-emphasized as sigma increases. As a result, they

* Corresponding author. Tel.: +1 780 461 8702; fax: +1 780 492 1811.
E-mail addresses: dgraves@ualberta.ca, dangraves77@gmail.com (D. Graves), j.noppen@uea.ac.uk (J. Noppen), wpedrycz@ualberta.ca (W. Pedrycz).

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.12.019

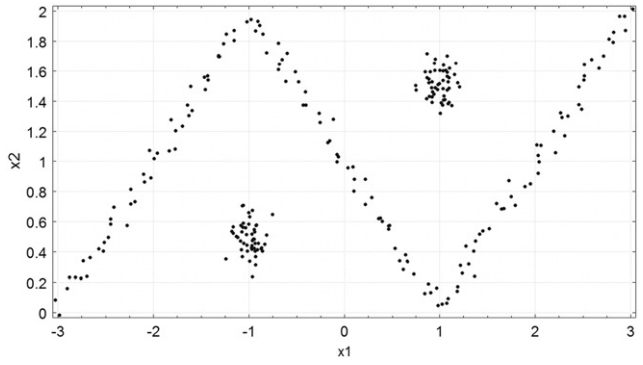

Fig. 1. An illustrative example.

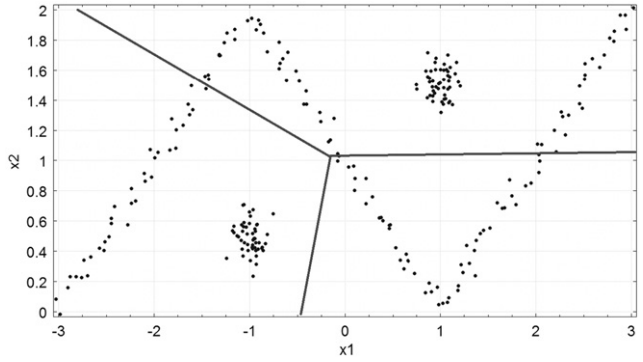

Fig. 2. Boundaries of FCM-constructed clusters.

Table 1
Common proximity functions.

Name	*Proximity function*	*Parameters*
Gaussian	$p(\mathbf{x},\mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|^2/\sigma^2)$	$\sigma^2 > 0$
Cosine	$p(\mathbf{x},\mathbf{y}) = \frac{1}{2}\left[\frac{\mathbf{x}^T\mathbf{y}}{\sqrt{(\mathbf{x}^T\mathbf{x})(\mathbf{y}^T\mathbf{y})}}\right] - \frac{1}{2}$	
Polynomial	$p(\mathbf{x},\mathbf{y}) = \frac{1}{2}\left[\frac{(\mathbf{x}^T\mathbf{y}+\theta)^p}{\sqrt{(\mathbf{x}^T\mathbf{x}+\theta)^p(\mathbf{y}^T\mathbf{y}+\theta)^p}}\right] - \frac{1}{2}$	$\theta \geq 0, p \in$ naturals;
ANOVA	$p(\mathbf{x},\mathbf{y}) = \frac{1}{n}\sum_{k=1}^{n} \exp\left(-\frac{1}{\sigma^2}\|\mathbf{x}^k-\mathbf{y}^k\|^{2p}\right)$	$\sigma \in \Re, p > 0, n \in$ naturals;

include part of the zig-zag pattern. Therefore, choosing proximity knowledge instead of the Euclidean distance (which is quite inflexible) has definite advantages including:

- semantics arising from graded similarity, e.g. "close" and "far," and
- flexibility.

This flexibility arises from the relationship of proximity functions to kernel functions since a wide range of kernel functions [27] can be employed to quantify proximity. This relationship is well known, cf. [28], where the authors show that clustering based on the adjacency matrix is equivalent to kernel-based clustering. The adjacency matrix is found in spectral clustering literature [9,29] where it is used to find a graph cut. It should be noted that the definition of adjacency is very similar to but not identical to our definition of proximity.

The problem of selecting a form of relational knowledge becomes even more complicated as the available information may originate from a number of different sources [2]:

- obtained from a human expert, and
- computed based on measures of distance, similarity, or proximity.

Since we have to adhere to the properties of distance, similarity or proximity, requesting accurate relational knowledge from a human expert can become a problem. This is because the properties of, for example, distance must obey the triangular inequality, which cannot always be easily guaranteed. Recently, proximity functions have become popular in deriving relational information for use in clustering [4–6]. Roughly speaking, proximity captures the degree of resemblance between objects and can be described in a form of a binary fuzzy relation quantifying the notion of "closeness" or resemblance (as similarity described in [2]). Distance, on the other hand, is a measure of the separation of objects. The formal definition of proximity relational information is that it must satisfy the properties of identity ($p(\mathbf{x},\mathbf{x})=1$) and symmetry $p(\mathbf{x},\mathbf{y})=p(\mathbf{y},\mathbf{x})$. Proximity does not require transitivity. In contrast, distance must satisfy the triangular inequality; therefore, proximity can be more easily obtained from a human expert. In addition, the properties of proximity are interesting from a research perspective as they offer a number of advantages over the more common forms of relational knowledge due to the semantics of "closeness." However, there are a number of open problems including how to select the best form of domain knowledge for a given problem and how to determine the most suitable relational measure of that knowledge, e.g. proximity function, distance function, or similarity function. In particular limited work has been done on quantifying the benefits of the different forms of relational knowledge and their impact on relational clustering frameworks.

To illustrate some of the current issues arising from relational knowledge, consider the problems of clustering a collection of trees such as those discussed in [21,22]. An example of a set of trees is depicted in Fig. 4 where the nodes are labeled by lower case letters. A serious concern is the lack of a numerical representation of this problem in a feature space. These trees are entities, which are much easier to capture by relational knowledge for the purposes of clustering.

One way to describe these trees relationally is to use the edit distance, cf. [23–26]. Note that there are polynomial time algorithms that have been devised [6] to determine this. Using this relational knowledge, one can cluster the trees shown in Fig. 4 using relational FCM or a similar relational clustering algorithm. We may also choose to represent the relational information using proximity, which is produced by a proximity function. One of the benefits of proximity knowledge is that there is semantics associated with the relational information, i.e. "close" (proximity values approaching 1) and "far" (proximity values approaching 0). With many different forms of relational knowledge, a question arises on how does one choose the most suitable form of knowledge for a given problem. One of the goals of this study is to characterize the merits of using either proximity or distance to aid in this decision making.

To summarize, the main objectives of our study are to

- introduce and discuss the advantages and disadvantages of distance and proximity relational knowledge, and
- compare two clustering frameworks that make use of distance and proximity relational knowledge to better understand their differences and benefits.

The novelty introduced in this study stems from:

1. an in-depth presentation of the new proximity fuzzy clustering framework only briefly introduced in [7],

