Oppositional Fruitfuly and IKFCM Algorithm Based Graph Pattern Matching

Sreedevi O.B
Register Number-12473,
Scott Christian College, Nagercoil-629 003, India.
S. Joseph Robin
Associate Professor, Department of Mathematics,
Scott Christian College, Nagercoil-629 003, India
Affiliated to Manonmaniam Sundaranar University, Abishekapatti, Tirunelveli - 627 012,
Tamil Nadu, India

ABSTRACT

Graph pattern matching has been widely used in e.g., social data analysis. A number of matching algorithms have been developed that, given a graph pattern. Social and economic data analysis, however, introduces new challenges to graph pattern matching. Social graphs are typically large with millions of nodes and billions of edges. This gives rise to the following problems with the matching algorithms. These are the major drawbacks of different existing works, which motivate us to do this research on graph pattern matching. In our proposed research in the initially generated the dependency graphs to place the Entropy, Mutual Information, Conditional Entropy and Relative Entropy from the input strategies. In the graph, graph kernels and graph matching are the two major techniques for evaluating the similarities among the graphs. In these techniques, Graph kernel-based technique is applied to evaluate the similarity between two graphs and compares all pairs of substructures of the graphs. So in this research, based on Improved Kernel Based Fuzzy C Means (IKFCM), a two-step graph pattern matching technique is suggested. Individually these two values are calculated for each diagram. The second phase is the Dependency Graph Matching; where by the IKFCM, the two dependency graph is matching. In our method kernel selection in IKFCM is done by Oppositional Fruitfly algorithm (OFA).

1. INTRODUCTION

Graphs are an effective tool for modelling complex structured data. However, few researches model the ensemble of local features by graphs in human action recognition. There are two nontrivial difficulties to be solved: i) how to construct graphs to model these local features; ii) how to measure similarity between the constructed graphs [5]. Graph pattern matching is long investigated in database study. Graphs are used to provide meaningful representations of objects and patterns, as well as more abstract descriptions. These graphs can be used on a variety of applications including network analysis, face recognition, and image segmentation. However at the heart of graph theory is the problem of graph matching, within exact graph matching, there are three graph isomorphism problems. This is an important issue, because graph matching techniques have found ample applications in various scientific disciplines [9]. Graphs have been widely used to model complex data in many real-world applications. In life sciences, graph pattern matching can be used for protein interaction networks comparison and protein structure matching [1].

Graph matching is widely used in pattern recognition, such as computer vision, scene analysis, chemistry, and biology, where the relationships and interactions between objects are modelled as graphs. To identify the similarities between two different graphs (i.e., patterns), various graph-matching methods have been developed to identify vertex correspondence between graphs [10]. In computer vision, it is widely known that the fundamental problem of establishing correspondences between two sets of visual features can be effectively solved by graph matching. [11].

2. PROBLEM DEFINITION

- It require enumerating sub-units exhaustively with high space and time overhead, and some of them do not capture the attributes on vertices or edges which are continuous values on graphs and often suffer from poor pruning power [2].
- Supporting of several new specifications, such as matching probabilistic multi-labelled graphs and graph based processing for large-scale distributed pattern matching is the key issue in graph match [12].
- Bipartite graph matching scheme does not require the one-to-one matching for vertices in the bipartite graph and also has relatively higher computational complexity than some other methods [13].

