Fuzzy Lattice Reasoning (FLR) Neural Computation for Weighted Graph Partitioning

Vassilis G. Kaburlasos*[1], Lefteris Moussiades[1], and Athena Vakali[2]

[1]Technological Educational Institution of Kavala
Department of Industrial Informatics
GR-65404 Kavala, Greece
Emails: {vgkabs,lmous}@teikav.edu.gr

[2]Aristotle University of Thessaloniki
Department of Informatics
GR-54124 Thessaloniki, Greece
Email: avakali@csd.auth.gr

Corresponding Author

Name:	Vassilis G. Kaburlasos
Postal Address:	Department of Industrial Informatics Technological Educational Institution of Kavala GR 654 04 Kavala Greece
Phone Number:	+30 (2510) 462-320
Fax Number:	+30 (2510) 462-348
E-mail Address:	vgkabs@teikav.edu.gr

* Corresponding Author Fax. no. + 30 (2510) 462-348, E-mail address: vgkabs@teikav.edu.gr

Fuzzy Lattice Reasoning (FLR) Neural Computation for Weighted Graph Partitioning

Abstract

The fuzzy lattice reasoning (FLR) neural network was introduced lately based on an inclusion measure function. This work presents a novel FLR extension, namely agglomerative similarity measure FLR, or asmFLR for short, for clustering based on a similarity measure function, the latter (function) is based on a metric. We demonstrate application in a metric space emerging from a weighted graph towards partitioning it. The asmFLR compares favorably with four alternative graph-clustering algorithms from the literature in a series of computational experiments on artificial data. In addition, our work introduces a novel index for the quality of clustering, which (index) compares favorably with two popular indices.

Keywords: Clustering; Graph partitioning; Fuzzy lattices; Measurable path; Metric; Similarity Measure

1. Introduction

Neural computation is typically pursued on numeric data in the Euclidean space R^N, where there is an abundance of mathematical tools available. Nonnumeric data were also considered including linguistic (fuzzy) data [32], [46]. Alternative (nonnumeric) data of practical interest include *structured* (*graph*) data.

Our long-term interest, in the context of this work, is in partitioning a weighted graph, which represents a WWW-site as explained below. Advantages of meeting the aforementioned task by neural computing techniques include a capacity for massively parallel data processing as well as a capacity for both learning and generalization. A number of authors have pursued neural computing involving graphs as described next.

A structure (graph) was used a vehicle for a unified data representation including arrays, sequences, and trees; in conclusion, a neural paradigm was proposed for learning, probabilistically, IO-isomorph transductions from an input- to an output- structured space, where transductions admit a recursive hidden state-space representation [18]. The latter work has spurred a lasting research activity including various enhancements and applications [1], [2], [8], [20], [47], [48]. In addition, a number of technical issues were studied including ambiguity [19], node-complexity, etc. [21]. Extensions to Kohonen's self-organizing map (SOM) were also reported [27], [29]. However, the aforementioned neural paradigm pursues learning by optimizing an "energy type" *objective* (error) function in R^N using "number crunching" techniques. Hence, even though semantics may exist in (structured) input/output data, nevertheless semantics is absent during data processing. Also, the aforementioned neural paradigm does not induce descriptive, decision-making knowledge for the general user.

On a different context, especially popular is an employment of the Hopfield neural network in graph theoretic problems including: maximum cut [9], search [58], minimum vertex cover [66], etc. Note that, typically, a Hopfield network ignores semantics and it pursues optimization of an "energy type" *objective* function. Nevertheless, a different (Hopfield) network, which does not neglect graph-theoretical properties, preserved under isomorphism, was also presented for deciding whether two graphs are structurally equivalent [31].

Several works have dealt with the problem of *graph matching* using neural networks [62]. For instance, a general framework was proposed for *approximate graph matching* problems in image retrieval tasks [22]. Moreover, a (neural) graph matching technique was presented for object recognition [57], where a graph representation of the neuron positions/interconnections reflects the structure of model objects. Furthermore, a pattern recognition problem was formulated as *labelled graph matching* towards finding the best match between an input graph and a stored graph [59].

The abovementioned graph matching techniques typically assume overlapping graphs, moreover an objective (cost) function is assumed. A recent work has introduced a (metric) *graph edit distance* (GED), derived from a metric cost function, in a set of graphs embedded in a labelled complete graph G_Ω, namely *edit grid*; moreover it demonstrated, comparatively, a successful pattern recognition application regarding a chemical information system [33]. Furthermore, three novel graph kernels were introduced for measuring similarity between feature vectors of chemical molecules towards classification of chemical compounds represented by graphs of covalent bonds [54].

The specifications of our task here, namely *weighted graph partitioning*, require different tools since we need to partition a graph into non-overlapping (sub)graphs by clustering, as explained below. There exists an abundance of graph (clustering) algorithms in various application domains including circuit partitioning [6], pattern recognition [15], [25], [65], structure comparison [26], [33], etc. Note that the literature is dominated by *divisive* type clustering algorithms [4], [16], [61], where clusters are computed "top-down" by successively batch-splitting a graph. A different type of clustering, namely *agglomerative* (clustering), may proceed "bottom-up" by incrementally augmenting graphs. However, agglomerative clustering is not usually pursued mainly due to a shortage of enabling mathematical tools.

A fundamentally novel approach to neural computing was introduced lately applicable on partially(lattice)-ordered data including logic values, numbers, sets, symbols, and graphs [36], [38], [40], [41]. In conclusion, the Fuzzy Lattice Reasoning (FLR) emerged as an enhanced version of the learning algorithm employed by the neural-fuzzy classifier σ-FLN(MAP) [42]. The latter (classifier), in turn, has emerged as a lattice data domain extension of the well-known neural classifier fuzzy-ARTMAP, or FAM for short [40].

The operation of FLR was originally based on an *inclusion measure* function [39], [42] towards inducing descriptive, decision-making knowledge (rules). A cluster, computed by FLR, is interpreted as an information granule [37]. A successful employment of a FLR version on graphs was already demonstrated using fuzzy lattice neurocomputing (FLN) [52], where clusters (namely, *hyperwords*) were computed in a master-graph, the latter encoded a thesaurus of English language synonyms; in conclusion, hyperwords were used for dimensionality reduction in a classification problem regarding large text documents. Nevertheless, the work in [52] ignores graph connectivity and treats, quite restrictively, a graph as an unstructured set of both *vertices* (or, *nodes*) and *edges* (or, *links*).

A substantial novelty of this work is consideration of graph connectivity by a different FLR version based on a novel *similarity measure* function. In conclusion, a master-graph is treated here as a (structured) data domain where, in contrast with the conventional data domain R, which includes a single shortest path between two different real numbers $a,b \in \mathsf{R}$ [40], [42], there might be multiple shortest paths between two different nodes a and b in a master-graph.

Previous versions of FLR dealt either with hyperboxes in R^N [42] or with FINs [39]; whereas, this work extends the applicability of FLR to graphs based on a metric distance – Note that a number of distances between graphs have been proposed by different authors [5], [55].

There are similarities as well as substantial differences between FLR and previous graph processing algorithms. For instance, the latter algorithms have proposed a unification of graph /set /series data at a “representation” (encoding) level in the Euclidean space R^N, whereas FLR proposes a disparate data fusion in a mathematical product-lattice $\mathsf{L}= \mathsf{L}_1 \times \ldots \times \mathsf{L}_N$ data domain [36], [40]. In conclusion, previous algorithms carry out data processing by “number crunching” techniques, whereas FLR may retain semantics throughout data processing [34]. In addition, the FLR may induce descriptive decision-making knowledge (rules) from the training data. An additional advantage for FLR includes the capacity to introduce tunable nonlinearities [34], [36], [37], [39], [40], [42].

This paper builds on previous work [43] including the following substantial novelties. First, we introduce a number of useful mathematical results including theoretical substantiation (i.e. proofs) and, second, we demonstrate comparatively a large number of new experimental results on artificial data. In addition, the work in [43] employs solely the *inner-transactions ratio* (ITR) index for evaluating the quality of graph clustering in a real-world application, where a weighted master-graph represents traffic. However, the ITR index "by definition" ascribes larger values to fewer clusters; hence, it encourages the computation of a single cluster. Whereas, the work here employs three different indices, including a novel one, for evaluating advantageously the quality of graph clustering. Future work will demonstrate a real-world application using the novel (mathematical) tools detailed here.

The layout is as follows. Section 2 defines metrics in measure (path) spaces. Section 3 summarizes the theory of fuzzy lattices including useful extensions; the practical relevance is also explained. Section 4 presents a novel neural computing algorithm for graph clustering. Comparative experimental results are presented in section 5 including also a discussion. Section 6 concludes by summarizing the contribution of this work. Finally, the Appendix summarizes useful mathematical definitions and proofs.

2. Measurable paths and metrics

This section introduces useful metric distances between sets in a metric space; the latter emerges from measurable paths in a graph. The Appendix lists useful definitions.

2.1. Measurable paths

Inspired from measure theoretic analysis for path-planning in robotics [44], this section introduces useful terminology. Consider the following definition regarding a *path*.

Definition 2.1. Let X be a set and D be a totally-ordered "indexing set" with least- and greatest- elements O and I, respectively. A *path* from $a\in X$ to $b\in X$, symbolically $a\to b$, is a function $p_{ab}(.)$: $\mathsf{D}\to X$ such that $p_{ab}(O)=a$ and $p_{ab}(I)=b$. ■

We point out that the indexing set D implies a totally-ordered complete lattice $(\mathsf{D},\leq)$, whose cardinality may be either *finite*, e.g. $\mathsf{D}=\{0,1,2,\ldots,M\}$, or *infinite*; in turn, the latter could be either *countable*, e.g. $\mathsf{D}=\{0,1,2,\ldots\}$, or *uncountable*, e.g. $\mathsf{D}=[0,1]$ or $\mathsf{D}=\mathsf{R}^{\geq 0}$ – We assume that both infinite sets $\{0,1,2,\ldots\}$ and $\mathsf{R}^{\geq 0}$ have greatest element $I=+\infty$.

Our interest is in "measurable" paths $p_{ab}(.)$: $\mathsf{D}\to X$ such that a *measure space* $(\mathsf{D},\Sigma_{\mathsf{D}},m_{\Sigma_{\mathsf{D}}})$ can be defined with $0<m_{\Sigma_{\mathsf{D}}}(\mathsf{D})<+\infty$. The σ-algebra Σ_{D} includes intervals $[t_1,t_2]$, where $t_1,t_2\in\mathsf{D}$ with $t_1\leq t_2$. In particular, $m_{\Sigma_{\mathsf{D}}}(\mathsf{D}) = m_{\Sigma_{\mathsf{D}}}([O,I])$ is called *length* (of the path $a\to b$ from $a\in X$ to $b\in X$).

A *measure* function $m_{\Sigma_\mathsf{D}}: \Sigma_\mathsf{D} \rightarrow \mathsf{R}^{\geq 0}$ is induced from a measure space $(X,\Sigma_X, m_{\Sigma_X})$ as follows. Number $m_{\Sigma_\mathsf{D}}(\mathsf{D})$ is calculated from a partition $\{\mathsf{D}_1,\ldots,\mathsf{D}_\mathrm{N}\}$ of D such that there are no $x \neq y$ with $p_{ab}(x) = p_{ab}(y)$, i.e. there are no "cycles" on the path $a \rightarrow b$. In conclusion, $m_{\Sigma_\mathsf{D}}(\mathsf{D}) = m_{\Sigma_X}(p_{ab}(\mathsf{D}_1)) + \ldots + m_{\Sigma_X}(p_{ab}(\mathsf{D}_\mathrm{N}))$, where $p_{ab}(\mathsf{D}_\mathrm{i})$ denotes the image (set) of D_i, i=1,…,N – We point out that any partition of D results in the same number $m_{\Sigma_\mathsf{D}}(\mathsf{D})$.

Measure space $(\mathsf{D},\Sigma_\mathsf{D}, m_{\Sigma_\mathsf{D}})$ implies complete lattice $(\Sigma_\mathsf{D},\subseteq)$ with least and greatest elements $\varnothing$ and D, respectively; moreover, function $m_{\Sigma_\mathsf{D}}: \Sigma_\mathsf{D} \rightarrow \mathsf{R}^{\geq 0}$ is a positive valuation in lattice $(\Sigma_\mathsf{D},\subseteq)$ [40]. Our interest, next, focuses on the (complete) indexing lattice $(\mathsf{D},\leq)$.

Function $v_\mathsf{D}(x) = m_{\Sigma_\mathsf{D}}([O,x])$, $x \in \mathsf{D}$ is a positive valuation in the indexing lattice $(\mathsf{D},\leq)$ because 1) $v_\mathsf{D}(x)+v_\mathsf{D}(y) = m_{\Sigma_\mathsf{D}}([O,x]) + m_{\Sigma_\mathsf{D}}([O,y]) = m_{\Sigma_\mathsf{D}}([O,x \vee y]) + m_{\Sigma_\mathsf{D}}([O,x \wedge y]) = v_\mathsf{D}(x \vee y)+v_\mathsf{D}(x \wedge y)$, and 2) $x<y \Rightarrow m_{\Sigma_\mathsf{D}}([O,x]) < m_{\Sigma_\mathsf{D}}([O,y]) \Rightarrow v_\mathsf{D}(x)<v_\mathsf{D}(y)$. Hence, a metric d_D: $\mathsf{D}\times\mathsf{D} \rightarrow \mathsf{R}^{\geq 0}$ is given by $d_\mathsf{D}(x,y) = v_\mathsf{D}(x \vee y) - v_\mathsf{D}(x \wedge y) = m_{\Sigma_\mathsf{D}}([O,x \vee y]) - m_{\Sigma_\mathsf{D}}([O,x \wedge y])$.