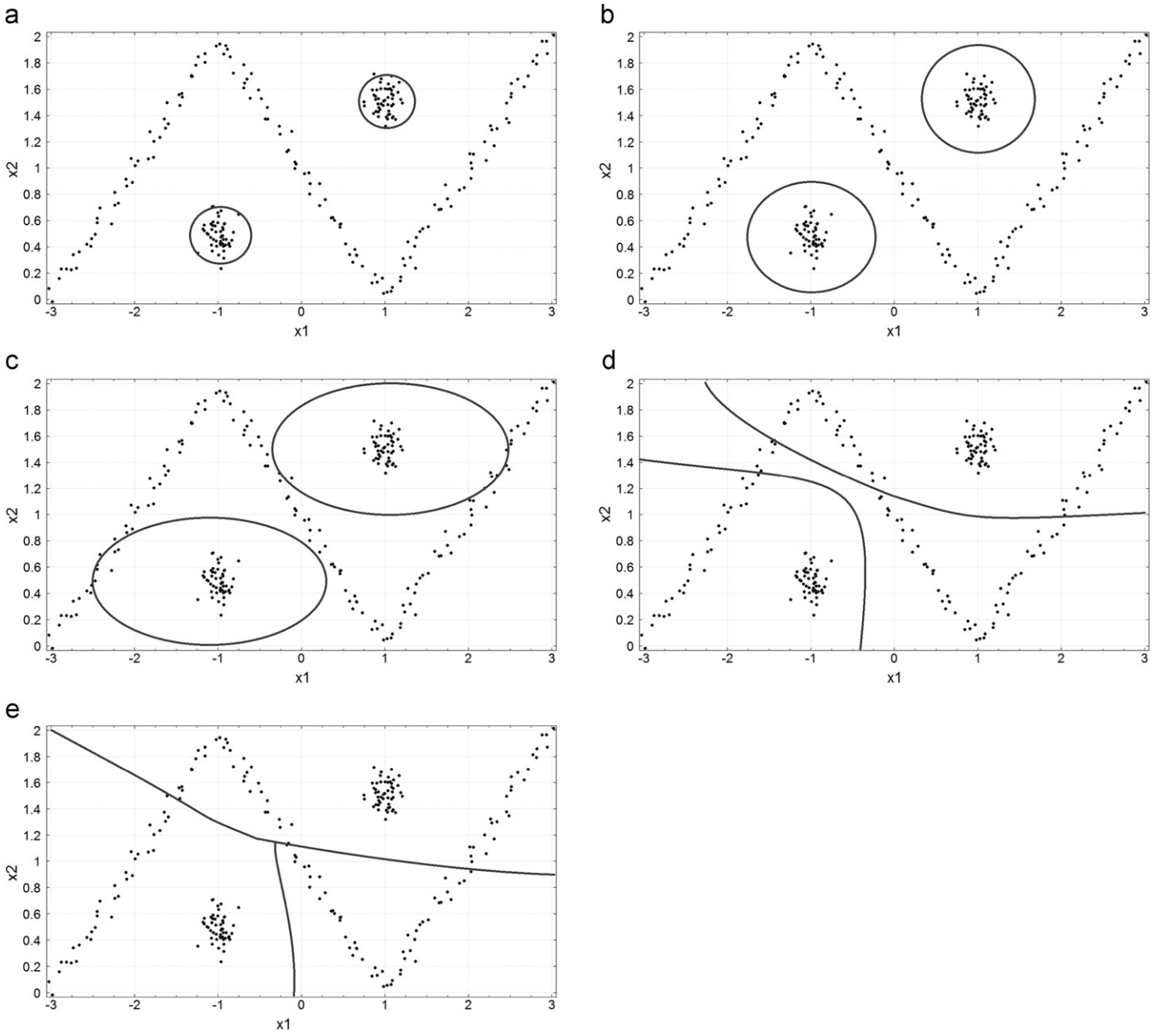

Fig. 3. Proximity cluster boundaries when using proximity knowledge (a) σ^2=0.1 (b) σ^2=0.7 (c) σ^2=1.0 (d) σ^2=2.0 (e) σ^2=10.0.

2. a comparison of proximity relational fuzzy clustering with relational FCM in order to gage the benefits of using proximity knowledge, and
3. a case study of relational clustering applied to software requirements engineering.

We also introduce a method of selecting the number of clusters with proximity fuzzy clustering. We apply this approach to the problem of clustering a set of trees that capture software requirements where the number of clusters is unknown.

1.1. Definitions

The notation used in this study adheres to the one commonly used in fuzzy clustering. A data set X is a set of objects O_k, $X=\{O_1, O_2, \ldots, O_M\}$ where M is the number of objects in the data set. The relational data is given by the matrix $R=[r_{ij}]$ for $i,j=1, \ldots, M$, where r_{ij} is a measure of the relationship (such as resemblance) between objects O_i and O_j. In this study, we consider the relationships that are expressed in terms of distance and proximity. The distance between two objects O_1, $O_2 \in X$ is given by $d(O_1, O_2)$ while the proximity is denoted by $p(O_1, O_2)$. The result of the clustering is the partition matrix $U=[u_{ik}]$, $i=1, \ldots, c$, and $k=1, \ldots, M$, where c is the number of clusters where $\sum_{i=1}^{c} u_{ik}=1$. The matrix U captures the degree of membership of object O_k in the ith cluster. The proximity values produced by proximity fuzzy clustering is expressed by the matrix $Q=[q_{ik}]$ $i=1, \ldots, c$, and $k=1, \ldots, M$. The matrix Q represents the proximity of object O_k to the prototype object, or most representative object, in the ith cluster. The Euclidean norm is denoted by $\|\mathbf{x}\|$, where $\mathbf{x}$ stands for any real-valued vector. We describe the cardinality using the notation $M=|X|$. The symbols $\wedge$ and $\vee$ are the Boolean operators AND and OR, respectively.

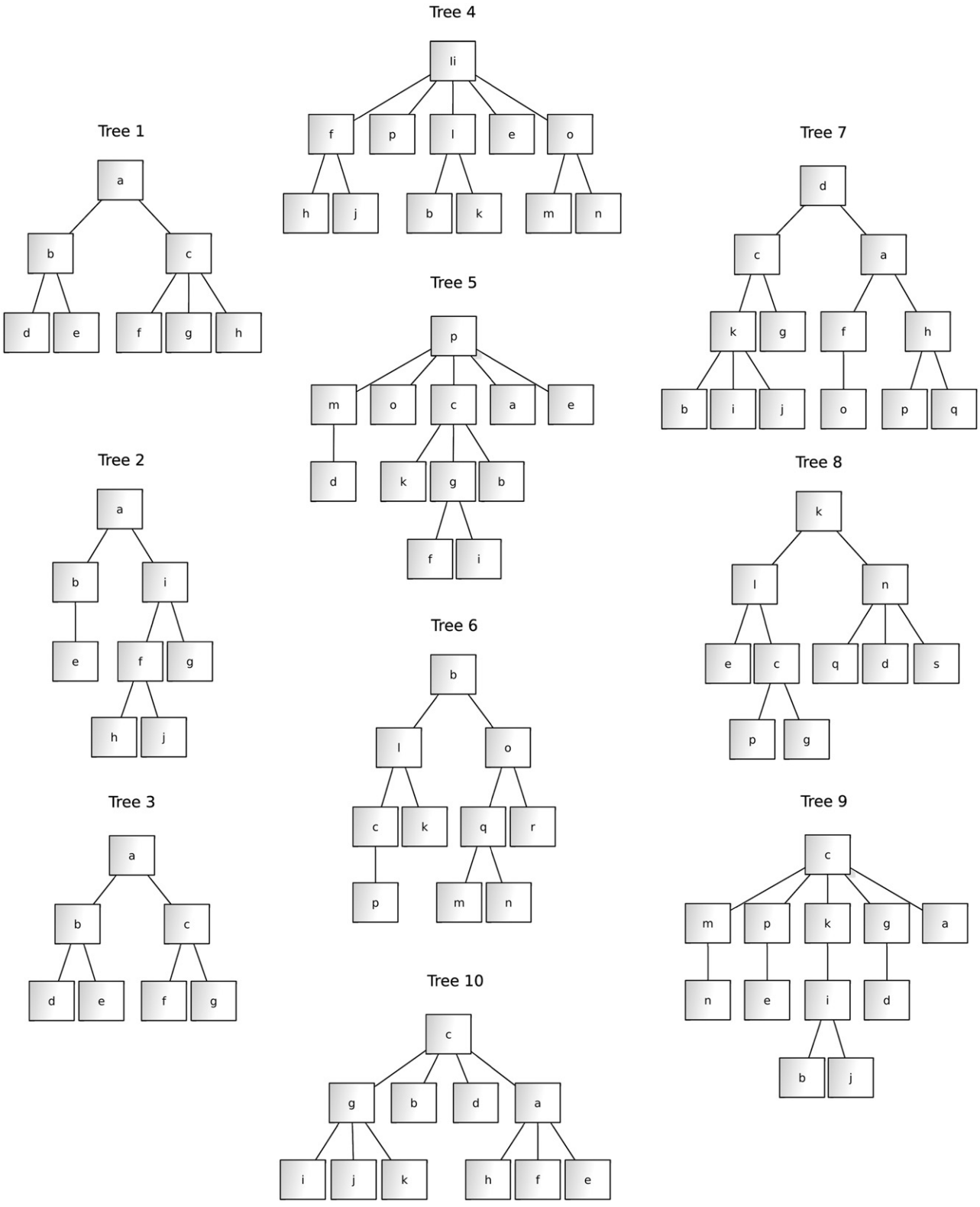

Fig. 4. An example of a collection of trees.