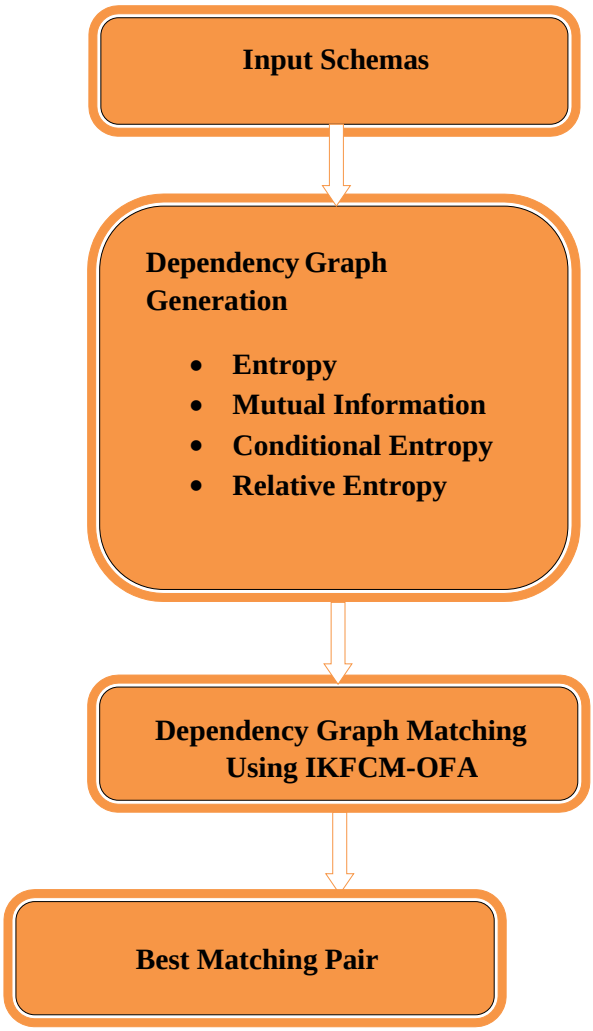

Fig 1: Proposed IKFCM-O Based Graph Pattern Matching

3. PROPOSED METHODOLOGY

The schema matching of the data integrity a proficient technique is proposed also this approach increases the accuracy of matching as it uses the dependency graph. This algorithm is used to find the matching between the dependency graph also we have introduced more attributes such as Entropy, Mutual Information, Conditional Entropy and Relative Entropy. In our proposed research in the initially generated the dependency graphs to place the Entropy, Mutual Information, Conditional Entropy and Relative Entropy from the input strategies. And the values of these attributes are individually computed for each schema. The second phase is the Dependency Graph Matching; where Improved Kernel Fuzzy C-Means Clustering (IKFCM) is accomplished for matching the two dependency graph. This suggested technique is suggested on kernel selection in IKFCM is attained by applying the Oppositional Fruitfly Algorithm (OFA). In addition, matching process is incorporated by computed the distance measures with the help of Euclidean Distance among the two dependencies graphs of schemas.

4.1 DEPENDENCY GRAPH GENERATION

This section gives brief description about dependency graph generation. Here, the instances of the two tables are utilized then they are transformed into graph format. After this, entropy and mutual information are calculated for producing the dependency graph. Moreover, for constructing dependency graph the entropy and mutual information values are derived separately for each table instances.

L	M	N	O	U	V	W	X
L1	M1	N1	O1	**U1**	V1	W1	X1
L2	M2	N2	O2	**U2**	V2	W2	X2
L3	M3	N3	O3	**U3**	V3	W3	X3
L4	M4	N4	O4	**U4**	V4	W4	X4

Fig. 2 example of input table instances of the proposed work

4.1.1 Entropy

The characteristic entropy decides the ambiguity of values with a non-negative real number. The value of entropy is creates separately for each Characteristic in the two table instances. Moreover, the values of entropy established on the probabilities rather than attribute based actual values. Here, we utilized entropy value for getting the mutual information but this technique established on conditional entropy [23] among the characteristics as calculated co-relation. Two characteristics are established by *R and S*, the conditional entropy can be defined as,

The attribute's entropy define the

$$C_y(R) = -\sum_{r \in R} p(r) \log\ p(r) \tag{1}$$

$$C_y(R \mid S) = -\sum_{m \in M} p(r) \sum_{n \in N} p(r \mid s) * \log_2(p(r \mid s)) \tag{2}$$

Here, *C()* mention the conditional entropy among the attributes *R* and *S*

p(r) denotes the probability of attribute *R*.

p(r|s) denotes the conditional probability among the attribute *R* and *S*.

4.1.2 Mutual Information

The entropy and mutual information are computed from the characteristics of two table instances. Therefore, entropy values are primarily calculated. Once determining entropy values of each characteristic, the mutual information is determined from two attributes. Because of the learning of the other characteristics mutual information evaluates the decreasing in susceptibility of one characteristic. In addition, by applying non-negative real number, mutual information demonstrates the relationship among both table instances. The mutual information is defined in the equation (2) which comes below,

For determining the mutual information from the two attributes once

$$RI(R,S) = C_y(R) - C_y(R \mid S)$$
$$(or) \qquad (3)$$
$$RI(R,S) = C_y(S) - C_y(S \mid R)$$