The *greatest lower bound* of all path $p_{ab}(.)$ lengths we call *distance* between a and b, symbolically $d_X(a,b)$. That is, distance $d_X(a,b)$ is the length of a shortest path from $a \in X$ to $b \in X$. Note that uniqueness of number $d_X(a,b)$ does not imply uniqueness of the shortest path between a and b since more than one path can have the same (shortest) length. In conclusion, a metric space (X,d_X) emerges from measurable paths between set X elements.

Definition 2.2. A subset $V \subseteq X$ in a metric space (X,d_X) is called *convex* if and only if it includes all shortest paths between set V elements. ■

In the context of this work, a convex set is interpreted as an *information granule* [50].

We define distance(s) between sets in a metric space (X,d_X), next.

2.2. Metrics in a σ-algebra

The previous section has detailed how a measure space $(X,\Sigma_X, m_{\Sigma_X})$ gives rise to a metric space (X,d_X). This section presents three metrics in a σ-algebra Σ_X.

Proposition 2.3. Function d_a: $\Sigma_X \times \Sigma_X \rightarrow \mathsf{R}^{\geq 0}$ such that $d_a(A,B) = 0 \Leftrightarrow A = B$, moreover $d_a(A,B) = \frac{1}{|A||B|}\sum_{\mathrm{i,j}} d_X(a_\mathrm{i},b_\mathrm{j})$ is a metric. ■

The proof of Proposition 2.3 is shown in the Appendix.

We remark that Proposition 2.3 regards only sets of finite cardinality. More specifically, $|A|$ in Proposition 2.3 denotes the *cardinality* of finite set A. In other words, $|A|$ denotes the (integer) number of elements contained in set A, e.g. $|A=\{a,b,c\}| = 3$. Moreover, expression $\sum_{\mathrm{i,j}} d_X(a_\mathrm{i},b_\mathrm{j})$ in Proposition 2.3 is a simplification for $\sum_{a_\mathrm{i}\in A}[\sum_{b_\mathrm{j}\in B} d_X(a_\mathrm{i},b_\mathrm{j})]$.

The following two metrics employ *unary* operations $\vee S$ and $\wedge S$, which equal, respectively, the supremum and the infimum of a set S of real numbers. Note that expressions $\vee S$ and $\wedge S$ are simplifications for expressions $\underset{x\in S}{\vee} S$ and $\underset{x\in S}{\wedge} S$, respectively – For a finite set S, numbers $\vee S$ and $\wedge S$ equal, respectively, the maximum and minimum number in S.

Proposition 2.4. Function d_M: $\Sigma_X\times\Sigma_X\to\mathsf{R}^{\geq 0}$ such that $d_M(A,B) = 0 \Leftrightarrow A = B$, moreover $d_M(A,B)= \underset{\mathrm{i}}{\vee}\underset{\mathrm{j}}{\vee} d_X(a_\mathrm{i},b_\mathrm{j})$ is a metric. ■

The proof of Proposition 2.4 is shown in the Appendix.

We remark that expression $\underset{\mathrm{i}}{\vee}\underset{\mathrm{j}}{\vee} d_X(a_\mathrm{i},b_\mathrm{j})$ in Proposition 2.4 is a simplification for expression $\underset{a_\mathrm{i}\in A}{\vee}\{\underset{b_\mathrm{j}\in B}{\vee}\{d_X(a_\mathrm{i},b_\mathrm{j})\}\}$. Consider the following condition.

Condition 2.5. Let A, B, C be sets in a metric space (X,d_X), and let $a_\mathrm{i}\in A$, $b_\mathrm{j}\in B$, $c_\mathrm{k}\in C$. For an index k suppose both $\exists \mathrm{I_k}$: $\mathrm{I_k}$ = $\arg\{\underset{\mathrm{i}}{\wedge} d_X(c_\mathrm{k},a_\mathrm{i})\}$ and $\exists \mathrm{J_k}$: $\mathrm{J_k}$ = $\arg\{\underset{\mathrm{j}}{\wedge} d_X(c_\mathrm{k},b_\mathrm{j})\}$. Then we define "Condition 2.5" as follows $d_X(a_{\mathrm{I_k}},b_{\mathrm{J_k}}) \leq \max\{\underset{\mathrm{i}}{\vee}\underset{\mathrm{j}}{\wedge} d_X(a_\mathrm{i},b_\mathrm{j}), \underset{\mathrm{j}}{\vee}\underset{\mathrm{i}}{\wedge} d_X(b_\mathrm{j},a_\mathrm{i})\}$. ■

Proposition 2.6. Condition 2.5 is *sufficient* for a metric function d_H: $\Sigma_X\times\Sigma_X\to\mathsf{R}^{\geq 0}$ given by $d_H(A,B)= \max\{\underset{\mathrm{i}}{\vee}\underset{\mathrm{j}}{\wedge} d_X(a_\mathrm{i},b_\mathrm{j}), \underset{\mathrm{j}}{\vee}\underset{\mathrm{i}}{\wedge} d_X(b_\mathrm{j},a_\mathrm{i})\}$. ■

The proof of Proposition 2.6 is shown in the Appendix.

The metrics above are different from other ones between graphs [26], [33] in that the latter quantify structural dissimilarity between graphs, whereas the metrics here quantify distance between graphs. More specifically, metric $d_H(.,.)$ is a generalization of the *Hausdorf* metric, the latter is typically defined in R^N [13], [35]. Moreover, metric d_a is applicable solely to sets of finite cardinality, whereas the other two metrics d_M and d_H are applicable between any sets.

2.3. Examples

In this section we demonstrate computation of metrics $d_M(.,.)$ and $d_H(.,.)$ including also geometric interpretations.

Example 2.7.

Consider intervals $[a,b]$ and $[c,d]$ on the real line (Fig. 1). Sufficient Condition 2.5 holds, therefore it follows

$d_M([a,b],[c,d]) = \max\{|a-d|, |b-c|\}$, for $[a,b] \neq [c,d]$, and

$d_H([a,b],[c,d]) = \max\{|a-c|, |b-d|\}$. ■

Example 2.8.

We represent a circle in the normed linear space R^N by a pair $(\mathbf{c},r)$, where $\mathbf{c}\in\mathsf{R}^N$ and $r\geq 0$ is the (circle) radius. For instance, Fig. 2 shows circles $(\mathbf{c}_A,r_A)$ and $(\mathbf{c}_B,r_B)$ on the plane. A vector $\mathbf{r}$, on the line defined by the centers of circles $(\mathbf{c}_A,r_A)$ and $(\mathbf{c}_B,r_B)$, is given by $\mathbf{r}(\lambda) = \mathbf{c}_A+\lambda(\mathbf{c}_B-\mathbf{c}_A)$, $\lambda\in\mathsf{R}$. For intervals on the line $\mathbf{r}(\lambda)$ sufficient Condition 2.5 holds. Therefore, it follows

$d_M((\mathbf{c}_A,r_A),(\mathbf{c}_B,r_B)) = r_A+||\mathbf{c}_A-\mathbf{c}_B||+r_B$, for $(\mathbf{c}_A,r_A) \neq (\mathbf{c}_B,r_B)$, and

$$d_H((\mathbf{c}_A,r_A),(\mathbf{c}_B,r_B)) = \max\{||(\mathbf{c}_A - \frac{\mathbf{c}_B-\mathbf{c}_A}{||\mathbf{c}_B-\mathbf{c}_A||}r_A) - (\mathbf{c}_B - \frac{\mathbf{c}_B-\mathbf{c}_A}{||\mathbf{c}_B-\mathbf{c}_A||}r_B)||,\ ||(\mathbf{c}_A + \frac{\mathbf{c}_B-\mathbf{c}_A}{||\mathbf{c}_B-\mathbf{c}_A||}r_A) - (\mathbf{c}_B + \frac{\mathbf{c}_B-\mathbf{c}_A}{||\mathbf{c}_B-\mathbf{c}_A||}r_B)||\} = \max\{|r_A+||\mathbf{c}_A-\mathbf{c}_B||-r_B|, |r_B+||\mathbf{c}_A-\mathbf{c}_B||-r_A|\},$$

where $||.||$ denotes the norm of its vector operand, and $|.|$ denotes the absolute value of its real number operand. ■

3. Fuzzy lattices and useful extensions

This section introduces useful functions based on the theory of fuzzy lattices [36], [37], [40], [42] summarized below including novel extensions.

3.1. Power-lattices

Lattice theory was compiled creatively by Garrett Birkhoff [3]. This section considers the power-set 2^{L} of a lattice $(\mathsf{L},\leq)$. Next, we introduce a *binary relation* $\leq\subseteq 2^{\mathsf{L}}\times 2^{\mathsf{L}}$ such that for $U= \{u_1,\ldots,u_I\}$ and $W= \{w_1,\ldots,w_J\}$ in 2^{L} it is $U\leq W$ if and only if $\forall i\in\{1,\ldots,I\}, \exists j\in\{1,\ldots,J\}: u_i\leq w_j$. It can be shown immediately that the aforementioned binary relation, first, is a partial order and, second, it is a lattice order. In conclusion, the *power-lattice* $(2^{\mathsf{L}},\leq)$ emerges with $U\wedge W = \bigcup_{i,j}\{u_i \wedge w_j\}$ and $U\vee W = \bigcup_{i,j}\{u_i \vee w_j\}$.

3.2. Fuzzy lattices

A *fuzzy set* is a pair (U,m), where U is a *universe of discourse* and m is a *membership* function m: $U\to[0,1]$. In particular, the *core* (of fuzzy set m) is the set $C\subseteq U$ for which $\sup_{x\in U} m(x)$ is attained. *Fuzzy lattices* emerged in mathematics as well as in computational intelligence [36] by fuzzifying the binary relation "≤"in a (crisp) lattice as follows.

Definition 3.1. A *fuzzy lattice* is a triple $(\mathsf{L},\leq,m)$, where $(\mathsf{L},\leq)$ is a crisp lattice and $(\mathsf{L}\times\mathsf{L},m)$ is a fuzzy set such that $m(x,y)=1$ if and only if $x\leq y$. ■

We remark that function m: $U\to[0,1]$ in Definition 3.1 is interpreted as a *weak* (*fuzzy*) *partial order* relation in the sense that both $m(x,y)=1$ and $m(y,z)=1$ imply $m(x,z)=1$, whereas if either $m(x,y)\neq 1$ or $m(y,z)\neq 1$ then $m(x,z)$ could be any number in the interval $[0,1]$. A (complete) lattice can be fuzzified by an *inclusion measure* function defined next.

Definition 3.2. Let $(\mathsf{L},\leq)$ be a complete lattice with least element O. An *inclusion measure* is a function σ: $\mathsf{L}\times\mathsf{L}\to[0,1]$, which satisfies conditions: I0) $\sigma(u,O)=0$, $u\neq O$; I1) $\sigma(u,u) = 1$, $\forall u\in\mathsf{L}$; I2) $u\wedge w < u \Rightarrow \sigma(u,w) < 1$; I3) $u \leq w \Rightarrow \sigma(x,u) \leq \sigma(x,w)$ (The Consistency Property). ■

We remark that $\sigma(x,y)$ is interpreted as a (fuzzy) degree of inclusion of x in y. Therefore, notations $\sigma(x,y)$ and $\sigma(x\leq y)$ are used interchangably. If σ: $\mathsf{L}\times\mathsf{L}\to[0,1]$ is an inclusion measure in lattice $(\mathsf{L},\leq)$ then $(\mathsf{L},\leq,\sigma)$ is a fuzzy lattice [36], [42]. A couple inclusion measures can be defined based on a *positive valuation* function as follows [36], [42], [52].

Theorem 3.3. If v: $\mathsf{L}\to\mathsf{R}^{\geq 0}$ is a positive valuation in a complete lattice $(\mathsf{L},\leq)$, with $v(O)=0$, then both functions $s(x,u) = v(x\wedge u)/v(x)$ and $k(x,u) = v(u)/v(x\vee u)$ are inclusion measures. ■

We point out that Theorem 3.3 calls for a *nonnegative* positive valuation function v.

A novel inclusion measure can be introduced in a power-lattice as follows.

Proposition 3.4. Let function σ_V: $\mathsf{L}\times\mathsf{L}\to[0,1]$ be an inclusion measure in a lattice $(\mathsf{L},\leq)$. Then function σ: $2^{\mathsf{L}}\times 2^{\mathsf{L}}\to[0,1]$ given by the *convex combination* $\sigma(\{u_1,\ldots,u_I\}=U\leq W=\{w_1,\ldots,w_J\}) = \lambda_1 \max_j \sigma_V(u_1\leq w_j)+\ldots+\lambda_I \max_j \sigma_V(u_I\leq w_j)$ is an inclusion measure. ■

The proof of Proposition 3.4 is shown in the Appendix.

We remark that by "convex combination" we mean a set $\lambda_1,\ldots,\lambda_I$ of positive numbers such that $\lambda_1+\ldots+\lambda_I = 1$.

3.3. Similarity measure functions

Various similarity measures are employed/defined in applications [11], [53], [55]. A novel definition is proposed next.

Definition 3.5. A *similarity measure* is a function μ: $U\times U\to[0,1]$, which satisfies conditions: S1) $\mu(x,y) = 1 \Leftrightarrow x = y$; S2) $\mu(x,y) = \mu(y,x)$. ■

We remark that the advantage of our proposed Definition 3.5 is that it retains rigorously the essentials of a "common sense" notion of similarity while avoiding redundancies.

We define a *similarity space* as a pair (U,μ) including a non-empty set U and a similarity measure function μ: $U\times U\to[0,1]$. The following proposition introduces a similarity measure in a (complete) lattice based on an inclusion measure function σ.