2. Relational fuzzy clustering

2.1. Relational fuzzy C-means (FCM)

Relational Fuzzy C-Means (relational FCM) produces a fuzzy partition when being provided with some relational data, cf. [2,3]. Relational FCM accepts a matrix of relationships between objects as its input $R=[r_{jk}]$, $j,k=1, \ldots, M$. It has been shown in [3] that relational FCM minimizes the same objective function as used in FCM, i.e.

$$Obj = \sum_{k=1}^{M} \sum_{i=1}^{c} u_{ik}^{m} \|\mathbf{x}_k - \mathbf{v}_i\|^2 \tag{1}$$

where $\mathbf{x}_k$ is a vector of attributes that describe the object O_k, $k=1, \ldots, M$ and $\mathbf{v}_i$ is the prototype, $i=1, \ldots, c$. The fuzziness of the partition matrix is controlled by the fuzzification coefficient $m>1$, where smaller values of m result in membership values closer to the extremes of 0 and 1. The distance $\|\mathbf{x}_k-\mathbf{v}_i\|^2$ can be reformulated and expressed entirely in terms of relational distance knowledge R. More specifically by Theorem 2 of [2], it is shown that this distance can be expressed as

$$\|\mathbf{x}_k-\mathbf{v}_i\|^2=(\mathrm{R}\mathbf{w}_i)-\tfrac{1}{2}\mathbf{w}_i^{\mathrm{T}}\mathrm{R}\mathbf{w}_i \tag{2}$$

where the relations are assumed to be

$$r_{jk}=\|\mathbf{x}_j-\mathbf{x}_k\|^2 \tag{3}$$

$j,k=1, \ldots, M$. The values for $\mathbf{w}_i$ are determined by

$$\mathbf{w}_i=\frac{[u_{i1}^m,u_{i2}^m,\ldots,u_{iM}^m]^{\mathrm{T}}}{\sum_{k=1}^{N_{size}}u_{ik}^m} \tag{4}$$

$i=1, \ldots, c$. Finally, the partition matrix is computed in the same way as in standard FCM, i.e.

$$u_{ik}=\frac{1}{\sum_{j=1}^{c}((\mathrm{R}\mathbf{w}_i)-(1/2)\mathbf{w}_i^{\mathrm{T}}\mathrm{R}\mathbf{w}_i/(\mathrm{R}\mathbf{w}_i)-(1/2)\mathbf{w}_i^{\mathrm{T}}\mathrm{R}\mathbf{w}_i)} \tag{5}$$

$i=1, \ldots, c$ and $k=1, \ldots, M$. The algorithm starts with an initial random partition matrix U that satisfies the standard well-known FCM constraint (*i.e.* $\sum_{i=1}^{c}u_{ik}=1$). The algorithm then iterates over expressions (2), (4) and (5) until a stopping criterion has been met. The typical stopping criterion is when the Frobenius norm of the difference between the partition matrix in the current iteration $U(t)$ and the partition matrix in the previous iteration $U(t-1)$ is less than a small positive threshold $\varepsilon>0$, i.e. $\|U(t)-U(t-1)\|_F<\varepsilon$.

It should be noted that relational FCM suffers from a few well-known drawbacks given its roots in FCM, cf. [3]. Particularly in relational FCM we make the assumption that the relational data satisfy the well-known properties of distance. Since the relational data may be obtained from a number of different sources mentioned earlier, these basic properties (such as non-negativity) may not be satisfied. In these cases relational FCM fails. Fortunately, in [3] the authors alleviate this problem by developing the Non-Euclidean Relational Fuzzy (NERF) c-means clustering algorithm, which forces the non-negativity property (7) to be true by introducing a scaling factor $\beta\geq 0$.

2.2. Relational knowledge

Relational knowledge coming in the form distance as described in (3) must satisfy the following well-known properties of distance [3]:

$$d(O_k,O_k)=0, \forall, \quad k=1,\ldots,M \text{ (identity)} \tag{6}$$

$$d(O_j,O_k)\geq 0, \forall j, \quad k=1,\ldots,M \text{ (non-negativity)} \tag{7}$$

$$d(O_j,O_k)=d(O_k,O_j), \forall j, \quad k=1\ldots M \text{ (symmetry)} \tag{8}$$

$$d(O_j,O_k)\leq d(O_j,O_i)+d(O_k,O_i), \forall i,j, \quad k=1\ldots M \text{ (triangular inequality)} \tag{9}$$

Note then that the assumption given in (3) enforces several constraints on the relational knowledge R. However, there are two problems with this form of knowledge:

1. distance values lack semantics, e.g. "close" or "far," and
2. distance is not bounded by a maximum value (and therefore sensitive to e.g. outliers).

We describe these points in further detail. To illustrate the first point, consider that distance often has an associated unit of measure. For example, one may measure physical distances by meters or feet. However, whether two objects are "close" or "far" depends on domain knowledge of what is being measured rather than the number of measureable units. Suppose the edit distance between Trees 1 and 2 in Fig. 4 is 4. Are they close or far? It should be noted that this is unimportant when we have domain knowledge of the problem such as information on the relative distances or expert knowledge on the meaning of distance and closeness in a given problem. This presents a problem because one must be an expert in a given problem to understand the meaning of the degree of closeness of a given distance value. However, without further knowledge on the relative distances of the other trees and some additional domain knowledge, there is no way to determine the answer to the question.

One way to address this is to scale the distance with a multiplicative factor to normalize the values between 0 and 1. The problem is that some distance values may be large (e.g. for outlier objects) thus having a drastic effect on the scaled distance values. This leads to the second point mentioned above. In some problems such as the graph clustering problem shown in Fig. 4, the maximum edit distance possible is where the two trees do not have any nodes in common. The distance values cannot be greater than this maximum value in the tree clustering problem and numerous other similar problems. On the other hand, for some problems the distance may not be bounded by a maximum. In this case, distance is sensitive to the presence of outliers, which will be extra-proportionally separated from much of the data set.

3. Proximity relational fuzzy clustering

3.1. Proximity knowledge

Informally, proximity is the degree of closeness (resemblance) between objects or entites. Formally, a proximity function $p(\cdot,\cdot)$ must satisfy the following properties:

$$p(O_k,O_k)=1, \forall k=1\ldots M \text{ (identity)} \tag{10}$$

$$p(O_j,O_k)=p(O_k,O_j), \forall j,k=1\ldots M \text{ (symmetry)} \tag{11}$$

$$0\leq p(O_j,O_k)\leq 1, \forall j,k=1,\ldots,M \tag{12}$$

In this definition proximity resembles membership grades captured by membership functions. This type of relational knowledge is a fuzzy binary relation (as described in [2]) and as such encompases semantics. For example suppose Trees 1 and 2 have been evaluated to have a proximity of 0.8, then one can conclude that these trees are in fact "close" without knowing the proximity to the other trees in the data set. The semantics (domain knowledge) provide a richer description of the relationships between objects. A list of some proximity functions for data in the feature space $\mathbf{x},\mathbf{y}\in\Re^d$ are given in Table 1.

The correlational function is a proximity function used in [7] for the clustering of time series in the creation of a fuzzy rule-based time series prediction model. Note that some proximity functions have adjustable parameters, which can be tuned (optimized) to reflect the specificity of the problem.

3.2. Clustering with proximity knowledge

The data is clustered into c clusters by minimizing the distance between pattern proximity values and prototype proximity values. More specifically, the optimization occurs under the

objective function

$$Obj = \sum_{i=1}^{c}\sum_{k=1}^{M} u_{ik}^{m} \sum_{j=1}^{M} (p(O_k,O_j)-q_{ij})^2 \quad (13)$$

where $Q=[q_{ij}]$ is the prototype proximity of object j to an unspecified prototype $\mathbf{v}_i$, i.e. $p(\mathbf{x}_j,\mathbf{v}_i)$ for $i=1, \ldots, c$ and $j=1, \ldots, M$. The expression is subject to the well-known FCM constraint (i.e. $\sum_{i=1}^{c} u_{ik} = 1$). The membership degrees are organized in the form of a partition matrix $U=[u_{ik}]$ for $i=1, \ldots, c$ and $k=1, \ldots, M$ and the fuzzification coefficient is m. The objective of (13) is to determine a partition U of the graphs such that the proximity of each graph O_k to all other graphs is closest to the proximity of the ith cluster prototype to all other graphs. It should be emphasized that q_{ij} is not an explicit prototype of the data unlike the one encountered in FCM or K-Means. The benefit offered by representing the cluster proximity Q by $q=p(\mathbf{x}_j,\mathbf{v}_i)$ for $i=1, \ldots, c$ and $j=1, \ldots, M$ is that when $X \subset \Re^d$ and the proximity function is also a kernel function (i.e. that satisfies Mercer's condition [20,27,30] as is the case with the proximity functions in Table 1), then the unspecified prototypes are implied in a higher dimensional space through the inner product [22]. As a result, in this special case some very interesting non-spherical cluster geometries are possible through the choice of the proximity function, cf. [31].