Where, *RI* is conceived as the mutual information value

R and *S* are the attributes with alphabet

4.1.3 Conditional Entropy

Let Y be a discrete random variable with outcomes, $\{y_1,..., y_m\}$, which occur with probabilities, $p_y(y_j)$. The avg. information you gain when told the outcome of *Y* is:

$$H_y = -\sum_{j=1}^{m} p_y(y_j) \log p_y(y_j) \qquad (4)$$

4.1.4 Relative Entropy

Let a discrete distribution have probability function *p_k*, and let a second discrete distribution have probability function *q_k*. Then the relative entropy of *p* with respect to *q*, also called the Kullback-Leibler distance, is defined by

$$d = \sum_k p_k \log_2 \left(\frac{p_k}{q_k} \right) \qquad (5)$$

Although *d(p,q)!=d(q,p),* so relative entropy is therefore not a true metric, it satisfies many important mathematical properties. For example, it is a convex function of *p_k*, is always nonnegative, and equals zero only if *p_k=q_k*.

Kernel Based Fuzzy C-Means Clustering (KFCM)

Diverse kernel related fuzzy c-means algorithmic is expanding the kernel fuzzy c-means algorithm through a dissimilar kernel learning surroundings. The projected process applies numerous kernel fuzzy c means for collecting the implemented situation. The intention task of projected numerous kernel fuzzy c-means algorithm is efficiently clarified as afforded below.

$$F(M,A) = \sum_{i=1}^{N} \sum_{j=1}^{a} M_{ij}^{m} (1 - K_{MK}(t_j, A_i)) \qquad (6)$$

Where,

M_{ij} is the membership of j^{th} data in the i^{th} cluster A_i
A is the cluster centre
K_{MK} is the multiple kernel function

Procedure for KFCM

Step 1: Initialize the number of test cases *(t)*, number of c
Cluster *(A)* and number of kernels *(K).*
Step 2: Initialize the membership matrix *M*.
Step 3: Compute the cluster centre by the following equation

$$A_j = \frac{\sum_{i=1}^{N} M_{ij}^{m} t_i}{\sum_{i=1}^{N} M_{ij}} \tag{7}$$

Step 4: Update the membership function by the following equation

$$M_{ij} = \frac{1}{\sum_{K=1}^{A} \left(\frac{\|t_i - A_j\|}{\|t_i - A_K\|} \right)^{\frac{2}{m-1}}} \tag{8}$$

m is any real number greater than 'one'. In multiple kernel fuzzy c means, t_i refers the kernel function k_{MK} *(x,y).* Here we are conceiving multiple kernels for our suggested work. So $k_{MK}(x,y) = k1(x,y)+k2(x,y)$ is a kernel.

$$K_{MK}(x, y) = K_1(x, y) + K_2(x, y) \tag{9}$$

$$K_1(x, y) = x^T y + c \tag{10}$$

$$K2(x, y) = 1 - \frac{\| x - y \|^2}{\| x - y \|^2 + c} \tag{11}$$

Where, c is the constant value. From the above equation the cluster centre equation (10) and membership equation (11) is changed. Now the cluster centre calculation is done by equation (12),

$$A_j = \frac{\sum_{i=1}^{N} M_{ij}^{m} k_{MK}(x, y)}{\sum_{i=1}^{N} M_{ij}} \tag{12}$$

Membership updating is done by equation (13),

$$M_{ij} = \frac{1}{\sum_{k=1}^{A} \left(\frac{\|K_{MK}(x, y) - A_j\|}{\|K_{MK}(x, y) - A_k\|} \right)^{\frac{2}{m-1}}} \tag{13}$$

Step 5: if $\left\| M^{(K+1)} - M^{(K)} \right\| < \in$ then stop, otherwise go to equation.

According to this kernel related FCM, we will predefine a fault as constantly.

In our method kernel selection IKFCM is done by oppositional fruitfly algorithm. The matching process works derived from the distance measures, Euclidean distance between the two dependencies graph of schemas.