Proposition 3.6. Let $(\mathsf{L},\leq)$ be a lattice with an inclusion measure σ: $\mathsf{L}\times\mathsf{L}\to[0,1]$. Then, function μ_σ: $\mathsf{L}\times\mathsf{L}\to[0,1]$ given by $\mu_\sigma(x,y) = \dfrac{\sigma(x\leq y)+\sigma(y\leq x)}{2}$ is a similarity measure. ■

The proof of Proposition 3.6 is shown in the Appendix.

The following proposition introduces a similarity measure, in a metric space.

Proposition 3.7. Let function d: $U\times U\to\mathsf{R}^{\geq 0}$ be a metric. Then, function μ_{d}: $U\times U\to[0,1]$ given by $\mu_{\mathrm{d}}(x,y) = \dfrac{1}{1+d(x,y)}$ is a similarity measure. ■

The proof of Proposition 3.7 is shown in the Appendix.

3.4. Practical relevance

A metric space emerges below by adding up the weights of links and nodes along shortest paths in a weighted master-graph $\mathsf{M}= (V,E)$, where $V= \{v_1,\ldots,v_N\}$ is the (non-empty) finite set of *vertices* (or, equivalently *nodes*) and $E\subseteq V\times V$ is the set of *edges* (or, equivalently, *links*).

A *partition*, also called *equivalence relation*, P of the set V of vertices in a master-graph is a (finite) collection of subsets $P_1,\ldots,P_N$, namely *parts*, of V such that both $P_i\cap P_j = \{\}$ for $i\neq j$ and $P_1\cup\ldots\cup P_N = V$ for an integer number N. Apparently, based on shortest path lengths, we can define different metric distances $d_a(.,.)$, $d_M(.,.)$, and $d_H(.,.)$ between *parts* of partitions, where a *part* represents a subgraph − Note that different authors have already proposed distances beyond space R^N, e.g. in metric spaces for queries [64], in spaces of convex /concave bodies for optimisation [7], etc. An additional, useful function is presented next.

Consider the family Π_V of all partitions of a set V. It is known that $(\Pi_V,\leq)$ is a lattice-ordered by (partitions') refinement [56]. More specifically, lattice $(\Pi_V,\leq)$ is a sublattice of power-lattice $(2^{\mathsf{L}},\leq)$, where $(\mathsf{L}=2^V,\leq)$ is the power-set (lattice) of V.

In this work we have considered (in a master-graph) inclusion measure "type" functions defined as in Proposition 3.4 with $\lambda_1=\ldots=\lambda_I$ for the following choices of σ_V:

1) $\sigma_{V1}(P_i\leq Q_j) = \dfrac{|P_i \wedge Q_j|}{|P_i|}$, 2) $\sigma_{V2}(P_i\leq Q_j) = \dfrac{|Q_j|}{|P_i \vee Q_j|}$, and 3) $\sigma_{V3}(P_i\leq Q_j) = \dfrac{|P_i \wedge Q_j|}{|P_i \vee Q_j|}$, where the unary operator $|.|$ returns the cardinality of its (set) operand; e.g. $|P_1=\{a,b,c\}| = 3$.

It turns out that only functions $\sigma_{V1}(.,.)$ and $\sigma_{V2}(.,.)$ above are inclusion measures, according to Theorem 3.3; whereas, function $\sigma_{V3}(.,.)$ is not an inclusion measure as demonstrated by the following counter-example.

Consider the (totally-ordered) lattice $(\mathsf{R},\leq)$ of real numbers. For $1 = u \leq w = 2$ it is straightforward to confirm that the Consistency Property $u \leq w \Rightarrow \sigma(x,u) \leq \sigma(x,w)$ of Definition 3.2 does not hold for $x = 0.5$, because $0.5\wedge1/0.5\vee1 = 0.5 > 0.25 = 0.5\wedge2/0.5\vee2$. Hence, function $\sigma_{V3}(.,.)$ is not an inclusion measure. Nevertheless, function $\sigma_{V3}(.,.)$ has produced the best experimental results as demonstrated below.

4. A fuzzy lattice reasoning (FLR) neural network

Fuzzy Lattice Reasoning (FLR) has emerged as an enhanced version of the learning algorithm employed by fuzzy neural network σ-FLN(MAP) [42]. The latter, in turn, has emerged as a lattice data domain enhancement of neural network fuzzy-ART(MAP) [40].

The operation of FLR was originally based on an *inclusion measure* function [37], [39], [42] towards inducing descriptive, decision-making knowledge (rules). Lately, there was an effort to extend FLR in space R^N based on a "non-rigorously defined" similarity measure function [12]. Inspired from the latter we introduce, in the following, a FLR version for clustering in a graph, based on a rigorously defined similarity measure function.

4.1. The agglomerative similarity measure FLR (asmFLR) algorithm

The *agglomerative similarity measure FLR* (*asmFLR*) algorithm for master-graph partitioning by clustering is presented in Fig. 3 for neural computing subgraphs, namely *granules* or, equivalently, *clusters*. Data processing by *asmFLR* repeats, conditioned on a user-defined *Assimilation Condition* (Fig. 3, Step-1), a number of cycles. Each aforementioned cycle carries out, in parallel, "batch processing" computations. Overall, it can be claimed that the *asmFLR* algorithm carries out *Lattice Computing* (LC), the latter is defined as lattice-theory-based Computational Intelligence [23].

4.2. Algorithm asmFLR details

The *asmFLR* may set out learning without *a priori* knowledge; however, a priori knowledge can be supplied in the form of an initial set of subgraphs/clusters (Fig. 3, Step-0). In particular, for n=N it follows that each master-graph node constitutes a (trivial) cluster.

There can be different user-defined *Assimilation Conditions* (Fig. 3, Step-1). For instance, a naive *Assimilation Condition* could be "n>1" meaning that clustering proceeds until all master-graph nodes are put in one cluster. Another *Assimilation Condition* may define a maximum threshold size for a computed cluster, etc.

Generalization can be effected as follows. A cluster $Q\subseteq V$ in a a weighted master-graph $\mathsf{M} = (V,E)$ defines a fuzzy set $(\Sigma_X, \mu_d(P;Q))$ such that cluster Q corresponds to the *core* of fuzzy set $(\Sigma_X, \mu_d(P;Q))$ – Note that in notation "$\mu_d(P;Q)$" symbol "P" denotes a variable, whereas symbol "Q" denotes a parameter. Hence, generalization becomes feasible beyond core Q.

We remark that the original *FLR* classifier, which employs an inclusion measure function, supports two different modes of reasoning, namely *Generalized Modus Ponens* and *Reasoning by Analogy* [42]. None of the aforementioned modes of reasoning is supported by *asmFLR* since the latter is a scheme for clustering based on a similarity measure function. Next, we compute the complexity of *asmFLR*.

Algorithm *asmFLR* (Fig. 3) includes a number of $O(N)$ cycles. Each cycle computes $O(N^2)$ similarity measure function values. Moreover, each of the latter (values) requires the length (i.e. the metric distance) of the shortest path between two master-graph nodes; nevertheless, there is no additional computational overhead since all aforementioned metric distances are computed once, in a data preprocessing step. It follows that the learning (clustering) complexity of *asmFLR* is cubic $O(N^3)$ in the number "N" of master-graph nodes.

4.3. Comparative discussion

There are inherent similarities as well as substantial differences between *asmFLR* and *FLR*. More specifically, a cluster computed by either algorithm corresponds to the core of a fuzzy set. Nevertheless, a cluster for *FLR* is a N-tuple FIN in space R^N, the latter is the Cartesian product of N totally-ordered lattices R [42]; whereas, the *asmFLR* here is applied in the partially-ordered lattice $(\Pi_V,\leq)$ of partitions of the set $V=\{v_1,\ldots,v_N\}$ of vertices in a weighted master-graph $\mathsf{M}= (V,E)$. Moreover, *asmFLR* is a data-order-independent extension of the data-order-dependent *FLR* such that *asmFLR* is based on a *similarity measure* function, whereas *FLR* is based on an *inclusion measure* function. In addition, the *FLR* learns rapidly with complexity $\mathsf{O}(n)$ in the number n of data, whereas the learning complexity of *asmFLR* is $\mathsf{O}(N^3)$ in the number N of master-graph nodes as shown above.

5. Comparative experimental results

We applied (neural) algorithm *asmFLR*, comparatively, on metric spaces emerged from master-graphs as described above. Master-graphs were generated as described next.

5.1. Artificial data generation and preprocessing

A user defined a number of parameters for generating, randomly, a *dataset*. The aforementioned parameters included: the number M of master-graphs in a dataset, the number C of *graph-clusters* in a master-graph, both the minimum number $v_{\min}$ and the maximum number $v_{\max}$ of vertices per graph-cluster and, finally, both an *intra-connection ratio* $r_{\text{in}}\in[0,1]$ and an *inter-connection ratio* $r_{\text{out}}>0$. A single master-graph was generated as detailed next.

For a graph-cluster $c\in\{1,\ldots,C\}$, a number $v_c\in[v_{\min},v_{\max}]$ of vertices was drawn randomly. Then, we computed the number n_{in} of *intra-cluster* links between a vertex and different ones in graph-cluster c such that n_{in} equals the integer nearest to real number $r_{\text{in}}(v_c\text{-}1)$. Next, we randomly selected (eligible) intra-cluster links from the list of all links in graph-cluster c. Hence, a total number C of graph-clusters were computed.

The graph-clusters computed above were connected by randomly generated *inter-cluster* links as follows. We computed the (constant) number n_{out} of links between a vertex in graph-cluster $c\in\{1,\ldots,C\}$ with vertices in different graph-clusters such that n_{out} equals the smallest integer above real number $r_{\text{in}}(v_c\text{-}1)r_{\text{out}}$. In addition, we required vertex intra-connectivity (within a cluster) to be larger than the corresponding vertex inter-connectivity (with different clusters) in order to produce clearly separated graph-clusters (in a master-graph) – Note that different authors often generate inter-cluster links “probabilistically” [49] resulting in, unfortunately, “not clearly separated” graph-clusters. Finally, we randomly drew (eligible) inter-cluster links from the list of all links in the master-graph. Hence, a master-graph was generated.

We repeated the above procedure M times. In conclusion, one *dataset* was generated including M (different) master-graphs. The next example demonstrates dataset generation.

Example 5.1.

Fig. 4 (as well as Fig. 5) displays a *dataset* with M=2 different master-graphs; each of the aforementioned master-graphs includes C=5 identical graph-clusters with v_{min}=3, v_{max}=5. Note that it is r_{in}=1.0 in both Fig. 4 and Fig. 5, i.e. each graph-cluster is completely intra-connected. A user defined r_{out}=0.3 for the dataset in Fig. 4, and r_{out}=0.6 for the dataset in Fig. 5.

Due to the master-graph generation procedure detailed above, it turns out that the (actual) r_{out} of a specific vertex is, typically, different than the user-defined r_{out}. Therefore, for each vertex, we computed the (actual) r_{out} as the ratio of (number of) inter-cluster links over (number of) intra-cluster links. Finally, we computed both the average r_{out} and the corresponding standard deviation in a master-graph in Fig. 4 as well as in Fig. 5 – It turns out that the average r_{out} is different than the user-defined r_{out}, as expected. ■

As soon as a master-graph was generated, we used (in a data-preprocessing step) Floyd's algorithm [17] in order to compute the distances between any two nodes in the master-graph. We point out that Floyd's algorithm receives as input the corresponding master-graph's "adjacency matrix" and outputs the required "distance matrix" with cubic complexity $O(N^3)$ in the number N of master-graph nodes. Note that in our experiments we considered master-graph links of "unit" length, moreover the vertices in a master-graph did not have a weight.

5.2. Comparative experiments

We generated 5 (different) datasets **ds1**, **ds2**, **ds3**, **ds4**, **ds5**, respectively, for $r_{\text{out}} \in \{0.5, 0.7, 0.9, 1.0, 1.2\}$. A master-graph in a dataset included C=10 (identical) graph-clusters with v_{min}= 5 and v_{max}= 15. Table 1 summarizes various dataset statistics including average cluster size as well as the corresponding standard deviation 11.6 and 3.565, respectively. Table 1 also shows the user-defined intra-connection ratio r_{in} = 1, that is each vertex in a graph-cluster was connected to all other vertices. The next column in Table 1 indicates the user-defined inter-connection ratio r_{out}. The last two columns in Table 1 display the actual (average) r_{out} as well as the corresponding standard deviation in a dataset.

In a data-preprocessing step we calculated (and stored) the metric distances between all pairs of nodes in a master-graph, for fast access.

We employed algorithm *asmFLR* using a similarity measure $\mu_d(.,.)$ based on three different metrics, namely $d_a(.,.)$, $d_M(.,.)$, and $d_H(.,.)$. Notation *asmFLR*(d_X) below means algorithm *asmFLR* based on metric d_X, $X \in \{a, M, H\}$. For comparison, we implemented and applied four alternative graph clustering algorithms, namely *MajorClust* [61], *MinCutTrees* [10], [16], *Modularity* [49], and *SingleLink* [30]. Where applicable, the aforementioned algorithms were executed until 10 graph-clusters were computed.

Table 2 through Table 6 summarize our experimental results regarding datasets **ds1**, **ds2**, **ds3**, **ds4**, and **ds5**, respectively. Algorithms *MajorClust* and *MinCutTrees* terminated by computing a small number $C\leq3$ of graph-clusters. The other algorithms terminated when a total number of C=10 graph-clusters were computed as shown in the second column of Tables 2 through Table 6. The next four columns in Tables 2 through Table 6 display size statistics regarding graph-clusters computed by an algorithm. Symbol "NaN", i.e. Not-a-Number, appears in the std (standard deviation) column in a Table when a single cluster was computed, hence computation of std is meaningless. Column "#trivials" in a Table indicates the (average) number of trivial graph-clusters in a total number of C=10 graph-clusters computed by an algorithm. The last three columns in a Table show the corresponding values of (graph-clustering) indices *Purity* [67], *Entropy* [68], and *similarity measure* $\mu_\sigma(.,.)$ – Note that we used a different *similarity measure* function $\mu_\sigma(.,.)$ as an index than the *similarity measure* function $\mu_d(.,.)$ used by algorithm *asmFLR* towards producing unbiased results.