Using Lagrange multipliers to accommodate the "standard" FCM constraints, we can find a set of expressions that minimize (14) by setting the partial derivatives to zero with respect to u_{ik} and q_{ij} and solving for u_{ik} and q_{ij}. The expression for the cluster proximity is given by

$$q_{ij} = \frac{\sum_{k=1}^{M} u_{ik}^{m} p(O_k,O_j)}{\sum_{k=1}^{M} u_{ik}^{m}} \quad (14)$$

$i=1, \ldots, c$ and $j=1, \ldots, M$. Similarly an expression is obtained for the membership values.

$$u_{ik} = \frac{1}{\sum_{h=1}^{c} (\sum_{j=1}^{M} (p(O_k,O_j)-q_{ij})^2)/(\sum_{j=1}^{M} (p(O_k,O_j)-q_{hj})^2)^{1/(m-1)}} \quad (15)$$

$i=1, \ldots, c$ and $k=1, \ldots, M$. The algorithm starts with an initial random partition matrix U that satisfies the standard well known FCM constraints (*i.e.* $\sum_{i=1}^{c} u_{ik} = 1$). The algorithm then iteratively loops over expressions (14) and (15) until a stopping criterion has been met.

3.3. Determining the number of clusters and the fuzzification coefficient

An open problem is the choice of the number of clusters "c" and the value of the fuzzification coefficient "m." Much work has been done in this area, but the approaches highly vary in both design and performance. Proximity fuzzy clustering offers a simple and effective (unsupervised) performance index by making use of the proximity relations directly. It has been shown in [6] that an estimation of the proximity relations can be formed from the graded membership. The difference between the proximity relations and the estimated proximity relations $\tilde{p}_{jk}$ is called the reconstruction error. The reconstruction error measures the quality of the clustering. However, unlike its use in [6], we propose using the reconstruction error to estimate both the number of clusters and the fuzzification coefficient. The reconstruction error given by

$$r_{error} = \frac{1}{M^2}\sum_{k=1}^{M}\sum_{j=1}^{M} (p(O_j,O_k)-\tilde{p}_{jk})^2 \quad (16)$$

where the reconstructed proximity is based on membership values given post-mortem clustering, cf. [6], via the expression

$$\tilde{p}_{jk} = \sum_{i=1}^{c} \min(u_{ij},u_{ik}) \quad (17)$$

It is noted that we need to ensure that two conditions are satisfied in order to improve the performance.

- No two cluster proximities are the same, i.e. $q_{ij} \neq q_{lj}$ for $i,l=1, \ldots, c, i \neq l$.
- No cluster is "empty" where the ith cluster is empty if and only if $\forall k = 1 \ldots M, i \neq \arg\max_{\forall j = 1 \ldots c} u_{jk}$.

Satisfying these conditions will avoid special cases that usually occur when the number of clusters is very high since a cluster may be empty or the prototypes may be identical in two or more of the clusters.

3.4. Example

We demonstrate the performance of both proximity and relational FCM [2,3] on the synthetic example given in Fig. 4 and compare the results with those obtained when running the relational FCM. In some cases it is not easy to determine a suitable proximity function such as one of those shown in Table 1. In this example, we can calculate the edit distance, which is well-studied in a number of graph problems, cf. [23–25]. We transform this distance knowledge into proximity using the expression

$$p(O_j,O_k) = 1 - \frac{d(O_j,O_k)}{\max_{\forall i,h = 1 \ldots M} (d(O_i,O_h))} \quad (18)$$

where $O_j,O_k \in X$ are objects or entities in the problem. The proximity values are given in Table 2.

Table 2
Proximity values of the synthetic graph data set given in Fig. 3.

Trees	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	1	0.83	1	0.4	0.57	0.39	0.46	0.29	0.64	0.67
2	–	1	0.85	0.46	0.36	0.27	0.36	0.17	0.5	0.46
3	–	–	1	0.31	0.54	0.39	0.46	0.29	0.64	0.53
4	–	–	–	1	0.42	0.71	0.33	0.46	0.46	0.43
5	–	–	–	–	1	0.5	0.77	0.43	0.86	0.8
6	–	–	–	–	–	1	0.46	0.57	0.43	0.44
7	–	–	–	–	–	–	1	0.31	0.77	0.57
8	–	–	–	–	–	–	–	1	0.5	0.4
9	–	–	–	–	–	–	–	–	1	0.8
10	–	–	–	–	–	–	–	–	–	1

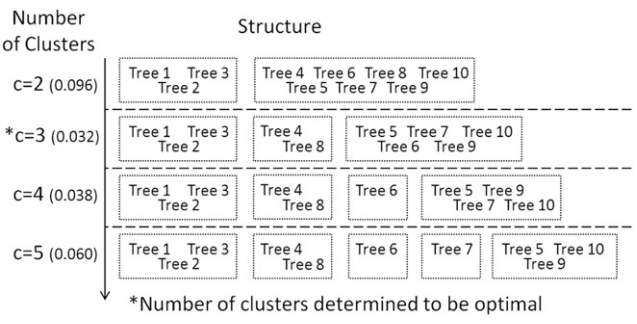

Fig. 5. Structure of the synthetic example with respect to the number of clusters c.

Proximity fuzzy clustering detects the number of clusters to be $c=3$ and the fuzzification coefficient to be $m=2.9$ under the same optimization vehicle discussed in the previous section. The structure discovered by proximity fuzzy clustering is given in the form: {Tree 1, Tree 2, Tree 3}, {Tree 4, Tree 6, Tree 8}, and {Tree 5, Tree 7, Tree 9, Tree 10}. The membership values for Trees 7, 8 and 10 are quite low since the structure of Tree 8 is not very similar to any tree and the structure of Trees 7 and 10 are not similar. Fig. 5 demonstrates how the structure of the clusters changes with the number of clusters (m is held at $m=2.9$) for proximity fuzzy clustering. The structure shown in Fig. 5 is determined by labeling the tree O_k according to its maximum membership, i.e. $cluster-label(O_k)=\arg\max_{\forall i=1..c} u_{ik}$ for $k=1, \ldots, M$.

There is an interesting trend shown in Fig. 5, particularly the consistency of the first cluster. In fact the membership values of Trees 1, 2 and 3 are all close to 1, which is intuitive since the trees are structurally quite similar. The membership of Trees 5 and 9 are fairly high as well. Otherwise the membership values are much lower. The structure of the clusters is the same as obtained when running relational FCM. The membership value of Tree 8 however is very low thus it does not belong strongly to any cluster. The reconstruction errors are provided in parentheses and indicate a minimum at $c=3$. The reconstruction error at $c=4$ is only slightly larger. Thus the reconstruction error supports a number of clusters that is 3 or possibly 4.

The plots given in Fig. 6 indicate the membership values for $c=3$ on a floating bar plot. The horizontal axis is the Tree 1–10 from the problem where the vertical bars indicate the membership values. The membership values of the clusters are stacked in a bar graph to visually indicate the degree of membership for that particular tree in each cluster. Each cluster is coded using different shades of gray: black, gray and light gray.

For example, we can see that the membership of Trees 1, 2 and 3 are quite high (close to 1) for cluster 1. This is intuitive since Fig. 4 shows that these trees have a lot of structural similarity. The membership values are mostly the same for both proximity and relational FCM when $c=3$.

The objective function calculated at each iteration is shown in Fig. 7 for proximity and relational FCM.

Relational FCM appears to have converged slightly faster than proximity fuzzy clustering on this data set. In summary, we have demonstrated how both proximity fuzzy clustering and relational FCM can be used to cluster a simple tree example. Particularly, the structure formed by the clusters indicates the trees that are most similar, i.e., Trees 1, 2 and 3. The other trees are more distinct from each other and as a result their membership values are lower.

4. Experimental results

In this section, we evaluate the clustering produced by relational FCM and proximity relational fuzzy clustering on several synthetic and real-world relational data. The performance indices used are described in the Section 4.1 while the results for the synthetic data and real-world data are given in Sections 4.2 and 4.3, respectively. Finally we discuss the performance of clustering in Section 4.3.