Opposition Fruit Fly Algorithm (OFA)

This proposed technique is established on kernel selection in IKFCM is reached by applying the Oppositional Fruitfly algorithm (OFA). In addition, matching process is incorporated by computing the distance measures with the help of Euclidean Distance between the two dependencies graphs of schemas. Fruit fly algorithm is an algorithm that simulates the foraging behaviour of fruit flies. The fruit fly algorithm is a new technique for seeking global optimization. It began from the examination on food hunting behaviours of fruit fly swarm. Fruit fly is a superb food hunter with sharp osphresis and vision. At to begin with, it identifies food source by noticing a wide range of fragrances floating all around and flies toward the corresponding place. After reaching close towards the food, it might discover food or go t o that particular place with its delicate vision. Food sources are referred by the optima and the methodology of foraging is reproduced by means of the iteratively seeking for the optima in the FOA. The improved form of fruit fly algorithm is said to be OFA. It provides developed performance than the fruitfly algorithm.

***Data*:** Initial low variance blocks position

***Result*:** Best position of blocks

***Step 1*:** Parameters initialization: the major parameters of the FOA are the total evolution number and low variance blocks position. In our suggested technique fruit fly refer the low variance block position. Initialize random location of low variance blocks position (*PX*_axis, *PY*_axis).

Step 2: To change the traditional fruit fly algorithm, oppositional method is introduced. According to opposition based learning (OBL) inaugurated by *Tizhoosh in 2005* [23], the current agent and its opposite agent are considered simultaneously to become a better approximation for current agent solution. It is afforded that an opposite agent solution has a better chance to be closer to the global optimal solution than random agent solution. The opposite variance blocks positions (OP_m) are completely determined by components of P_m .

$$OP_m = \left[op_m^1, op_m^2, ...op_m^d\right] \tag{14}$$

Where $OP_m = Low_m + Up_m - P_m$ with $OP_m \in \left[Low_m, Up_m\right]$ is the position of m^{th} low variance blocks OP_m in the d^{th} dimension of oppositional blocks.

Step 3: Exploration applying arbitrary path and low variance block selection. Here, P_m is the m^{th} location of low variance blocks.

$$P_m(x, y) = \left(PX_m, PY_m\right)^T$$

$$\text{PX}_m = PX\_axis + RandomValue$$

$$\text{PY}_m = PY\_axis + RandomValue \tag{15}$$

Step 4: Position Evaluation of suggested technique,

$$\text{BP}_\text{m} = PSNR \tag{16}$$

Step 5: Substitute position of low variance blocks into fitness function

$$best block = function(Max\, BP_m) \tag{17}$$

Step 6: Detect the most excellent positions of low variance blocks.

$$[\text{Excellent block Excellent selection}] = \max(\text{PSNR}) \tag{18}$$

Step 7: Retains the best position of low variance block value and *x*, *y* coordinate, the fruit fly swarm will utilize visualization to flutter in that direction.

$$selected block = \max PSNR$$

$$\text{PX\_}axis = PX(Excellent index)$$

$$\text{PY\_}axis = PY(Excellent index) \tag{19}$$

Step 8: Enter successive optimization to replicate the execution of stages 3-6, then decide if the position of low variance block is better than the past position of low variance blocks, if yes, execute task 7.

Hence applying the OFA algorithm the positions for embedding the watermark bits are optimally chosen. After finding the position for embedding the watermark bits it is required to embed the bits. The process admitted in watermark embedding is as follows,

5. RESULT AND DISCUSSION

Our suggested schema matching work is implemented in the platform of JAVA. To estimate the performance of our schema matching work two real time census data schemas are applied.

5.1 Experimental Results of our proposed work

Initially two input schemas are afforded to determine the schema elements. Using these two input schemas, the matching process is proceeding. The attributes are introduced in schema-1 and schema-2 are State, District, Level, Name, TRU, No_HH, TOT_HL_P, TOT_HL_M , TOT_HL_F , P_06 , M_06 , F_06 , P_SC , M_SC , F_SC , P_ST , M_ST , F_ST , P_LIT , M_LIT , F_LIT , P_ILL , M_ILL and F_ILL . At first find the entropy value for each attributes for both schema. The table.1 demonstrates the output value for both schemas.