Fig. 6, as well as Fig. 7, demonstrates comparatively the performance of algorithms *asmFLR*(d_a) and *Modularity* using three indices, namely *Purity* (curve in light gray color), *Entropy* (curve in dark gray color), and index $\mu_\sigma(.,.)$ (curve in black color), versus a (decreasing) number of computed graph-clusters for dataset **ds1**. First, Fig. 6 (a) and (b) correspond to algorithms *asmFLR*(d_a) and *Modularity*, respectively, where both aforementioned figures use index $\mu_\sigma(.,.)$ based on inclusion measure $\sigma_{V2}(P_i\leq Q_j)= |Q_j|/|P_i\vee Q_j|$ – Note that similar curves were obtained by the aforementioned alternative graph clustering algorithms, also for $\sigma_{V1}(P_i\leq Q_j) = |P_i\wedge Q_j|/|P_i|$, for other datasets as well. Second, Fig. 7 (a) and (b) correspond to algorithms *asmFLR*(d_a) and *Modularity*, respectively, where both aforementioned figures use index $\mu_\sigma(.,.)$ based on $\sigma_{V3}(P_i\leq Q_j)= |P_i\wedge Q_j|/|P_i\vee Q_j|$ – Note that similar curves were obtained also by the aforementioned alternative graph clustering algorithms, for other datasets as well.

5.3. Discussion of the results

Our computational experiments have clearly demonstrated a favorable comparison for our proposed *asmFLR* algorithms. In addition, our experimental work has confirmed the value of our proposed index function $\mu_\sigma(.,.)$ as detailed next.

First, an index value (in Fig. 6 and Fig. 7) was calculated by comparing a partition, computed by a graph clustering algorithm, to an "optimum" partition – The latter was the one, which included the C=10 graph-clusters used for generating a master-graph as described above. Note that index *Purity*, by definition, indicates the average largest percentage of a computed cluster inside an "optimum" cluster; hence, larger values of *Purity* are preferable. Whereas, index *Entropy*, by definition, is (roughly) the complement of index *Purity*; hence, smaller values of *Entropy* are preferable.

A disadvantage of both indices *Purity* and *Entropy* is that "more" graph-clusters may be characterized by better (index) values than the optimum number of C=10 graph-clusters as demonstrated in both Fig. 6 and Fig. 7. Whereas, index $\mu_\sigma(.,.)$ has a global optimum value at C=10 graph-clusters as demonstrated in both Fig. 6 and Fig. 7. Note in both Fig. 6 and Fig. 7 that the *Purity* index drops sharply for fewer than C=10 graph-clusters, as expected, since "10" is the optimum number of clusters. Moreover, the *Purity* index in both Fig. 6 and Fig. 7 is less than 1 at C=10 due to the fact that the computed 10 clusters are not identical to the original ("optimal") ones in a dataset. An inherent drawback of index *Purity* is its trustworthy capacity mainly for comparing partitions of the same cardinality. Whereas, by definition, index $\mu_\sigma(.,.)$ is also reliable for comparing partitions of different cardinalities.

Index $\mu_\sigma(.,.)$ based on $\sigma_{V3}(P_i \leq Q_j) = |P_i \wedge Q_j|/|P_i \vee Q_j|$ (Fig. 7) is preferable to index $\mu_\sigma(.,.)$ based on $\sigma_{V2}(P_i \leq Q_j) = |Q_j|/|P_i \vee Q_j|$ (Fig. 6) because the index $\mu_\sigma(.,.)$ curves in Fig. 7 have a "sharper" global maximum than the corresponding curves in Fig. 6 − Hence, the graph-clustering algorithms in Tables 2 through 6 were arranged in the order of "decreasing" values of index $\mu_\sigma(.,.)$ based on $\sigma_{V3}(P_i \leq Q_j) = |P_i \wedge Q_j|/|P_i \vee Q_j|$. Note also that, in our computational experiments, only index $\mu_\sigma(.,.)$ based on σ_{V3} has retained nearly the same ordering in the performance of the alternative four algorithms *Modularity*, *MajorClust*, *MinCutTrees*, and *SingleLink* as the popular *Purity* index; whereas, index $\mu_\sigma(.,.)$ based on either σ_{V1} or σ_{V2} has produced different orderings. The latter is one more reason for preferring σ_{V3}. Furthermore, note that function σ_{V3}, also known in the literature as *Jaccard* (similarity measure) *coefficient*, has demonstrated a superior performance as compared to seven well-known similarity measure functions in a series of computational experiments elsewhere [45]. The extensive experimental evidence presented in this work has confirmed the superiority of the *Jaccard coefficient* σ_{V3}.

Second, algorithm *asmFLR*(d_a) invariably tops all the other algorithms in Tables 2 through 6. In the second and third places appear algorithms *Modularity* and *asmFLR*(d_M). In the fourth and fifth place appear algorithms *MajorClust* and *asmFLR*(d_H). Finally, algorithm *MinCutTrees* is always in the sixth place, whereas algorithm *SingleLink* is always in the last (seventh) place in all Tables 2 through 6. It is remarkable that the aforementioned arrangement of algorithms essentially remains the same (with minor differences) in Tables 2 through 6, for either index *Purity* or *Entropy*. Therefore, based on experimental evidence, it can be claimed that algorithm *asmFLR*(d_a) approaches the "optimum" graph clustering better than any other graph clustering algorithm in this work. Furthermore, it is a remarkable advantage that algorithm *asmFLR*(d_a) typically avoids computing trivial graph-clusters.

We, further, pursued computation of *convex* subgraphs in a master-graph. In a large number of experiments, using various indices of performance, we recorded a "non-statistically significant" deterioration of performance for the same number of computed subgraphs. In addition, the time required for computing convex subgraphs grew significantly (i.e. exponentially) in the number of nodes. Therefore, we conclude that convex subgraphs do not improve performance here. The explanation is that the artificial master-graphs, generated in the context of this work, as described above, were not convex in the first place.

6. Conclusion

This work has introduced novel mathematical perspectives and tools for clustering in a general metric space. In conclusion, the *agglomerative similarity measure FLR* (*asmFLR*) neural computing algorithm was introduced for partitioning-by-clustering. In addition, our work introduced a novel index for evaluating the quality of clustering.

A metric space emerged here from a weighted graph. Experimental results have confirmed, comparatively, the viability of our proposed techniques/tools on artificial data (graphs) generated randomly. A real-world application of practical interest is to partition a master-graph, whose link-weights represent Web-traffic, towards Web-navigation support [14], [51], [60], [63]. The preliminary experimental work in [43] will be extended in the future using the tools here. Future work may also consider alternative weighted graph partitioning problems including an approximate solution to the "minimum cut" problem [16].

Appendix

This Appendix lists useful definitions. It also includes the proofs of six novel propositions.

A.1. Lattice theory, basic definitions

For elementary definitions regarding lattice theory the reader may refer to [3], [24], and [39]. The following definition is important in the sequel.

Definition A.1. A *positive valuation* in a lattice $(\mathsf{L},\leq)$ is a real function v: $\mathsf{L}\rightarrow\mathsf{R}$, which satisfies both $v(x)+v(y) = v(x\wedge y)+v(x\vee y)$ and $x<y \Rightarrow v(x)<v(y)$. ■

A positive valuation v: $\mathsf{L}\rightarrow\mathsf{R}$ in a lattice $(\mathsf{L},\leq)$ implies a *metric* function d: $\mathsf{L}\times\mathsf{L}\rightarrow\mathsf{R}^{\geq 0}$ given by

$$d(a,b) = v(a\vee b) - v(a\wedge b) \qquad \text{(E1)}$$

– For definition of a metric function see below.

It is remarkable that the latter metric (distance) is used implicitly in the "graph literature" without reference to lattice theory. More specifically, (metric) distance $d_{\mathrm{mcs1}}(G_0,G_1) = |V_0| + |V_1| - 2|V_{01}|$ is used in [33], which (distance d_{mcs1}), in the context of lattice theory, can be produced as follows. Function $v(G(V,E,l)) = v(V) = |V|$ is a positive valuation in the power-set (lattice) of vertices. Hence, $d_{\mathrm{mcs1}}(G_0,G_1) = v(G_0\vee G_1) - v(G_0\wedge G_1) = v(V_0\vee V_1) - v(V_0\wedge V_1)$, where both $V_0\wedge V_1 = V_{01}$ and $v(V_0\vee V_1) + v(V_0\wedge V_1) = v(V_0) + v(V_1)$ hold. It follows, $d_{\mathrm{mcs1}}(G_0,G_1) = v(V_0) + v(V_1) - 2v(V_0\wedge V_1) = |V_0| + |V_1| - 2|V_{01}|$.

A.2. Measure spaces, basic definitions

Consider the following definition [28].

Definition A.2. A σ-algebra Σ_X over a set X is a collection of subsets of X that satisfies:

Σ1) $\varnothing\in\Sigma_X$,

Σ2) $A\in\Sigma_X$ implies $(X-A)\in\Sigma_X$, and

Σ3) for a collection of sets $A_i\in\Sigma_X$ indexed by a countable indexing set D it follows $(\bigcup_{i\in\mathsf{D}} A_i)\in\Sigma_X$. ■

In words, a σ-algebra includes the empty set and it is closed under both complementation and countable (including finite) unions.

A *measure* is a set function m_S: $S \rightarrow \mathsf{R}^{\geq 0}$, which assigns a *size* to every "measurable set" element in S. Note that a measure m_S is required, by definition, to satisfy: 1) $m_S(\emptyset)=0$, and 2) for any countable (including finite) indexing set D, and any collection or pairwise disjoint sets $A_i \in S$ indexed by $i \in \mathsf{D}$ it holds $m_S(\bigcup_{i\in\mathsf{D}} A_i) = \sum_{i\in\mathsf{D}} m_S(A_i)$. In this work S is a σ-algebra, i.e. $S = \Sigma_X$.

A *measure space* $(X, \Sigma_X, m_{\Sigma_X})$ includes a set X, a σ-algebra Σ_X over X, and a measure m_{Σ_X} over Σ_X. We remark that a *probability space* is a measure space such that $m_{\Sigma_X}(X)=1$.

A.3. Metric spaces, basic definition

Consider the following definition.

Definition A.3. A *metric* in a set U is a nonnegative real function d: $U\times U \rightarrow \mathsf{R}^{\geq 0}$, which satisfies the following laws.

M0) $d(x,y) = 0 \Rightarrow x = y$;

M1) $d(x,x) = 0$;

M2) $d(x,y) = d(y,x)$;

M3) $d(x,y) \leq d(x,z)+d(z,y)$ (Triangle Inequality). ■

If only conditions M1) to M3) are satisfied in Definition A.3 then d is called *pseudo-metric*. A *metric space* is a pair (U,d) including both a set U and a metric d: $U\times U \rightarrow \mathsf{R}^{\geq 0}$.

A.4. Proofs of novel propositions

This section presents the proofs of six novel propositions.

Proposition 2.3. Function d_a: $\Sigma_X\times\Sigma_X \rightarrow \mathsf{R}^{\geq 0}$ such that $d_a(A,B) = 0 \Leftrightarrow A = B$, moreover $d_a(A,B)= \frac{1}{|A||B|}\sum_{i,j} d_X(a_i,b_j)$ is a metric.

Proof

Let $A,B,C\in\Sigma_X$. Real function $d_a(A,B)$ is nonnegative. We show next that function $d_a(A,B)$ satisfies the four laws of a metric (in Definition A.3).

M0) and M1) hold by definition.

M2) $d_a(A,B) = \frac{1}{|A||B|}\sum_{\mathrm{i,j}} d_X(a_\mathrm{i},b_\mathrm{j}) = \frac{1}{|B||A|}\sum_{\mathrm{j,i}} d_X(b_\mathrm{j},a_\mathrm{i}) = d_a(B,A)$.

M3) Given $(a_\mathrm{i},b_\mathrm{j})\in A\times B$ it follows $d_X(a_\mathrm{i},b_\mathrm{j}) \leq d_X(a_\mathrm{i},c_\mathrm{k}) + d_X(c_\mathrm{k},b_\mathrm{j})$, for $c_\mathrm{k}\in C$. Summing up for all $c_\mathrm{k}\in C$ it follows $\sum_\mathrm{k} d_X(a_\mathrm{i},b_\mathrm{j}) \leq \sum_\mathrm{k}[d_X(a_\mathrm{i},c_\mathrm{k}) + d_X(c_\mathrm{k},b_\mathrm{j})] \Rightarrow d_X(a_\mathrm{i},b_\mathrm{j}) \leq \frac{1}{|C|}\sum_\mathrm{k}[d_X(a_\mathrm{i},c_\mathrm{k}) + d_X(c_\mathrm{k},b_\mathrm{j})]$. Furthermore, summing up for all pairs $(a_\mathrm{i},b_\mathrm{j})\in A\times B$ it follows $\sum_{\mathrm{i,j}} d_X(a_\mathrm{i},b_\mathrm{j}) \leq \sum_{\mathrm{i,j}}\frac{1}{|C|}\sum_\mathrm{k}[d_X(a_\mathrm{i},c_\mathrm{k}) + d_X(c_\mathrm{k},b_\mathrm{j})]$. Hence,

$$\frac{1}{|A||B|}\sum_{\mathrm{i,j}} d_X(a_\mathrm{i},b_\mathrm{j}) = d_a(A,B) \leq \frac{1}{|A||B||C|}\sum_{\mathrm{i,j,k}}[d_X(a_\mathrm{i},c_\mathrm{k}) + d_X(c_\mathrm{k},b_\mathrm{j})] =$$

$$= \frac{1}{|A||C|}\sum_{\mathrm{i,k}} d_X(a_\mathrm{i},c_\mathrm{k}) + \frac{1}{|C||B|}\sum_{\mathrm{k,j}} d_X(c_\mathrm{k},b_\mathrm{j}) = d_a(A,C) + d_a(C,B).$$

■

Proposition 2.4. Function d_M: $\Sigma_X\times\Sigma_X\to\mathsf{R}^{\geq 0}$ such that $d_M(A,B) = 0 \Leftrightarrow A = B$, moreover $d_M(A,B) = \bigvee_\mathrm{i}\bigvee_\mathrm{j} d_X(a_\mathrm{i},b_\mathrm{j})$ is a metric.