4.1. Evaluation criteria

The evaluation criteria used here are the commonly used classification rate (CR), and the reconstruction error (RE). Classification rate is an evaluation measure that extracts a partition from the grades of membership (U) by assigning each pattern to exactly one cluster. The assignment is done by selecting the cluster with the maximum membership grade for each pattern as done in the previous example, i.e. $cluster-label(O_k)=\arg\max_{\forall i=1..c} u_{ik}$. Despite the fact that clustering is unsupervised and thus there is no way to label the clusters with class labels, the classification rate is a common way to compare the quality of partition produced by a clustering algorithm and a ground truth (or known) partition. The classification rate is computed by labeling the patterns in a cluster with the class label that appears most frequently in that cluster. Therefore, the classification rate is a measure of the disagreement between the partition produced by clustering and ground truth partition.

The second performance measure is the reconstruction error, which considers the graded membership values directly. The reconstruction error is computed for proximity fuzzy clustering using expression (17), which is a post-mortem clustering performance index. Since it is not possible to compute this reconstruction error for relational FCM, we use the reconstruction error simply for post-mortem evaluation of proximity fuzzy clustering. The reconstruction error is not used to estimate the number of clusters or the value of m in this section in order to better understand the differences between the two algorithms. However, the reconstruction error will be used to determine the number of clusters and the value of m in Section 5. Note that the reconstruction error does not use any class information to evaluate the performance and is therefore an unsupervised evaluation of proximity fuzzy clustering where lower values of the reconstruction error mean better clustering performance.

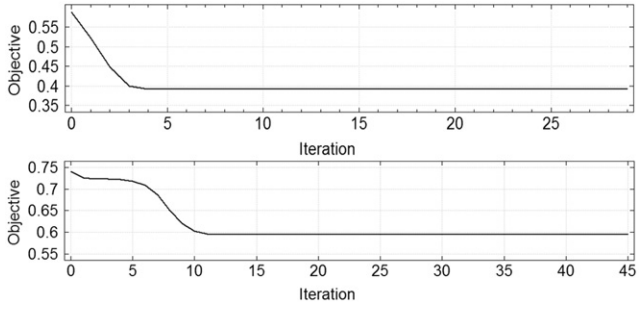

Fig. 7. Convergence of the clustering proximity fuzzy clustering (top) and relational FCM (bottom) objective functions.

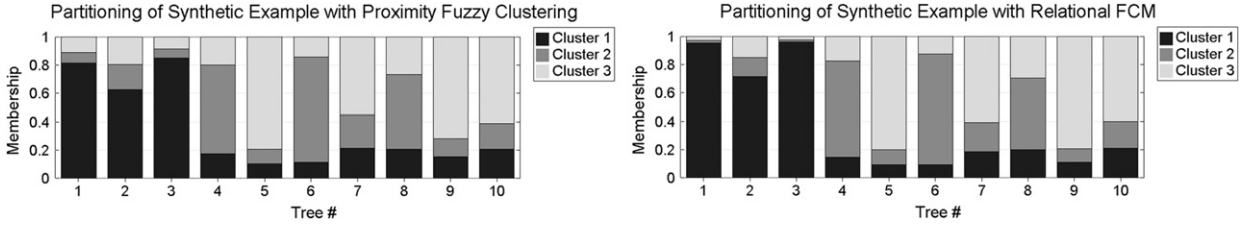

Fig. 6. Membership grades of the synthetic example ($c=3$).

4.2. Synthetic relational data

The performance of relational FCM and proximity relational fuzzy clustering is evaluated on several synthetic relational data sets.

The first synthetic data set is shown in Fig. 8. With this data set we aim to demonstrate that interesting non-spherical cluster geometries are possible when clustering with proximity relational knowledge. We consider two cases: the first is when the distance relational data R is computed using the Euclidean distance. We then convert the distance relational data R to proximity knowledge via expression (18). The second case is when the proximity function is given by the Gaussian proximity function. The number of clusters is set to the number of known clusters ($c=2$) and the fuzzification coefficient is varied over the values $m=1.2$, 1.4, 1.7, 2.0, 2.5, and 3.0. The fuzzification coefficient that results in the highest performance is reported. The results are averaged over 20 runs and the mean and standard deviation of classification rate (CR) and reconstruction error (RE) are reported in Table 3.

In this example, it is demonstrated that the selection of the proximity function impacts the performance and can provide greater flexibility in terms of the cluster geometries when compared with relational FCM with the Euclidean distance.

We evaluate proximity fuzzy clustering and the NERF C-Means algorithm on Bezdek's relational data set [2], and the Windham data set [3]. The number of clusters and the value of the fuzzification coefficient are determined by the procedure outlined in Section 3.2.

From the experiments reported in Table 4, the two algorithms perform similarly in these examples.

Two more experiments with synthetic data are conducted. The first is where the proximity function is the identity function, i.e.

$$p(O_j,O_k)=\begin{cases}1, & \forall j=k\\ 0, & \forall j\neq k\end{cases} \tag{18}$$

for $j,k=M$ where $M=10$. The aim is to measure how well proximity clustering forms clusters when the proximity information is at the extremes. The distance knowledge used for relational FCM is where all objects are equidistant, i.e. $d(O_j,O_k)=1-p(O_j,O_k)$. The second consists of an unbalanced synthetic data set shown in Fig. 9. The results are shown in Table 5.

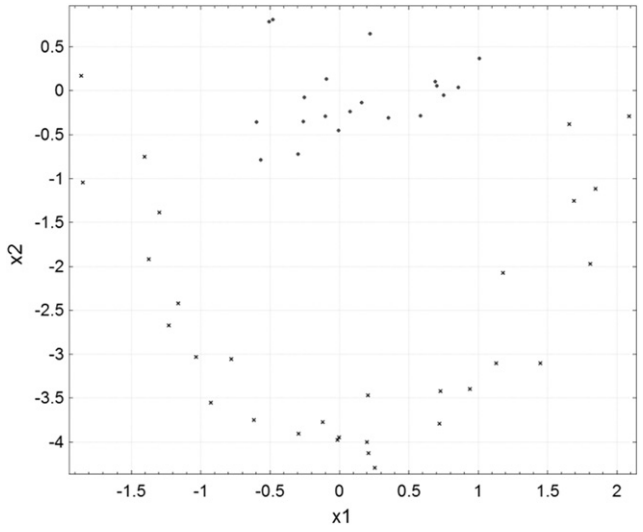

Fig. 8. A synthetic half-ring data set in R^2.

Interestingly in both of these cases, proximity fuzzy clustering outperforms relational FCM. The reconstruction error is 0 with the identity data set because the number of clusters equals the number of entities in the data set; therefore, the membership values are encoded without losing any proximity information. The proximity fuzzy clustering algorithm is able to identify the small cluster in the unbalanced data set. The relational FCM is unable to cluster the identity data set. Several of the clusters contain more than one object despite the fact that no objects bear any similarity. The proximity fuzzy clustering is able to place each object in its own cluster for the identity data set.

Table 4
Performance on Bezdek and Windham relational data sets.

Relational data set	*Proximity fuzzy clustering*				*Relational FCM*			
	c	*m*	*CR* (%)	*RE*	*c*	*m*	*β*	*CR* (%)
Bezdek	2	2.5	96.0 ± 0.0	0.119 ± 0.000	2	2.0	0.0	96.0 ± 0.0
Windham	2	3.0	100.0 ± 0	0.298 ± 0.000	2	2.0	0.0	100.0 ± 0.0

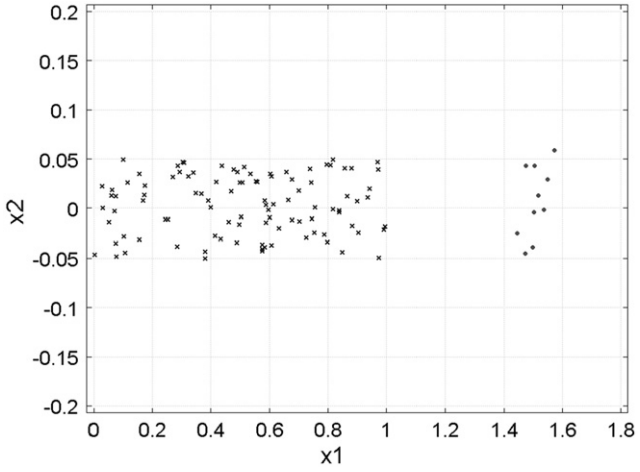

Fig. 9. An unbalanced synthetic data set in R^2.