Table.1 Entropy value for both schemas

Entropy of Graph 1	Entropy of Graph 2
State --> 306.1997424985836 District --> 95.57926911412548 Level --> 306.1997424985836 Name --> 85.69175851611251 TRU --> 254.1315299656756 No_HH --> 31.205174158575893 TOT_HL_P --> 14.386191754963455 TOT_HL_M --> 12.136851176488221 TOT_HL_F --> 13.862943611198906 P_06 --> 34.230434420031386 M_06 --> 51.853394644704636 F_06 --> 54.79928036437072 P_SC --> 184.18637251337128 M_SC --> 194.13933093305695 F_SC --> 211.61346215789973 P_ST --> 75.89051407135351 M_ST --> 79.412780790938 F_ST --> 86.63892911387471 P_LIT --> 23.227206065447348 M_LIT --> 27.046291075216224 F_LIT --> 45.66058738258129 P_ILL --> 14.386191754963455 M_ILL --> 23.549099073705005 F_ILL --> 30.411349410262506	State --> 291.50225376689224 District --> 85.69175851611251 Level --> 332.43157072776546 Name --> 85.69175851611251 TRU --> 258.408615359234 No_HH --> 12.999897393843565 TOT_HL_P --> 4.1588830833596715 TOT_HL_M --> 7.4547199493640015 TOT_HL_F --> 6.931471805599453 P_06 --> 19.068322982087675 M_06 --> 25.136748570331786 F_06 --> 34.07598838659475 P_SC --> 19.166548487866756 M_SC --> 26.86284100504247 F_SC --> 24.371927858755523 P_ST --> 258.6605711271696 M_ST --> 266.0125436868908 F_ST --> 277.25887222397813 P_LIT --> 9.704060527839234 M_LIT --> 15.772486116083346 F_LIT --> 36.2271034592909 P_ILL --> 6.931471805599453 M_ILL --> 12.476649250079015 F_ILL --> 15.772486116083346

Mutual information of Schema 1	Mutual Information of Schema 2
State-->District == 199.47939017504262 State-->Level == 4001.6268320526337 State-->Name == 181.28571428571453 District-->Level == 184.31168831168853 District-->Name == 6.077922077922078 District-->TRU == 94.76923076923065 Level-->Name == 181.28571428571453 Level-->TRU == 1581.4615384615404 Name-->P_ILL == 2.164502164502167 Name-->M_ILL == 3.105627705627707 Name-->F_ILL == 3.3939393939393945 TRU-->M_ILL == 41.19100899100893 TRU-->F_ILL == 43.68781218781214 No_HH-->M_ILL == 1.605735930735931 No_HH-->F_ILL == 1.949621212121212 TOT_HL_P-->F_LIT == 1.7232683982683983 TOT_HL_P-->P_ILL == 0.5681818181818182 TOT_HL_P-->M_ILL == 1.0248917748917747 P_06-->M_ILL == 3.2714574314574323 P_06-->F_ILL == 2.813434343434343 M_06-->F_06 == 11.29420831693559 M_06-->P_ILL == 3.9563164108618656 M_06-->M_ILL == 6.79392955529319 M_06-->F_ILL == 5.669293585202676 F_06-->P_SC == 81.37037310332771 F_06-->M_ILL == 4.82135642135642 F_06-->F_ILL == 4.676527176527175 P_SC-->M_ILL == 50.566932310114154 P_SC-->F_ILL == 50.420123311032405 M_SC-->P_ILL == 39.48076651609262 M_SC-->M_ILL == 53.85074816487864 M_SC-->F_ILL == 54.13747412008283 F_SC-->P_ILL == 49.18843901196841 F_SC-->M_ILL == 65.7405143875732 F_SC-->F_ILL == 67.23562303709362 P_ST-->M_ILL == 9.922918258212373 M	State-->District == 116.14046822742458 State-->Level == 2918.580344569616 State-->Name == 116.14046822742458 State-->TRU == 1055.1863860330045 State-->No_HH == 47.9292971521233 District-->No_HH == 1.9829059829059823 District-->TOT_HL_P == 1.4743589743589722 District-->TOT_HL_M == 1.6965811965811948 District-->TOT_HL_F == 1.5897435897435879 Level-->Name == 214.60222672064754 Level-->TRU == 1904.5566418740607 Level-->No_HH == 91.14439946018881 Name-->TRU == 77.3336498892054 Name-->No_HH == 1.9829059829059823 Name-->TOT_HL_P == 1.4743589743589722 Name-->TOT_HL_M == 1.6965811965811948 Name-->TOT_HL_F == 1.5897435897435879 TRU-->No_HH == 31.27067657730046 TRU-->TOT_HL_P == 26.666684929505394 TRU-->TOT_HL_M == 28.641056079090212 TRU-->TOT_HL_F == 28.024691358024647 No_HH-->TOT_HL_P == 0.20085470085470084 No_HH-->TOT_HL_M == 0.2521367521367521 No_HH-->TOT_HL_F == 0.22649572649572647 TOT_HL_P-->P_06 == 0.7179487179487181 TOT_HL_P-->M_06 == 0.7948717948717949 P_06-->F_SC == 2.7472527472527473 P_06-->P_ST == 58.886609686609724 P_06-->M_ST == 60.750376558778115 M_06-->F_06 == 3.1267043142043143 M_06-->P_SC == 2.6906288156288154 M_06-->F_ILL == 1.0972222222222225 F_06-->P_SC == 3.6527777777777777 F_06-->M_SC == 4.655982905982907 F_06-->F_SC == 3.806980056980057