Proof

Let $A,B,C\in\Sigma_X$. Real function $d_M(A,B)$ is nonnegative. We show next that function $d_M(A,B)$ satisfies the four laws of a metric (in Definition A.3).

M0) and M1) hold by definition.

M2) $d_M(A,B) = \bigvee_\mathrm{i}\bigvee_\mathrm{j} d_X(a_\mathrm{i},b_\mathrm{j}) = \bigvee_\mathrm{j}\bigvee_\mathrm{i} d_X(b_\mathrm{j},a_\mathrm{i}) = d_M(B,A)$.

M3) Let $a_\mathrm{i}\in A$, $b_\mathrm{j}\in B$, and $c_\mathrm{k}\in C$. Apparently, $d_X(a_\mathrm{i},b_\mathrm{j}) \leq d_X(a_\mathrm{i},c_\mathrm{k}) + d_X(c_\mathrm{k},b_\mathrm{j})$. Since both inequalities $d_X(a_\mathrm{i},c_\mathrm{k}) < \bigvee_\mathrm{i}\bigvee_\mathrm{k} d_X(a_\mathrm{i},c_\mathrm{k})$ and $d_X(c_\mathrm{k},b_\mathrm{j}) < \bigvee_\mathrm{k}\bigvee_\mathrm{j} d_X(c_\mathrm{k},b_\mathrm{j})$ hold, it follows $d_X(a_\mathrm{i},b_\mathrm{j}) \leq \bigvee_\mathrm{i}\bigvee_\mathrm{k} d_X(a_\mathrm{i},c_\mathrm{k}) + \bigvee_\mathrm{k}\bigvee_\mathrm{j} d_X(c_\mathrm{k},b_\mathrm{j})$. Furthermore, since the latter inequality holds for any $(a_\mathrm{i},b_\mathrm{j})\in A\times B$ it follows $\bigvee_\mathrm{i}\bigvee_\mathrm{j} d_X(a_\mathrm{i},b_\mathrm{j}) \leq \bigvee_\mathrm{i}\bigvee_\mathrm{k} d_X(a_\mathrm{i},c_\mathrm{k}) + \bigvee_\mathrm{k}\bigvee_\mathrm{j} d_X(c_\mathrm{k},b_\mathrm{j})$. Hence, $d_M(A,B) \leq d_M(A,C) + d_M(C,B)$.

■

Proposition 2.6. Condition 2.5 is *sufficient* for a metric function d_H: $\Sigma_X\times\Sigma_X\to\mathsf{R}^{\geq 0}$ given by $d_H(A,B) = \max\{\bigvee_\mathrm{i}\bigwedge_\mathrm{j} d_X(a_\mathrm{i},b_\mathrm{j}), \bigvee_\mathrm{j}\bigwedge_\mathrm{i} d_X(b_\mathrm{j},a_\mathrm{i})\}$.

Proof

Let $A,B,C \in \Sigma_X$. Real function $d_H(A,B)$ is nonnegative. We show next that function $d_H(A,B)$ satisfies the four laws of a metric (in Definition A.3).

M0) $d_H(A,B) = 0 \Rightarrow \max\{\vee_{i}\wedge_{j} d_X(a_i,b_j),\ \vee_{j}\wedge_{i} d_X(b_j,a_i)\} = 0$. Therefore, $\forall a_i \in A$, $\exists b_j \in B$: $d_X(a_i,b_j) = 0$, and $\forall b_j \in B$, $\exists a_i \in A$: $d_X(b_j,a_i) = 0$. Hence, $\forall a_i \in A$, $\exists b_j \in B$ such that $a_i = b_j$, and $\forall b_j \in B$, $\exists a_i \in A$ such that $b_j = a_i$. In conclusion, $A = B$.

M1) $d_H(A,A) = \max\{\vee_{a_i \in A}\wedge_{b_j \in A} d_X(a_i,b_j),\ \vee_{b_j \in A}\wedge_{a_i \in A} d_X(b_j,a_i)\} = 0$.

M2) $d_H(A,B) = \max\{\vee_{i}\wedge_{j} d_X(a_i,b_j),\ \vee_{j}\wedge_{i} d_X(b_j,a_i)\} = \max\{\vee_{j}\wedge_{i} d_X(b_j,a_i),\ \vee_{i}\wedge_{j} d_X(a_i,b_j)\} = d_H(B,A)$.

M3) Let $a_i \in A$, $b_j \in B$, and $c_k \in C$. Consider, first, Condition 2.5 $d_X(c_{K_i},b_{J_i}) \le \max\{\vee_{k}\wedge_{j} d_X(c_k,b_j),\ \vee_{j}\wedge_{k} d_X(b_j,c_k)\}$ and, second, triangle inequality $d_X(a_i,b_j) \le d_X(a_i,c_k) + d_X(c_k,b_j) \Rightarrow \wedge_{j} d_X(a_i,b_j) \le \wedge_{k} d_X(a_i,c_k) + d_X(c_{K_i},b_{J_i})$. Hence, $\vee_{i}\wedge_{j} d_X(a_i,b_j) \le \vee_{i}\wedge_{k} d_X(a_i,c_k) + \vee_{i} d_X(c_{K_i},b_{J_i}) \le \vee_{i}\wedge_{k} d_X(a_i,c_k) + \max\{\vee_{k}\wedge_{j} d_X(c_k,b_j), \vee_{j}\wedge_{k} d_X(b_j,c_k)\}$.

By rotating sets A and B we obtain $\vee_{j}\wedge_{i} d_X(b_j,a_i) \le \vee_{j}\wedge_{k} d_X(b_j,c_k) + \max\{\vee_{k}\wedge_{i} d_X(c_k,a_i),\ \vee_{i}\wedge_{k} d_X(a_i,c_k)\}$. Hence, $\max\{\vee_{i}\wedge_{j} d_X(a_i,b_j),\ \vee_{j}\wedge_{i} d_X(b_j,a_i)\} \le \max\{\vee_{i}\wedge_{k} d_X(a_i,c_k) + \max\{\vee_{k}\wedge_{j} d_X(c_k,b_j),\ \vee_{j}\wedge_{k} d_X(b_j,c_k)\},\ \vee_{j}\wedge_{k} d_X(b_j,c_k) + \max\{\vee_{k}\wedge_{i} d_X(c_k,a_i),\ \vee_{i}\wedge_{k} d_X(a_i,c_k)\}\}$.

We consider four cases, next.

1) Both $\vee_{k}\wedge_{j} d_X(c_k,b_j) \ge \vee_{j}\wedge_{k} d_X(b_j,c_k)$ and $\vee_{k}\wedge_{i} d_X(c_k,a_i) \ge \vee_{i}\wedge_{k} d_X(a_i,c_k)$.

Hence, $\max\{\vee_{i}\wedge_{k} d_X(a_i,c_k) + \vee_{k}\wedge_{j} d_X(c_k,b_j),\ \vee_{j}\wedge_{k} d_X(b_j,c_k) + \vee_{k}\wedge_{i} d_X(c_k,a_i)\} \le \vee_{k}\wedge_{j} d_X(c_k,b_j) + \vee_{k}\wedge_{i} d_X(c_k,a_i)$.

2) Both $\vee_{k}\wedge_{j} d_X(c_k,b_j) \ge \vee_{j}\wedge_{k} d_X(b_j,c_k)$ and $\vee_{k}\wedge_{i} d_X(c_k,a_i) < \vee_{i}\wedge_{k} d_X(a_i,c_k)$.

Hence, $\max\{\vee_{i}\wedge_{k} d_X(a_i,c_k) + \vee_{k}\wedge_{j} d_X(c_k,b_j),\ \vee_{j}\wedge_{k} d_X(b_j,c_k) + \vee_{i}\wedge_{k} d_X(a_i,c_k)\} = \vee_{i}\wedge_{k} d_X(a_i,c_k) + \vee_{k}\wedge_{j} d_X(c_k,b_j)$.

3) Both $\bigvee_{k}\bigwedge_{j} d_X(c_k,b_j) < \bigvee_{j}\bigwedge_{k} d_X(b_j,c_k)$ and $\bigvee_{k}\bigwedge_{i} d_X(c_k,a_i) \geq \bigvee_{i}\bigwedge_{k} d_X(a_i,c_k)$.

Hence, $\max\{\bigvee_{i}\bigwedge_{k} d_X(a_i,c_k) + \bigvee_{j}\bigwedge_{k} d_X(b_j,c_k),\ \bigvee_{j}\bigwedge_{k} d_X(b_j,c_k) + \bigvee_{k}\bigwedge_{i} d_X(c_k,a_i)\} = \bigvee_{j}\bigwedge_{k} d_X(b_j,c_k) + \bigvee_{k}\bigwedge_{i} d_X(c_k,a_i)$.

4) Both $\bigvee_{k}\bigwedge_{j} d_X(c_k,b_j) < \bigvee_{j}\bigwedge_{k} d_X(b_j,c_k)$ and $\bigvee_{k}\bigwedge_{i} d_X(c_k,a_i) < \bigvee_{i}\bigwedge_{k} d_X(a_i,c_k)$,

Hence, $\max\{\bigvee_{i}\bigwedge_{k} d_X(a_i,c_k) + \bigvee_{j}\bigwedge_{k} d_X(b_j,c_k),\ \bigvee_{j}\bigwedge_{k} d_X(b_j,c_k) + \bigvee_{i}\bigwedge_{k} d_X(a_i,c_k)\} = \bigvee_{i}\bigwedge_{k} d_X(a_i,c_k) + \bigvee_{j}\bigwedge_{k} d_X(b_j,c_k)$.

In conclusion, $d_H(A,B) = \max\{\bigvee_{i}\bigwedge_{j} d_X(a_i,b_j),\ \bigvee_{j}\bigwedge_{i} d_X(b_j,a_i)\} \leq \max\{\bigvee_{i}\bigwedge_{k} d_X(a_i,c_k) + \max\{\bigvee_{k}\bigwedge_{j} d_X(c_k,b_j), \bigvee_{j}\bigwedge_{k} d_X(b_j,c_k)\}, \bigvee_{j}\bigwedge_{k} d_X(b_j,c_k) + \max\{\bigvee_{k}\bigwedge_{i} d_X(c_k,a_i), \bigvee_{i}\bigwedge_{k} d_X(a_i,c_k)\}\}$ $= \max\{\bigvee_{i}\bigwedge_{k} d_X(a_i,c_k),\ \bigvee_{k}\bigwedge_{i} d_X(c_k,a_i)\} + \max\{\bigvee_{k}\bigwedge_{j} d_X(c_k,b_j),\ \bigvee_{j}\bigwedge_{k} d_X(b_j,c_k)\} = d_H(A,C) + d_H(C,B)$.

■

Proposition 3.4. Let function σ_V: $\mathsf{L}\times\mathsf{L}\rightarrow[0,1]$ be an inclusion measure in a lattice $(\mathsf{L},\leq)$. Then function σ: $2^{\mathsf{L}}\times2^{\mathsf{L}}\rightarrow[0,1]$ given by the *convex combination* $\sigma(\{u_1,\ldots,u_I\}=U\leq W=\{w_1,\ldots,w_J\}) = \lambda_1 \max\limits_{j} \sigma_V(u_1\leq w_j)+\ldots+\lambda_I \max\limits_{j} \sigma_V(u_I\leq w_j)$ is an inclusion measure.

Proof

First, we prove the following Lemma (by contraposition).

Lemma: $U\wedge W<U \Rightarrow \exists i\in\{1,\ldots,I\}$ such that $\forall j\in\{1,\ldots,J\}$ it is $u_i\wedge w_j<u_i$.

Proof of the Lemma: NOT[$\exists i\in\{1,\ldots,I\}$ such that $\forall j\in\{1,\ldots,J\}$ it is $u_i\wedge w_j<u_i$] $\Rightarrow \forall i\in\{1,\ldots,I\}$, $\exists j\in\{1,\ldots,J\}$ such that $u_i\wedge w_j=u_i$ $\Rightarrow$ $\forall i\in\{1,\ldots,I\}$, $\exists j\in\{1,\ldots,J\}$ such that $u_i\leq w_j$ $\Rightarrow$ NOT[$U\wedge W<U$]. It follows, $U\wedge W<U \Rightarrow \exists i\in\{1,\ldots,I\}$ such that $\forall j\in\{1,\ldots,J\}$ it is $u_i\wedge w_j<u_i$.

Next, we resume the proof of Proposition 3.4. More specifically, we show that function $\sigma(U\leq W)$ satisfies the four laws of an inclusion measure (in Definition 3.2).