Table 5
Performance on the "identity" and "unbalanced" synthetic relational data sets.

Relational data set	*Proximity fuzzy clustering*				*Relational FCM*			
	c	*m*	*CR* (%)	*RE*	*c*	*m*	*β*	*CR* (%)
Identity	10	1.4	100.0 ± 0.0	0.000 ± 0.000	10	2.0	0.0	83.5 ± 5.9
Unbalanced	2	1.4	100.0 ± 0	0.070 ± 0.000	2	2.0	0.0	90.9 ± 0.0

Table 3
Performance on the half-ring data set.

Algorithm	*c*	*m*	*CR* (%)	*RE*
Proximity fuzzy clustering	2	2.0	84.0 ± 0.0	0.059 ± 0.000
Proximity fuzzy clustering (Gaussian, $\sigma^2=1$)	2	2.0	100.0 ± 0	$1.2e-4\pm0.0e-4$
Relational FCM	2	1.7	84.0 ± 0.0	n/a

4.3. Real-world relational data

Several experiments are conducted on some real-world relational data including the cat cortex [4,5,32] and the fat oil data set [3]. The cat cortex data set is a similarity matrix consisting of the connection strengths between 65 cortical areas of the cat brain. The connection strengths are rated on a scale of 0–4 where 0 is a full self connection and 4 is a completely unconnected. This data description involves fuzzy sets and therefore it is appropriate for representation via a proximity matrix. We transform this data into membership values indicative of closeness, i.e. a proximity value of 1 is self-connection and proximity of 0 is no connection. It is noted that the relational data are not Euclidean [5]. Therefore, we must apply the non-Euclidean version of the relational FCM algorithm. No changes are required to the proximity fuzzy clustering as the requirement of Euclidean is not necessary. The second data set is a commonly encountered data set in relational fuzzy clustering [3,39], where a similarity matrix is given for 8 different types of oil. We use the number of clusters equal to the number of known classes in the problem. Unfortunately, there are not many relational data sets available due to the current preference of storing data sets as a vector of attributes rather than a similarity to dissimilarity matrix. The experiments are repeated 20 times and the means and standard deviations of the performance (classification rate) are reported in Table 6.

The cat cortex is a challenging relational data set to cluster [4,5]. The scaling factor β for relational FCM is 1.1, which means that the relational distance information does not satisfy the properties of a Euclidean metric. In particular, the distance between patterns and prototypes, which are not explicitly calculated in relational FCM, is less than zero in some cases. Thus a positive scaling factor $\beta=4.87$ is needed to ensure the distance values are strictly non-negative. Interestingly, the performance varies greatly with relational FCM. Proximity fuzzy clustering on the other hand performs much better and demonstrates more consistent performance. The fat oil data set is quite small and thus the method achieves a classification rate of 100%. We note that the standard deviations are quite small. It was observed that the performance was often very consistent even with different starting conditions (i.e. different random initial membership matrices for each run).

We now consider UCI Machine Learning real-world data sets [38] where the relational information is derived from data stored as a vector of attributes. The proximity information comes from a proximity function and the distance information from a distance function, e.g. Euclidean distance. This experiment looks at the effect of the choice of the proximity function on the clustering. We consider three proximity functions: cosine, Gaussian and polynomial. The Gaussian kernel width is varied over the set of values {0.01, 0.05, 0.1, 0.25, 0.5, 1, 2, 4, 7, 10, 25, 50}. The polynomial kernel is varied over the Cartesian product of the parameter sets $p\in\{1, 2, 3, 4, 5, 6\}$ and $\theta\in\{0, 0.5, 1, 2, 4, 7, 10, 15, 20, 25, 50\}$. The proximity parameters reported are those that result in the best performance. The number of clusters selected equals the number of known classes in this problem, i.e. c=2.

When we cluster the data with relational FCM using the standard Euclidean distance, we obtain a classification rate of $70.9\pm0.0\%$. The value of m is 1.2 and the value of β is $1.1e^{-7}$. Here we observe that clustering with the Gausian and polynomial proximity functions perform much better than both relational FCM and proximity fuzzy clustering with the cosine proximity function. Interestingly, the reconstruction error when clustering with the polynomial proximity function is much higher than clustering with the Gaussian proximity function indicating the Gaussian proximity function does a better job of retaining proximity information in membership values. The next UCI Machine Learning data set we consider is the wine data set [38]. The number of clusters is set to $c=3$ as this is the number of known classes.

Clustering the data with relational FCM using the standard Euclidean distance results in a classification rate of $97.2\pm0.0\%$ with $\beta=0$ and $m=1.2$. Interestingly, the performance of relational FCM degrades considerably with larger values of m. For example when $m=2$, the classification rate of relational FCM is $60.1\pm0.0\%$. It is also observed here that clustering with the Gaussian proximity function produces a lower reconstruction error than clustering with the polynomial proximity function meaning that the Gaussian proximity function results in higher quality clusters. From Tables 7 and 8, we can see that the choice of the proximity function has an important impact on the clustering performance. For the ionosphere data set in particular, clustering with the polynomial proximity function outperforms the others. Interestingly, the power parameter is $p=1$ and it is observed that the performance degrades as p increases. The value of m does not appear to have a significant impact on the performance. For the wine data set, clustering with the Gaussian and polynomial proximity functions out-perform the results obtained when using cosine.

Table 7
Performance on the ionosphere data set.

Proximity function	*Proximity fuzzy clustering*			
	m	*params*	*CR* (%)	*RE*
Cosine	3.0		68.9 ± 0.0	0.127 ± 0.000
Gaussian	2.0	$\sigma=50$	73.8 ± 0.0	0.097 ± 0.000
Polynomial	2.0	$p=1, \theta=50$	75.5 ± 0.0	0.287 ± 0.000

Table 8
Performance on the wine data set.

Proximity function	*Proximity fuzzy clustering*			
	m	*params*	*CR* (%)	*RE*
Cosine	2.5		93.8 ± 0.0	0.120 ± 0.000
Gaussian	1.4	$\sigma=10$	97.8 ± 0.0	0.096 ± 0.000
Polynomial	2.0	$p=3, \theta=10$	97.8 ± 0.0	0.177 ± 0.000

Table 6
Performance on the "cat cortex" and "fat oil" relational data sets.

Relational data set	*Proximity fuzzy clustering*				*Relational FCM*			
	c	*m*	*CR* (%)	*RE*	*c*	*m*	β	*CR* (%)
Cat cortex	4	1.4	84.6 ± 0.0	0.507 ± 0.000	4	2.0	4.87 ± 0.07	70.9 ± 9.7
Fat oil	2	2.0	100.0 ± 0.0	0.103 ± 0.000	2	1.7	0.0	100.0 ± 0.0

5. Software requirements analysis—a real-world practical application

5.1. Introduction

We demonstrate and analyze the usability of these clustering methods in an interesting real-world practical application for relational clustering, which arises from software product lines (SPL) development [33] and requirements analysis. This problem involves the clustering of variability models that describe a Crisis Management System (CMS), which was first published in [34]. The CMS is a software system that automates the coordination of crisis situations, ranging from identifying the occurrence of a crisis and dispatching resources such as ambulances to reporting and storing the completion of the crisis. The CMS covers a large set of responsibilities and the case description introduces approximately 35 textual requirements that describe the base functionality and quality of service that is to be expected of the CMS. The case description in [34] also makes clear that the CMS case in essence describes an SPL, as a typical instantiation of the CMS will be more specialized, e.g. a car crash CMS or an airport CMS. We have compiled a set of variability models that describe some CMS's.

The data set consists of 15 variability models of the CMS that originate from various sources. Seven have been produced using an automated tool called Arborcraft [35]. Arborcraft was provided with the textual description of the CMS case and the derivation process was executed with various different parameters, such as the width and depth of the variability model. Also, the CMS description was provided to expert software designers to provide three variability models. The data set is completed by including variability models that result from a related textual description of a more specialized crisis management system, resulting in one expert-derived and four Arborcraft derived variability models. The models have been clustered with respect to three criteria: *Logging*, *Missions* and *Strategies*. Each of these criteria defines a number of nodes based on which the proximity of the models in the data set is calculated and consequently the clustering is performed. Each node in a criterion is represented by a textual description of the sub-concept it represents.