Conclusion

In this paper improved k means clustering algorithm based schema matching was suggested. An applying JAVA platform our suggested method was accomplished. To prove the suggested method attains better results. Functional dependencies among attributes in the tables to be matched are extracted applying information theoretic measures and directed dependency graph is constructed. In the next stage, matching node pairs across the dependency graphs are inquired by running a graph-matching algorithm. It is shown that, although entropy based schema matching is effective, further developments is possible by exploiting inter attribute correlations like mutual information or functional dependency. The four algorithms for the schema matching problem are distinguished and it is proved experimentally that the algorithms applying relationships existing between attributes produce better results equated to the ones using individual attribute's value distribution by this work. The suggested approach applies fine grained functional dependency relationships and produces accurate results

REFERENCE:

[1] Cheng, Jiefeng, Jeffrey Xu Yu, and S. Yu Philip. "Graph pattern matching: A join/semijoin approach." IEEE Transactions on Knowledge and Data Engineering 23, no. 7, pp.1006-1021, 2011.

[2] Du, Ruihuan, Jiannan Yang, Yongzhi Cao, and Hanpin Wang "Personalized graph pattern matching via limited simulation" Knowledge-Based Systems, vol.141, pp.31-43, 2018.

[3] Singh, Komal, and Vikram Singh "Graph pattern matching: A brief survey of challenges and research directions" In Computing for Sustainable Global Development (INDIA Com), 2016 3rd International Conference on pp. 199-204. IEEE, 2016.

[4] Lozano, Miguel Angel, and Francisco Escolano "Graph matching and clustering using kernel attributes" Neurocomputing 113, pp.177-194, 2013.

[5] Zhu, Linhong, Wee Keong Ng, and James Cheng. "Structure and attribute index for approximate graph matching in large graphs" Information Systems, vol.36, no. 6, pp.958-972, 2011.

[6] De La Higuera, Colin, Jean-Christophe Janodet, Émilie Samuel, Guillaume Damiand, and Christine Solnon "Polynomial algorithms for open plane graph and sub graph isomorphisms" Theoretical Computer Science,vol.498, pp.76-99, 2013.

[7] Liu, Zhi-Yong, Hong Qiao, and Lei Xu "An extended path following algorithm for graph-matching problem" IEEE transactions on pattern analysis and machine intelligence, vol. 34, no. 7, pp.1451-1456, 2012.

[8] Wang, Hongzhi, Ning Li, Jianzhong Li, and Hong Gao "Parallel algorithms for flexible pattern matching on big graphs" Information Sciences, vol.436, pp. 418-440, 2018.

[9] Yuan, Ye, Guoren Wang, Lei Chen, and Bo Ning "Efficient pattern matching on big uncertain graphs" Information Sciences, vol.339, pp.369-394, 2016.

[10] Yao Lu, Kaizhu Huang, Cheng-Lin LiuDoubly Stochastic Projected Fixed-Point Algorithm for Large Graph Matching, pp.1-17, 2016.

[11] Tang, Jin, Bo Jiang, Aihua Zheng, and Bin Luo "Graph matching based on spectral embedding with missing value" Pattern Recognition 45, no. 10, pp.3768-3779, 2012.

[12] Liu, Zhi-Yong, Hong Qiao, Li-Hao Jia, and Lei Xu "A graph matching algorithm based on concavely regularized convex relaxation", Neuro computing, vol.134, pp.140-148, 2014.

[13] Krleža, Dalibor, and Krešimir Fertalj "Graph matching using hierarchical fuzzy graph neural networks." IEEE Transactions on Fuzzy Systems, vol. 25, no. 4, pp.892-904, 2017.