I0) $\sigma(U\leq O) = \lambda_1 \max\limits_{j} \sigma_V(u_1\leq O)+\ldots+\lambda_I \max\limits_{j} \sigma_V(u_I\leq O) = \lambda_1 0+\ldots+\lambda_I 0 = 0$.

I1) $\sigma(U\leq U) = \lambda_1 \max\limits_{i} \sigma_V(u_1\leq u_i)+\ldots+\lambda_I \max\limits_{i} \sigma_V(u_I\leq u_i) = \lambda_1+\ldots+\lambda_I = 1$.

I2) From the above Lemma we have $U \wedge W < U \Rightarrow \exists i \in \{1,\ldots,I\}$ such that $\forall j \in \{1,\ldots,J\}$ it is $u_i \wedge w_j < u_i$. Hence, $U \wedge W < U \Rightarrow \exists i \in \{1,\ldots,I\}$ such that $\forall j \in \{1,\ldots,J\}$ it is $\sigma(u_i,w_j)<1$. In conclusion, $\sigma(U \leq W) = \lambda_1 \max\limits_{j} \sigma_V(u_1 \leq w_j) + \ldots + \lambda_I \max\limits_{j} \sigma_V(u_I \leq w_j) < \lambda_1 + \ldots + \lambda_I = 1$.

I3) $U \leq W \Rightarrow \forall i \in \{1,\ldots,I\}, \exists j \in \{1,\ldots,J\}$: $u_i \leq w_j \Rightarrow \sigma_V(x_k \leq u_i) \leq \sigma_V(x_k \leq w_j)$, $\forall k \in \{1,\ldots,K\} \Rightarrow \forall k \in \{1,\ldots,K\}$, $\max\limits_{i} \sigma_V(x_k \leq u_i) \leq \max\limits_{j} \sigma_V(x_k \leq w_j) \Rightarrow \lambda_1 \max\limits_{i} \sigma_V(x_1 \leq u_i) + \ldots + \lambda_K \max\limits_{i} \sigma_V(x_K \leq u_i) \leq \lambda_1 \max\limits_{j} \sigma_V(x_1 \leq w_j) + \ldots + \lambda_K \max\limits_{j} \sigma_V(x_K \leq w_j) \Rightarrow \sigma(X,U) \leq \sigma(X,W)$.

■

Proposition 3.6. Let $(\mathsf{L},\leq)$ be a lattice with an inclusion measure σ: $\mathsf{L}\times\mathsf{L}\to[0,1]$. Then, function μ_σ: $\mathsf{L}\times\mathsf{L}\to[0,1]$ given by $\mu_\sigma(x,y) = \dfrac{\sigma(x \leq y) + \sigma(y \leq x)}{2}$ is a similarity measure.

Proof

We show next that function $\mu_\sigma(x,y)$ satisfies the two laws of a similarity measure (in Definition 3.5).

S1) $\mu_\sigma(x,y) = \dfrac{\sigma(x \leq y) + \sigma(y \leq x)}{2} = 1 \Leftrightarrow \sigma(x \leq y) = 1 = \sigma(y \leq x) \Leftrightarrow x \leq y$ and $y \leq x \Leftrightarrow x = y$.

S2) $\mu_\sigma(x,y) = \dfrac{\sigma(x \leq y) + \sigma(y \leq x)}{2} = \dfrac{\sigma(y \leq x) + \sigma(x \leq y)}{2} = \mu_\sigma(y,x)$.

■

Proposition 3.7. Let function d: $U \times U \to \mathsf{R}^{\geq 0}$ be a metric. Then, function μ_d: $U \times U \to [0,1]$ given by $\mu_d(x,y) = \dfrac{1}{1+d(x,y)}$ is a similarity measure.

Proof

We show next that function $\mu_\sigma(x,y)$ satisfies the two laws of a similarity measure (in Definition 3.5).

S1) $\mu_d(x,y) = \dfrac{1}{1+d(x,y)} = 1 \Leftrightarrow d(x,y) = 0 \Leftrightarrow x = y$.

S2) $\mu_d(x,y) = \dfrac{1}{1+d(x,y)} = \dfrac{1}{1+d(y,x)} = \mu_d(y,x)$.

■

References

[1] M. Bianchini, M. Maggini, L. Sarti, and F. Scarselli, Recursive neural networks for processing graphs with labeled edges: theory and applications, Neural Networks 18 (8) (2005) 1040-1050.

[2] M. Bianchini, M. Gori, L. Sarti, and F. Scarselli, Recursive processing of cyclic graphs, IEEE Trans Neural Networks 17 (1) (2006) 10-18.

[3] G. Birkhoff, Lattice Theory (AMS, Providence, Colloquium Publications 25, 1967).

[4] U. Brandes, M. Gaertler, and D. Wagner, Experiments on graph clustering algorithms, in: G. Di Batista and U. Zwick, eds., (Springer, Heidelberg, LNCS 2832, 2003) 568-579.

[5] F. Buckley and F. Harary, Distance in Graphs (Addison-Wesley, Redwood City, 1990).

[6] T. Bultan and C. Aykanat, Circuit partitioning using mean field annealing, Neurocomputing 8 (2) (1995) 171-194.

[7] J.A. Carretero and M.A. Nahon, Solving minimum distance problems with convex or concave bodies using combinatorial global optimization algorithms, IEEE Trans Systems, Man and Cybernetics – B 35 (6) (2005) 1144-1155.

[8] A. Ceroni, P. Frasconi, and G. Pollastri, Learning protein secondary structure from sequential and relational data, Neural Netwoks 18 (8) (2005) 1029-1039.

[9] Y. Che and Z. Tang, An efficient parallel algorithm for maximum cut problem, Neural Information Processing Letters and Reviews 11 (8) (2007) 175-180.

[10] C. Chekuri, A.V. Goldberg, D. Karger, M. Levine, and C. Stein, Experimental study of minimum cut algorithms, in: Proc. of 8th SODA, 324-333, 1997.

[11] S.-J. Chen and S.-M. Chen, Fuzzy risk analysis based on similarity measures of generalized fuzzy numbers, IEEE Trans Fuzzy Systems 11 (1) (2003) 45-56.

[12] A. Cripps and N. Nguyen, Fuzzy lattice reasoning (FLR) classification using similarity measures, in: V.G. Kaburlasos and G.X. Ritter, eds., Computational Intelligence Based on Lattice Theory (Springer, Heidelberg, Studies in Computational Intelligence 67, 2007) 263-284.

[13] P. Diamond and P. Kloeden, Metric Spaces of Fuzzy Sets (World Scientific, Singapore, 1994).

[14] D. Donato, L. Laura, S. Leonardo, and S. Millozzi, Simulating the webgraph: a comparative analysis of models, Neural Networks 6 (6) (2004) 84-89.

[15] M.-L. Fernández and G. Valiente, A graph distance metric combining maximum common subgraph and minimum common supergraph, Pattern Recognition Letters 22 (6-7) (2001) 753-758.

[16] G.W. Flake, R.E. Tarjan, and K. Tsioutsiouliklis, Graph clustering and minimum cut trees, Internet Mathematics 1 (4) (2004) 385-408.

[17] R.W. Floyd, Algorithm 96 ancestor, Communications of the ACM 5 (6) (1962) 344-345.

[18] P. Frasconi, M. Gori, and A. Sperduti, A general framework for adaptive processing of data structures, IEEE Trans Neural Networks 9 (5) (1998) 768-786.

[19] M. Gori and A. Petrosino, Encoding nondeterministic fuzzy tree automata into recursive neural networks, IEEE Trans Neural Networks 15 (6) (2004) 1435-1449.

[20] M. Gori and A. Sperduti, The loading problem for recursive neural networks, Neural Networks 18 (8) (2005) 1064-1079.

[21] M. Gori, A. Küchler, and A. Sperduti, On the implementation of frontier-to-root tree automata in recursive neural networks, IEEE Trans Neural Networks 10 (6) (1999) 1305-1314.

[22] M. Gori, M. Maggini, and L. Sarti, Exact and approximate graph matching using random walks, IEEE Trans Pattern Analysis and Machine Intelligence 27 (7) (2005) 1100-1111.

[23] M. Graña, Lattice computing: lattice theory based computational intelligence, in: Proceedings of the Kosen Workshop on Mathematics, Technology, and Education (MTE) 19-27, Ibaraki, Japan, 2008.

[24] G. Grätzer, General Lattice Theory, 2nd ed. (Birkhäuser, Basel, 2003).

[25] S. Günter and H. Bunke, Self-organizing map for clustering in the graph domain, Pattern Recognition Letters 23 (4) (2002) 405-417.

[26] S. Günter and H. Bunke, Validation indices for graph clustering, Pattern Recognition Letters 24 (8) (2003) 1107-1113.

[27] M. Hagenbuchner, A. Sperduti, and A.C. Tsoi, A self-organizing map for adaptive processing of structured data, IEEE Trans Neural Networks 14 (3) (2003) 491-505.

[28] P.R. Halmos, Measure Theory, 2nd ed. (Springer, Berlin, 1978).

[29] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert, Recursive self-organizing network models, Neural Networks 17 (8-9) (2004) 1061-1085.

[30] D. Hand, H. Mannila, and P. Smyth, Principles of Data Mining, 2001 Massachusetts Institute of Technology.

[31] B.J. Jain and F. Wysotzki, Solving inexact graph isomorphism problems using neural networks, Neurocomputing 63 (2005) 45-67.

[32] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, Neuro-Fuzzy and Soft Computing (Prentice-Hall, Upper Saddle River, NJ, 1997).

[33] D. Justice and A. Hero, A binary linear programming formulation of the graph edit distance, IEEE Trans Pattern Analysis Machine Intelligence 28 (8) (2006) 1200-1214.

[34] V.G. Kaburlasos, Improved fuzzy lattice neurocomputing (FLN) for semantic neural computing, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN) vol. 3 1850-1855, Portland, OR, 2003.

[35] V.G. Kaburlasos, FINs: lattice theoretic tools for improving prediction of sugar production from populations of measurements, IEEE Trans Systems, Man and Cybernetics – B 34 (2) (2004) 1017-1030.

[36] V.G. Kaburlasos, Towards a Unified Modeling and Knowledge-Representation Based on Lattice Theory – (Springer, Heidelberg, Studies in Computational Intelligence 27, 2006).

[37] V.G. Kaburlasos, Granular enhancement of fuzzy-ART/SOM neural classifiers based on lattice theory, in: V.G. Kaburlasos and G.X. Ritter, eds., Computational Intelligence Based on Lattice Theory (Springer, Heidelberg, Studies in Computational Intelligence 67, 2007) 3-23.

[38] V.G. Kaburlasos and S.E. Papadakis, Granular self-organizing map (grSOM) for structure identification, Neural Networks 19 (5) (2006) 623-643.

[39] V.G. Kaburlasos and S.E. Papadakis, A granular extension of the fuzzy-ARTMAP (FAM) neural classifier based on fuzzy lattice reasoning (FLR), Neurocomputing (*this volume*).

[40] V.G. Kaburlasos and V. Petridis, Fuzzy lattice neurocomputing (FLN) models, Neural Networks 13 (10) (2000) 1145-1170.

[41] V.G. Kaburlasos and G.X. Ritter (eds.) Computational Intelligence Based on Lattice Theory (Springer, Heidelberg, Studies in Computational Intelligence 67, 2007).

[42] V.G. Kaburlasos, I.N. Athanasiadis, and P.A. Mitkas, Fuzzy lattice reasoning (FLR) classifier and its application for ambient ozone estimation, International Journal of Approximate Reasoning 45 (1) (2007) 152-188.

[43] V.G. Kaburlasos, L. Moussiades, and A. Vakali, Granular graph clustering in the Web, in: Joint Conference on Information Sciences (JCIS), Proceedings of the 8th International Conference on Natural Computing (NC) 1639-1645, Salt Lake City, Utah, 2007.

[44] A.M. Ladd and L.E. Kavraki, Measure theoretic analysis of probabilistic path planning, IEEE Trans Robotics and Automation 20 (2) (2004) 229-242.

[45] L. Lee, Measures of distributional similarity, in: 37th Annual Meeting of the Association for Computational Linguistics (ACL) 25-32, East Stroudsburg, Maryland, 1999.

[46] G. Leng, T.M. McGinnity, and G. Prasad, An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network, Fuzzy Sets Systems 150 (2) (2005) 211-243.

[47] A. Micheli, F. Portera, and A. Sperduti, A preliminary empirical comparison of recursive neural networks and tree kernel methods on regression tasks for tree structured domains, Neurocomputing 64 (2005) 73-92.

[48] A. Micheli, D. Sona, and A. Sperduti, Contextual processing of structured data by recursive cascade correlation, IEEE Trans Neural Networks 15 (6) (2004) 1396-1410.

[49] M.E.J. Newman, Fast algorithm for detecting community structure in networks, Phys Rev E 69, 066133 (2004).

[50] W. Pedrycz, Knowledge-Based Clustering – From Data to Information Granules (John Wiley & Sons, Hoboken, 2005).

[51] M. Perkowitz and O. Etzioni, Towards adaptive web sites: conceptual framework and case study, Artificial Intelligence 118 (1-2) (2000) 245-275.

[52] V. Petridis and V.G. Kaburlasos, Clustering and classification in structured data domains using fuzzy lattice neurocomputing (FLN), IEEE Trans Knowledge and Data Engineering 13 (2) (2001) 245-260.

[53] S. Raha, N.R. Pal, and K.S. Ray, Similarity-based approximate reasoning: methodology and application, IEEE Trans Systems, Man and Cybernetics – A 32 (4) (2002) 541-547.

[54] L. Ralaivola, S.J. Swamidass, H. Saigo, P. Balbi, Graph kernel for chemical informatics, Neural Networks 18 (8) (2005) 1093-1110.