To motivate the application of relational clustering to discovering structure in variability models, we refer to [36,37]. The idea is to cluster the variability models to determine commonalities between the variability models and use that information to eventually facility software design reuse. The focus of this small case study is simply on the clustering of the variability models.

5.2. Clustering results

The data sets are clustered using proximity fuzzy clustering and the approach provided in Section 3.3 for determining the number of clusters and fuzzification coefficient m. For comparative analysis, we also perform relational FCM clustering with the same number of clusters. The results are shown in Table 9.

The membership values for proximity fuzzy clustering and relational FCM are given in the Fig. 10.

Graphs M14 and M15 with the Logging criteria have considerably lower membership values with relational FCM, which results in a significant difference in the overall clusters. The clustering produced by proximity fuzzy clustering is closer to the result expected by an expert since the relational FCM is unable to determine that M15 belongs in its own cluster as it is very different from the other graphs. We conclude that on this problem proximity fuzzy clustering provides better clustering performance than the relational FCM.

6. Conclusions

In this study, we introduced two types of domain knowledge: distance and proximity. We have demonstrated that these fundamental forms of knowledge can be used to cluster data relationally. However, deciding on the form of domain knowledge to use in a particular problem is an important consideration that can either positively or negatively impact clustering performance. In this paper we looked at the advantages and disadvantages of these forms of domain knowledge by comparing two very similar yet distinctly different relational clustering algorithms: proximity fuzzy clustering which uses proximity knowledge and relational FCM, which uses distance knowledge. Clustering with proximity presents a number of interesting advantages. Firstly, proximity is the degree of similarity between objects or entities in the problem and therefore acts as a membership function to the notion of closeness. Secondly, proximity is closely related to the kernel function. While this connection has been well known, the benefit of clustering with proximity knowledge is that proximity is a more natural way to express relational information while also being able to reap the benefits of kernel-based clustering, namely very interesting non-spherical cluster geometries [22]. Clustering with this kind of information presents some benefits including:

- A diverse set of proximity functions including kernel functions that can often significantly enhance performance.
- An intuitive measure of closeness that can be derived directly from proximity functions or even ad hoc human knowledge (e.g. ratings on a scale of 1-10) since the triangular inequality is not required with proximity.
- No need to re-derive the proximity fuzzy clustering algorithm when choosing different proximity functions as typically encountered with standard FCM.

Experiments on both synthetic and real-world relational data demonstrate that clustering with proximity information provides important benefits including:

- Increased performance with the appropriate selection of the proximity function.
- Better performance on unbalanced data sets.
- Ability to form non-spherical clusters.

Table 9
Tree clustering results.

Data Set	*Proximity fuzzy clustering*			*Relational FCM*		
	c	m	Clusters	c	m	Clusters
Logging	3	2.3	{M15}, {M7,M11,M12,M13}, {M1,M2,M3,M4,M5,M6,M8,M9,M10,M14}	3	2.0	{M8,M9,M10}, {M7,M11,M12,M13}, {M1,M2,M3,M4,M5,M6,M14,M15}
Missions	3	4.0	{M1,M2,M3,M4,M5,M6,M7,M11}, {M8,M9,M10}, {M12,M13,M14,M15}	3	2.0	{M1,M2,M3,M4,M5,M6,M7,M11}, {M8,M9,M10}, {M12,M13,M14,M15}
Strategies	2	4.0	{M1,M2,M3,M4,M5,M6,M7,M11}, {M8,M9,M10,M12,M13,M14,M15}	2	2.0	{M8,M9,M10,M12,M13,M14,M15}, {M1,M2,M3,M4,M5,M6,M7,M11}

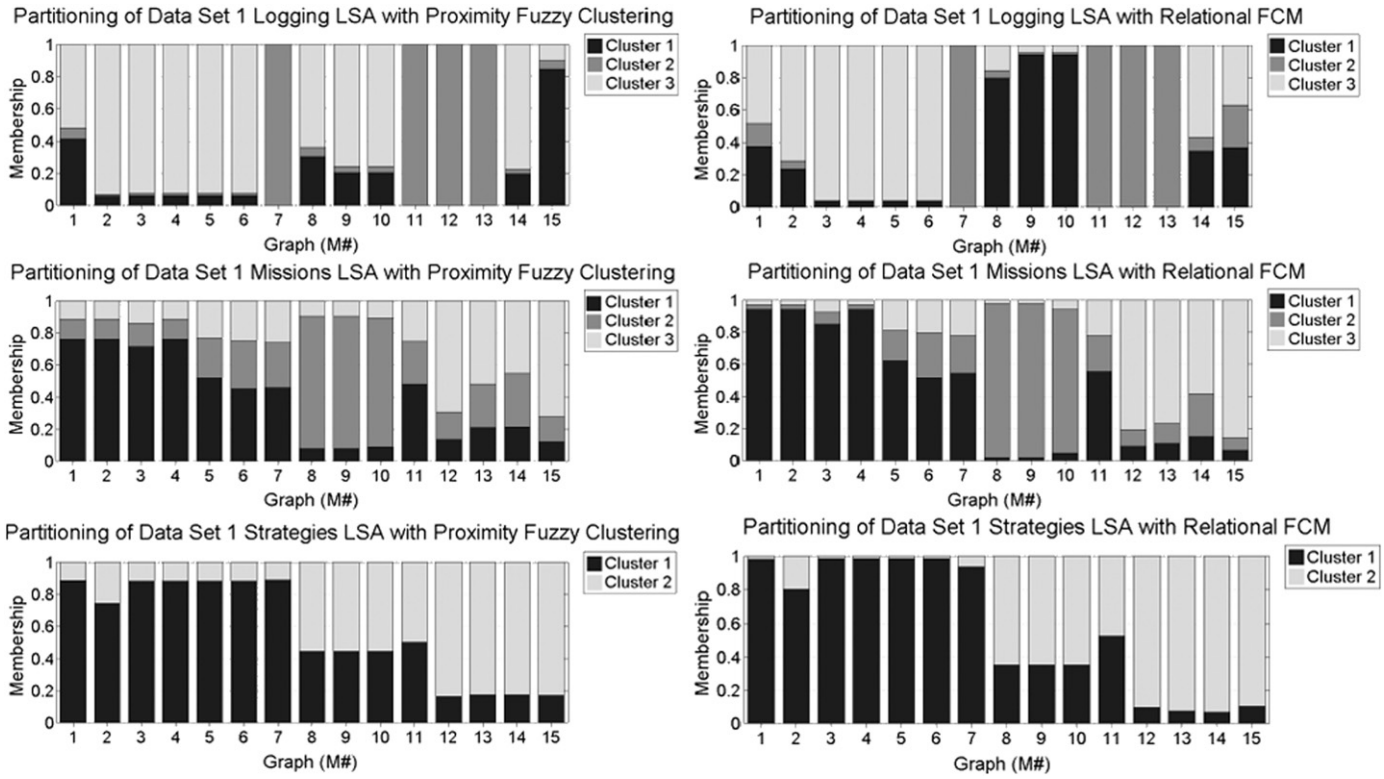

Fig. 10. Membership grades of the tree data set.

We mention that the performance depends on the selection of the proximity function (structural optimization) and any associated parameters (parametric optimization). These are still open research problems. Our future direction of research will include developing methods to address these issues.

Acknowledgments

Support from the Natural Sciences and Engineering Research Council of Canada (NSERC) and Canada Research Chair (CRC) Program (W. Pedrycz) is gratefully acknowledged. This research is sponsored by the European Union as part of the DISCS project (IEF-221280).