[55] C.J. Romanowski and R. Nagi, On comparing bills of materials: a similarity /distance measure for unordered trees, IEEE Trans Systems, Man and Cybernetics – A 35 (2) (2005) 249-260.

[56] G.C. Rota, The many lives of lattice theory, Notices of the American Mathematical Society 44(11) (1997) 1440-1445.

[57] S. Schmalz and B. Mertsching, Object recognition with structural descriptions and deformable models, Neurocomputing 31 (1-4) (2000) 143-151.

[58] G. Serpen and A. Parvin, On the performance of Hopfield network for graph search problem, Neurocomputing 14 (4) (1997) 365-381.

[59] L.B. Shams, M.J. Brady, and S. Schaal, Graph matching vs mutual information maximization for object detection, Neural Networks 14 (3) (2001) 345-354.

[60] K.A. Smith and A. Ng, Web page clustering using a self-organizing map of user navigation patterns, Decision Support Systems 35 (2) (2003) 245-256.

[61] B. Stein and O. Niggemann, On the nature of structure and its identification, in: P. Widmayer, G. Neyer, and S. Eidenbenz, eds., Graph-Theoretic Concepts in Computer Science (Springer, Heidelberg, LNCS 1665, 1999) 122-134.

[62] C. von der Malsburg, Pattern recognition by labeled graph matching, Neural Networks 1 (2) (1988) 141-148.

[63] Y. Wang, D. Wang, and W.H. Ip, Optimal design of link structure for e-supermarket website, IEEE Trans Systems, Man and Cybernetics – A 36 (2) (2006) 338-355.

[64] J.T.L. Wang, X. Wang, D. Shasha, and K. Zhang, Metricmap: an embedding technique for processing distance-based queries in metric spaces, Trans Systems, Man and Cybernetics – B 35 (5) (2005) 973-987.

[65] L. Xu and I. King, A pca approach for fast retrieval of structural patterns in attributed graphs, IEEE Trans Systems, Man and Cybernetics – B 31 (5) (2001) 812-817.

[66] X. Xu, Z. Tang, R. Wang, and X. Wang, A new motion equation for the minimum vertex cover problem, Neurocomputing 56 (2004) 441-446.

[67] Y. Zhao and G. Karypis, Criterion functions for document clustering: experiments and analysis, Technical Report TR#01-40, Dept. of Computer Science, Univ. of Minnesota, Minneapolis, MN (2001). Available at http://cs.umn.edu/~karypis/publications.

[68] Y. Zhao and G. Karypis, Empirical and theoretical comparisons of selected criterion functions for document clustering, Machine Learning 55 (2004) 311-331.

Figure Captions

Fig. 1. (a) Non-overlapping intervals $[a,b]$ and $[c,d]$ on the real line.
(b) Overlapping intervals $[a,b]$ and $[c,d]$ on the real line.

Fig. 2. (a) Non-overlapping circles $(\mathbf{c}_A,r_A)$ and $(\mathbf{c}_B,r_B)$ on the plane.
(b) Overlapping circles $(\mathbf{c}_A,r_A)$ and $(\mathbf{c}_B,r_B)$ on the plane.

Fig. 3. The *agglomerative similarity measure FLR* (*asmFLR*) neural computing algorithm for partitioning the nodes of a master-graph by clustering.

Fig. 4. A *dataset* including two master-graphs, i.e. M=2, each of whom has C=5 identical graph-clusters with 3, 4, or 5 vertices. A master-graph has both a user-defined *intra-connection ratio* r_{in}=1.0 and a user-defined *inter-connection ratio* r_{out}=0.3. (a) A master-graph with average r_{out} equal to 0.38 and standard deviation 0.23. (b) A master-graph with average r_{out} equal to 0.36 and standard deviation 0.24.

Fig. 5. A *dataset* including two master-graphs, i.e. M=2, each of whom has C=5 identical graph-clusters with 3, 4, or 5 vertices. A master-graph has both a user-defined *intra-connection ratio* r_{in}=1.0 and a user-defined *inter-connection ratio* r_{out}=0.6. (a) A master-graph with average r_{out} equal to 0.63 and standard deviation 0.10. (b) A master-graph with average r_{out} equal to 0.62 and standard deviation 0.13.

Fig. 6. Values of (graph clustering) indices *Purity* (curve in light gray color), *Entropy* (curve in dark gray color), and index $\mu_\sigma(.,.)$ (curve in black color), the latter is based on inclusion measure $\sigma_{V2}(P_i \leq Q_j) = |Q_j|/|P_i \vee Q_j|$, versus a (decreasing) number of graph-clusters for dataset **ds1** by
(a) neural algorithm *asmFLR*(d_a), and
(b) algorithm *Modularity*.

Fig. 7. Values of (graph clustering) indices *Purity* (curve in light gray color), *Entropy* (curve in dark gray color), and index $\mu_\sigma(.,.)$ (curve in black color), the latter is based on

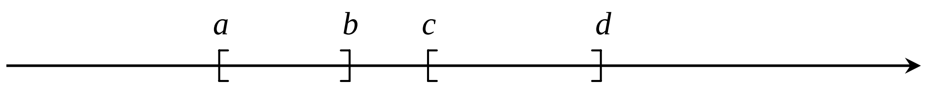
function $\sigma_{V3}(P_i \le Q_j) = |P_i \wedge Q_j| / |P_i \vee Q_j|$, versus a (decreasing) number of graph-clusters for dataset **ds1** by

(a) neural algorithm *asmFLR*(d_a), and

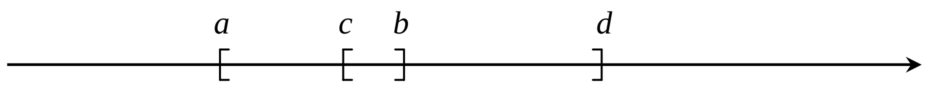
(b) algorithm *Modularity*.

Table Captions

Table 1	Statistics regarding five datasets **ds1**, **ds2**, **ds3**, **ds4**, and **ds5**
Table 2	Statistics of experimental results regarding dataset **ds1** by various graph clustering algorithms. The results are arranged in the order of decreasing *similarity measure* $\mu_\sigma(.,.)$ values
Table 3	Statistics of experimental results regarding dataset **ds2** by various graph clustering algorithms. The results are arranged in the order of decreasing *similarity measure* $\mu_\sigma(.,.)$ values
Table 4	Statistics of experimental results regarding dataset **ds3** by various graph clustering algorithms. The results are arranged in the order of decreasing *similarity measure* $\mu_\sigma(.,.)$ values
Table 5	Statistics of experimental results regarding dataset **ds4** by various graph clustering algorithms. The results are arranged in the order of decreasing *similarity measure* $\mu_\sigma(.,.)$ values
Table 6	Statistics of experimental results regarding dataset **ds5** by various graph clustering algorithms. The results are arranged in the order of decreasing similarity measure $\mu_\sigma(.,.)$ values



(a)



(b)

Fig. 1.

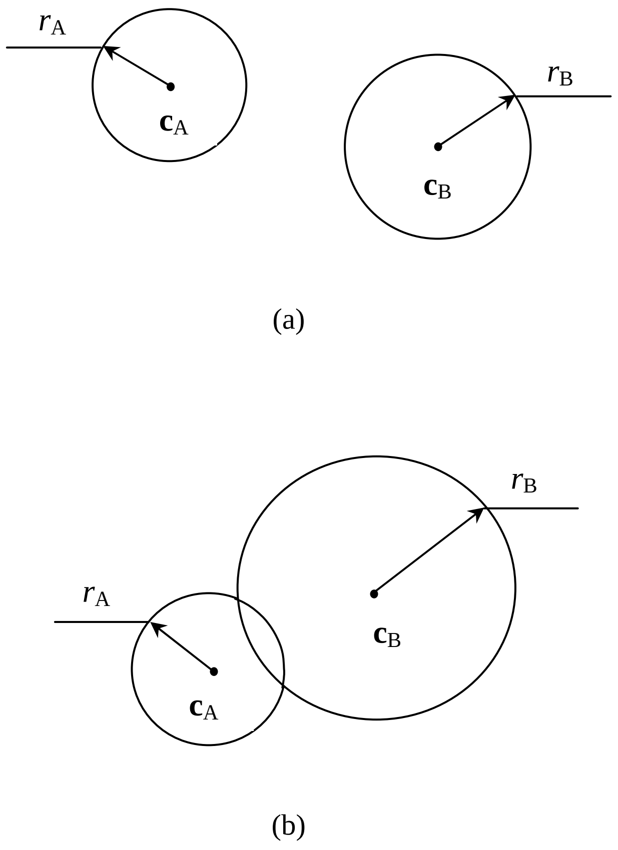
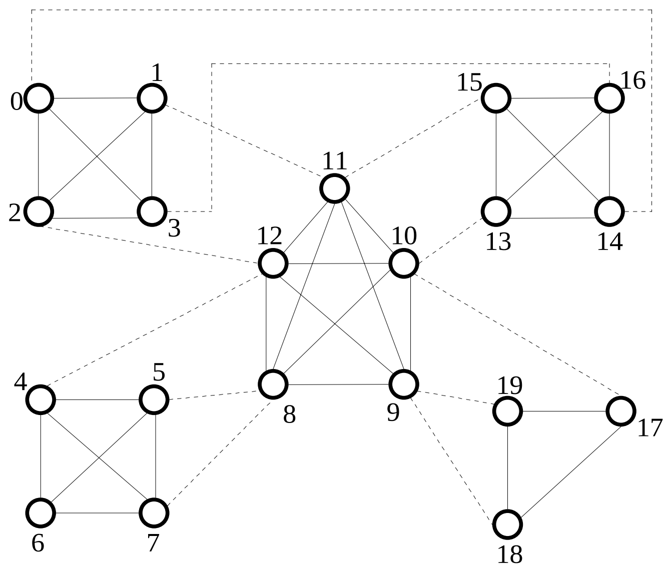


Fig. 2.

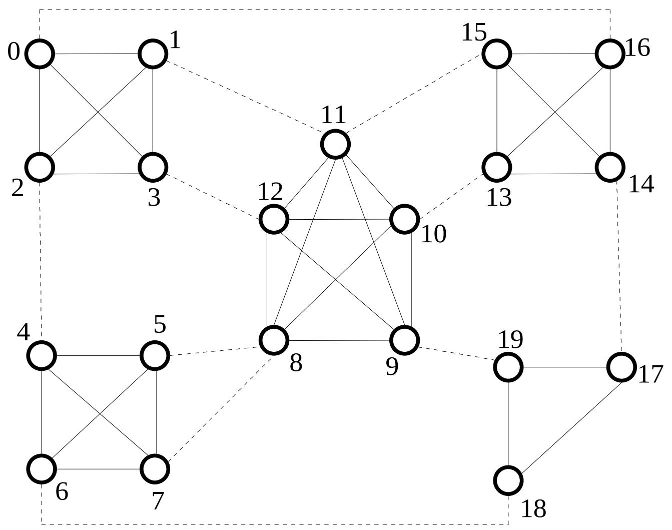
```
Step-0:  Consider a partition $\mathsf{P} = \{P_1,\ldots,P_n\}$ of the vertices $V = \{v_1,\ldots,v_N\}$ of a weighted
         master-graph $\mathsf{M} = (V, E)$, where clusters $P_i \subseteq V$, i=1,…,$n$ are such that both $\bigcup_{i=1}^{n} P_i$ =
         $V$ and $P_i \cap P_j = \{\}$ for $i \neq j$.

Step-1:  While (a user-defined *Assimilation Condition* is satisfied) do

Step-2:      Compute, in parallel, the similarity measure values $\mu_d(P_i,P_j) = 1/(1+d(P_i,P_j))$,
             $i,j \in \{1,\ldots,n\}$.

Step-3:      Competition among pairs of clusters.
             Winner in pair $(I,J) \in \mathsf{P}\times\mathsf{P}$ such that $(I,J) \doteq \arg\max_{i,j} \mu_d(P_i,P_j)$, where $I \neq J$.

Step-4:      Merge $P_I$ and $P_J$ by replacing them with the (single) cluster $P_I \cup P_J$; $n \leftarrow n-1$.
```

Fig. 3.