References

[1] P. Berkhin, A survey of clustering data mining techniques, in: J. Kogan, C. Nicholas, M. Teboulle (Eds.), Grouping Multidimensional Data, Spring, Berlin, Heidelberg, 2006, pp. 25–71.
[2] R.J. Hathaway, J.W. Davenport, J.C. Bezdek, Relational duals of the c-means clustering algorithms, Pattern Recognition 22 (2) (1989) 205–212.
[3] R.J. Hathaway, J.C. Bezdek, Nerf c-means: Non-Euclidean relational fuzzy clustering, Pattern Recognition 27 (1994) 429–437.
[4] T. Denoeux, M.H. Masson, EVCLUS: evidential clustering of proximity data, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 34 (1) (2004) 95–108.
[5] M.H. Masson, T. Denoeux, RECM: relational evidential c-means algorithm, Pattern Recognition Letters 30 (2009) 1015–1026.
[6] W. Pedrycz, V. Loia, S. Senatore, P-FCM: a proximity—based fuzzy clustering, Fuzzy Sets and Systems 148 (2004) 23–41.
[7] D. Graves, W. Pedrycz, Proximity fuzzy clustering and its application to time series clustering and prediction, in: Proceedings of the IEEE International Conference on Intelligence Systems Design and Applications, 2010, pp. 49–54.
[8] W. Pedrycz, K. Hirota, A consensus-driven fuzzy clustering, Pattern Recognition Letters 29 (2008) 1333–1343.
[9] U. von Luxburg, A tutorial on spectral clustering, Statistics and Computing 17 (4) (2007) 395–416.
[10] S. Bocker, S. Briesemeister, G.W. Klau, Exact algorithms for cluster editing: evaluation and experiments, Algorithmica 60 (2) (2009) 316–334.
[11] M. De Caceres, F. Oliva, X. Font, On relational possibilistic clustering, Pattern Recognition 39 (11) (2006) 2010–2024.
[12] J.P. Mei, L. Chen, Fuzzy clustering with weighted medoids for relational data, Pattern Recognition 43 (5) (2010) 1964–1974.
[13] R.J. Hathaway, J.C. Bezdek, Clustering incomplete relational data using the non-Euclidean relational fuzzy c-means algorithm, Pattern Recognition Letters vol. 23 (1-3) (2002) 151–160.
[14] D.S. Guru, B.B. Kiranagi, P. Nagabhushan, Multivalued type proximity measure and concept of mutual similarity value useful for clustering symbolic patterns, Pattern Recognition Letters 25 (10) (2004) 1203–1213.
[15] H. Frigui, C. Hwang, F.C.H. Rhee, Clustering and aggregation of relational data with applications to image database categorization, Pattern Recognition 40 (11) (2007) 3053–3068.
[16] R. Bisdorff, Electre-like clustering from a pairwise fuzzy proximity index, European Journal of Operational Research 138 (2002) 320–331.
[17] R.B.A. Bakar, J. Watada, W. Pedrycz, A proximity approach to DNA based clustering analysis, International Journal of Innovative Computing, Information and Control 4 (5) (2008) 1203–1212.
[18] S. Shenoi, A. Melton, Proximity relations in the fuzzy relational database model, Fuzzy Sets and Systems 31 (1989) 285–296.
[19] M. Popescu, J.M. Keller, J.A. Mitchell, Fuzzy measures on the gene ontology for gene product similarity, IEEE Transactions on Computational Biology and Bioinformatics 3 (3) (2006) 263–274.
[20] V. Loia, W. Pedrycz, S. Senatore, Semantic web content analysis: a study in proximity-based collaborative clustering, IEEE Transactions on Fuzzy Systems 15 (6) (2007) 1294–1312.
[21] B. Xiao, A. Torsello, E.R. Hancock, Isotree: tree clustering via metric embedding, Neurocomputing 71 (2008) 2029–2036.
[22] R.C. Wilson, E.R. Hancock, B. Luo, Pattern vectors from algebraic graph theory, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (7) (2005) 1112–1124.
[23] M. Neuhaus, H. Bunke, Edit distance-based kernel functions for structural pattern classification, Pattern Recognition 39 (2006) 1852–1863.
[24] K. Riesen, H. Bunke, Approximate graph edit distance computation by means of bipartite graph matching, Image and Vision Computing 27 (2009) 950–959.
[25] X. Gao, B. Xiao, D. Tao, X. Li, A survey of graph edit distance, Journal of Pattern Analysis & Applications 13 (1) (2010) 113–129.
[26] E.R. Hancock, R.C. Wilson, Pattern analysis with graphs: parallel work at Bern and York, Pattern Recogition Letters (2011). doi:10.1016/j.patrec.2011.08.012.
[27] K.R. Muller, S. Mika, G. Ratsch, K. Tsuda, B. Scholkopf, An introduction to kernel-based learning algorithms, IEEE Transactions on Neural Networks 12 (2) (2001) 181–201.
[28] M. Filippone, F. Camastra, F. Masulli, S. Rovetta, A survey of kernel and spectral methods for clustering, Pattern Recognition 41 (2008) 176–190.

[29] M. Shiga, I. Takigawa, H. Mamitsuka, A spectral approach to clustering numerical vectors as nodes in a network, Pattern Recognition 44 (2) (2011) 236–251.
[30] M. Girolami, Mercer kernel-based clustering in feature space, IEEE Transactions on Neural Networks 10 (5) (1999) 1000–1017.
[31] D. Graves, W. Pedrycz, Kernel-based fuzzy clustering and fuzzy clustering: a comparative experimental study, Fuzzy Sets and Systems 161 (2010) 522–543.
[32] J.W. Scannell, C. Blakemore, M.P. Young, Analysis of connectivity in the cat cerebral cortex, The Journal of Neuroscience 15 (2) (1995) 1463–1483.
[33] K. Pohl, G. Böckle, F.J. van der Linden, Software Product Line Engineering: Foundations, Principles and Techniques Springer-Verlag New York, Inc, 2005.
[34] J. Kienzle, N. Guelfi, S. Mustafiz, Crisis Management Systems: A Case Study for Aspect-Oriented Modeling, 2009, 〈http://www.cs.mcgill.ca/~joerg/taosd/TAOSD/TAOSD_files/AOM_Case_Study.pdf〉 (retrieved 08/03/2010).
[35] N. Weston, R. Chitchyan, A. Rashid, A framework for constructing semantically composable feature models from natural language requirements, in: Proceedings of the ACM International Conference on Software Product Line, vol. 446, 2009, pp. 211–220.
[36] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1994.
[37] K. Kang, S. Cohen, J. Hess, W. Novak, A.S. Peterson, Feature Oriented Domain Analysis (FODA) Feasibility Study, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-90-TR-21, ESD-90-TR-222, 1990.
[38] A. Asuncion, D.J. Newman, UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science, 2007 [Online] Available: 〈http://archive.ics.uci.edu/beta/〉.
[39] K.C. Gowda, E. Diday, Symbolic clustering using a new similarity measure, IEEE Transactions on System Man Cybernetics 22 (1992) 368–378.

Daniel Graves received his B.Sc. in Computing Science at Thompson Rivers University, Kamloops, BC, Canada and recently completed his PhD at the Department of Electrical and Computer Engineering, University of Alberta, Edm., Alb., Canada. His research interests are in computational intelligence, pattern recognition, fuzzy sets, neural networks, acoustics and signal processing. He is co-inventor of a commercial product called ClassTalk and won the Hetu prize from the Canadian Acoustical Association (CAA) for undergraduate research in acoustical signal processing. He did an internship with Raumfahrt Systemtechnik (RST), Salem, Germany in synthetic aperture radar (SAR).

Joost Noppen is a senior research fellow at the Lancaster University in the UK and holds an M.Sc. degree and a PhD degree in Computer Science from the University of Twente in the Netherlands. Joost's long-standing research interest is in software engineering, software design and development processes. He has been actively involved in software design process research, and design assistance based on mathematical models is a key focus of his work. His research has resulted in extensions to current design methods with advanced modeling techniques for capturing and reasoning with impreciseness, including models based on probability theory and fuzzy set theory and has been published in the Soft Computing Journal, IEE Software and at various conferences. His current research interest is the application of these techniques in the context of software product line development.

Witold Pedrycz (M'88–SM'94) is Professor of Computer Engineering, Department of Electrical and Computer Engineering, University of Alberta, Edm., Alb., Canada. He isactively pursuing research in computational intelligence, fuzzy modeling, data mining,fuzzy control including fuzzy controllers, pattern recognition, knowledge-based neuralnetworks, and relational computation. He has published numerous papers in the area ofapplied fuzzy sets as well three research monographs (*Fuzzy Control and Fuzzy Systems*, 988 and 1993; Fuzzy Relation Equations and Their Applications to Knowledge *Engineering*, 1988; and *Fuzzy Sets Engineering*, 1995). He is also one of the Editors-in-Chief of the *Handbook of Fuzzy Computation*. Dr. Pedrycz is a member of many programcommittees of international conferences and serves on editorial boards of journals onfuzzy sets (including the IEEE TRANSACTIONS ON FUZZY SYSTEMS, the IEEE TRANSACTIONS ON NEURAL NETWORKS, and *Fuzzy Sets and Systems*) and pattern recognition (*Pattern Recognition Letters*).