0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

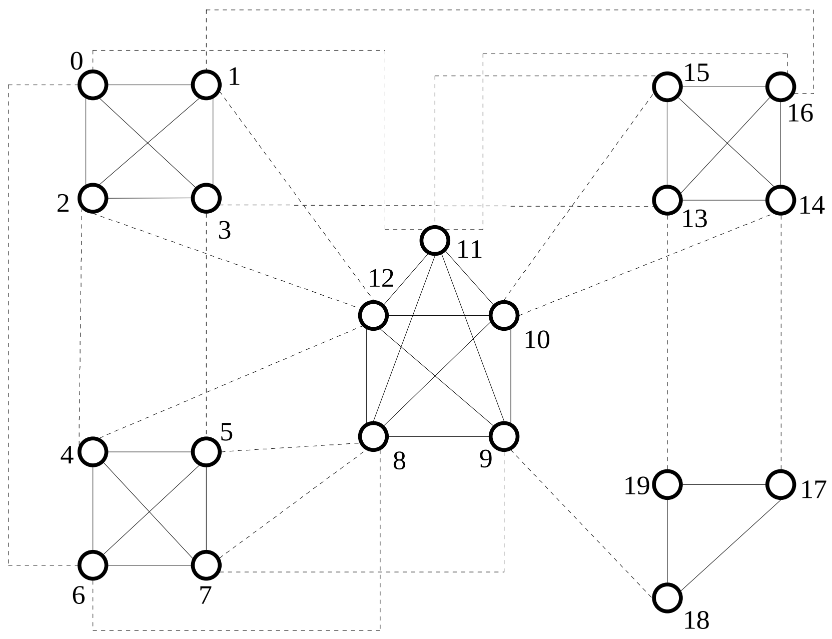
(a)



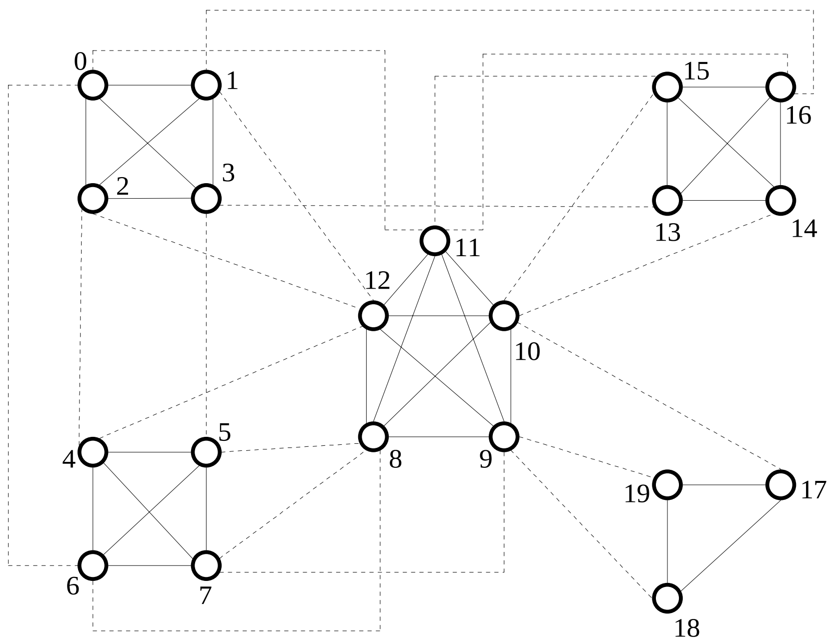
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

(b)

Fig. 4.

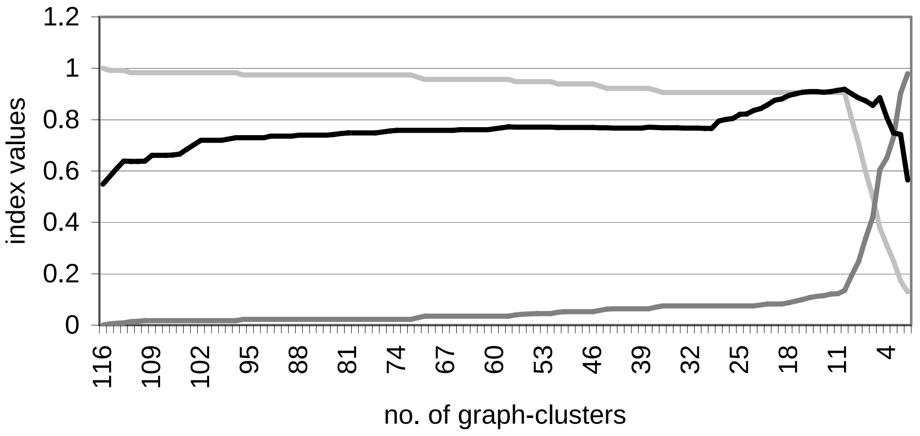

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

(a)

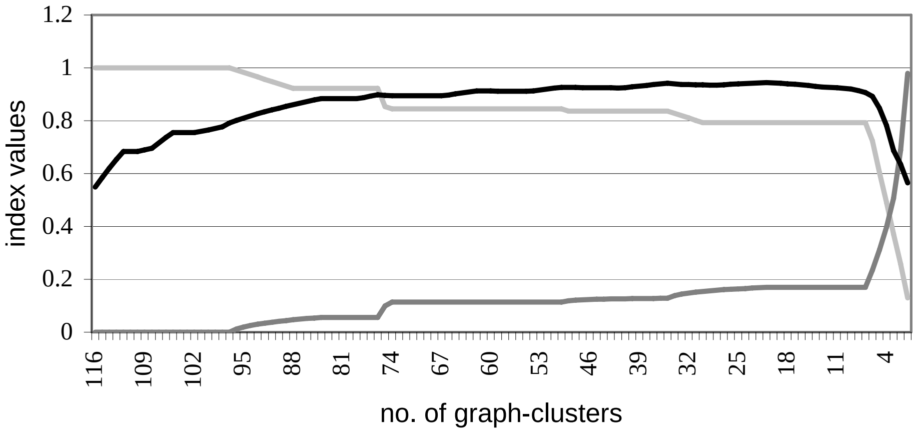

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

(b)

Fig. 5.

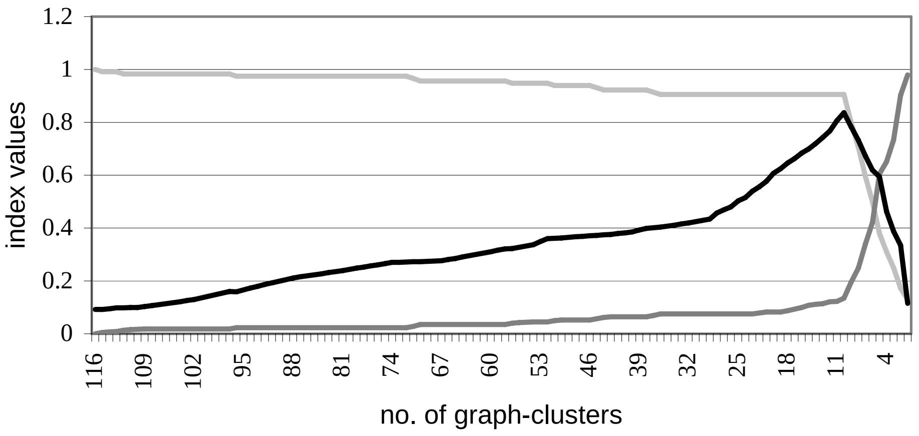

1.2
1
0.8
0.6
0.4
0.2
0
index values
116
109
102
95
88
81
74
67
60
53
46
39
32
25
18
11
4
no. of graph-clusters

(a)


1.2
1
0.8
0.6
0.4
0.2
0
index values
116
109
102
95
88
81
74
67
60
53
46
39
32
25
18
11
4
no. of graph-clusters

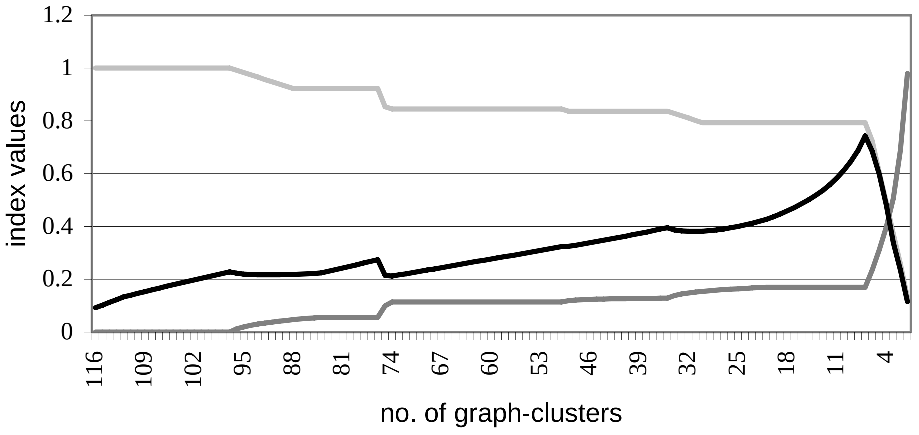
(b)

Fig. 6.



1.2
1
0.8
0.6
0.4
0.2
0
index values
116
109
102
95
88
81
74
67
60
53
46
39
32
25
18
11
4
no. of graph-clusters

(a)



1.2
1
0.8
0.6
0.4
0.2
0
index values
116
109
102
95
88
81
74
67
60
53
46
39
32
25
18
11
4
no. of graph-clusters

(b)

Fig. 7.

Table 1

Statistics regarding five datasets **ds1**, **ds2**, **ds3**, **ds4**, and **ds5**

dataset	#clusters	cluster size				r_{in}	user-defined	r_{out}	
	C	$v_{\min}$	$v_{\max}$	average	std.		r_{out}	average	std.
ds1	10	5	15	11.6	3.565	1	0.5	0.532	0.031
ds2	10	5	15	11.6	3.565	1	0.7	0.755	0.051
ds3	10	5	15	11.6	3.565	1	0.9	0.937	0.043
ds4	10	5	15	11.6	3.565	1	1.0	0.996	0.031
ds5	10	5	15	11.6	3.565	1	1.2	1.255	0.046

Table 2

Statistics of experimental results regarding dataset **ds1** by various graph clustering algorithms. The results are arranged in the order of decreasing *similarity measure* $\mu_\sigma(.,.)$ values

Algorithm	#clusters	cluster size				#trivials	Purity	Entropy	similarity
name	C	min	max	average	std.				measure $\mu_\sigma(.,.)$
asmFLR **(*d_a*)**	10	4.8	16	11.6	3.818	0	0.957	0.062	0.921
asmFLR **(*d_M*)**	10	3.7	33.7	11.6	9.049	0	0.731	0.252	0.665
Modularity	10	1	22.5	11.6	7.529	2.5	0.816	0.156	0.659
MajorClust	3	9	92	38.6	46.285	0	0.336	0.698	0.510
asmFLR **(*d_H*)**	10	1	90.2	11.6	27.704	3.2	0.309	0.729	0.269
MinCutTrees	1	116	116	116	NaN	0	0.129	0.979	0.114
SingleLink	10	1	107	11.6	33.520	9	0.206	0.899	0.111

Table 3

Statistics of experimental results regarding dataset **ds2** by various graph clustering algorithms. The results are arranged in the order of decreasing *similarity measure* $\mu_\sigma(.,.)$ values

Algorithm	#clusters	cluster size				#trivials	Purity	Entropy	similarity
name	*C*	min	max	average	std.				measure $\mu_\sigma(.,.)$
asmFLR **(*d_a*)**	10	5	17.9	11.6	3.818	0	0.888	0.148	0.817
Modularity	10	1	23.7	11.6	7.529	2.1	0.824	0.147	0.705
asmFLR **(*d_M*)**	10	2.1	38.4	11.6	9.049	0.3	0.606	0.375	0.512
MajorClust	2	32	84	58	46.285	0	0.258	0.723	0.261
asmFLR **(*d_H*)**	10	1	94.3	11.6	27.704	3.8	0.282	0.780	0.220
MinCutTrees	1	116	116	116	NaN	0	0.129	0.979	0.114
SingleLink	10	1	107	11.6	33.520	9	0.206	0.899	0.111

Table 4

Statistics of experimental results regarding dataset **ds3** by various graph clustering algorithms. The results are arranged in the order of decreasing *similarity measure* $\mu_\sigma(.,.)$ values

Algorithm	#clusters	cluster size				#trivials	Purity	Entropy	similarity
name	C	min	max	average	std.				measure $\mu_\sigma(.,.)$
asmFLR **(*d_a*)**	10	5.6	18.4	11.6	4.109	0	0.869	0.176	0.788
Modularity	10	1	24.4	11.6	8.269	2.7	0.787	0.190	0.623
asmFLR **(*d_M*)**	10	1.5	49.2	11.6	14.720	0.6	0.517	0.470	0.429
asmFLR **(*d_H*)**	10	1	98.5	11.6	30.562	4.4	0.256	0.819	0.185
MajorClust	1	116	116	116	NaN	0	0.129	0.979	0.114
MinCutTrees	1	116	116	116	NaN	0	0.129	0.979	0.114
SingleLink	10	1	107	11.6	33.520	9	0.206	0.899	0.111

Table 5

Statistics of experimental results regarding dataset **ds4** by various graph clustering algorithms. The results are arranged in the order of decreasing *similarity measure* $\mu_\sigma(.,.)$ values

Algorithm	#clusters	cluster size				#trivials	Purity	Entropy	similarity
name	*C*	min	max	average	std.				measure $\mu_\sigma(.,.)$
asmFLR **(*d_a*)**	10	4.2	21	11.6	5.296	0.3	0.804	0.246	0.697
Modularity	10	1	22.4	11.6	7.281	2.4	0.799	0.185	0.629
asmFLR **(*d_M*)**	10	1.2	54.3	11.6	16.323	1.1	0.466	0.532	0.370
asmFLR **(*d_H*)**	10	1	96.4	11.6	29.841	4	0.274	0.794	0.207
MajorClust	1	116	116	116	NaN	0	0.129	0.979	0.114
MinCutTrees	1	116	116	116	NaN	0	0.129	0.979	0.114
SingleLink	10	1	107	11.6	33.520	9	0.206	0.899	0.111

Table 6

Statistics of experimental results regarding dataset **ds5** by various graph clustering algorithms. The results are arranged in the order of decreasing similarity measure $\mu_\sigma(.,.)$ values

Algorithm	#clusters	cluster size				#trivials	Purity	Entropy	similarity
name	*C*	min	max	average	std.				measure $\mu_\sigma(.,.)$
asmFLR **(*d_a*)**	10	3.4	21.6	11.6	5.566	0.3	0.800	0.258	0.687
Modularity	10	1	23.5	11.6	7.175	2.1	0.806	0.197	0.649
asmFLR **(*d_M*)**	10	1.2	62.8	11.6	19.133	1.1	0.402	0.602	0.317
asmFLR **(*d_H*)**	10	1	94.1	11.6	29.032	3.7	0.287	0.786	0.214
MajorClust	1	116	116	116	NaN	0	0.129	0.979	0.114
MinCutTrees	1	116	116	116	NaN	0	0.129	0.979	0.114
SingleLink	10	1	107	11.6	33.520	9	0.206	0.899	0.111