Analysis of Current Flows in Electrical Networks for Error-Tolerant Graph Matching

Alejandro Gutiérrez-Muñoz and Lawrence O. Hall
{agutierrez,hall}@cse.usf.edu

Department of Computer Science and Engineering, University of South Florida
4202 E. Fowler Ave. Tampa, FL 33620-9951, USA

***Abstract*—In this paper, we present a novel approach to error-tolerant graph matching based on current flow analysis. Current flow analysis in electrical networks shares interesting connections to the number of random walks along the graph. We propose an algorithm to calculate a similarity measure between two graphs based on the current flows along geodesics of the same degree. This similarity measure can be applied over large graph datasets, making it feasible to compare datasets of significant size in a reasonable amount of time. We describe our operational prototype and evaluate its effectiveness on the NCI-HIV dataset.**

***Index Terms*—graph matching, graph kernels, current flows, electrical networks.**

I. Introduction

SEVERAL error-tolerant graph matching techniques have been developed over the last two decades. Some of these techniques rely upon similarity measures based on the topology of the graphs [1], [2], [3], [4], [5], [6]. Random walks and edit distance kernels are examples of such methods, that in conjunction with learning algorithms like k-nearest neighbors and support vector machines (SVM) provide a way of classifying graphs based on a training set of labeled instances. Analysis of current flows in an electrical network is a technique that uses the voltages and currents obtained through nodal analysis of a graph represented as an electrical circuit [7]. A similarity measure, called the *match value*, based on the current flows along geodesics of a graph was proposed in [8]. Here, we propose a modified version of this similarity measure, following a similar analysis of the current flows along the shortest and longest geodesics of a graph, but proposing a different match value evaluation; one that provides a similarity value in the range of 0 to 1.

This paper is organized as follows. In section II, we introduce the concept of electrical nodal analysis for fast discovery of connection subgraphs as presented by Faloutsos et al. Nodal analysis applied to undirected graphs is at the heart of our current flows for error-tolerant graph matching approach, and as shown by [7], [9], in a graph where the edge weights represent the conductance of the edge and vertices represents the nodes of the circuit, the electrical current along an edge is proportional to the net number of times that a random walk along the same edge will traverse it.

In Section III, we show the current flows for error-tolerant graph matching approach as presented in [8], where several geodesics (shortest paths) are extracted from the graph using as starting and ending points nodes of equal degree. Two set of geodesics are then evaluated: shortest geodesics and longest geodesics. Current flow analysis is then performed over the two sets of geodesics in order to produce a n-dimensional vectorial representation of the graph.

Section IV introduces a similarity measure based on n-dimensional vectorial representation of graphs generated using current flows. This new similarity measure is a function $\mathbf{k} : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}$, where two graphs (represented by their current flow vectors) are compared to each other, and a real number between 0 and 1 is returned as the similarity value between the two graphs. As we will observe, this similarity measure can be used as a kernel in a support vector machine, since k is symmetric and non-negative, it can make up a positive definite matrix [10].

In Section V, we describe the implementation details of our prototype, which consists of two main programs: *cf-vectors* and *cf-compare*. ***cf-vectors*** is the program used to generate the vectorial representation of the graphs using current flow analysis. ***cf-compare*** is the program used to compare two set of graphs and generate a similarity value between each pair of graphs among both sets. Other tools were developed as part of this research in order to facilitate the analysis and visualization of the results. These tools are: *sdf2gds* and *gds2dot.* Both of these tools are used to transform the file format used by *cf-vectors* and *cf-compare* to a more standard format.

Section VI describes the results obtained on the NCI-HIV dataset [11]. Some examples of different compounds represented as graphs, and their closest matches are presented in order to provide a graphical comparison between them. As we will show, the ability to store the graph information as a current flows vector, and later, use this representation to find similar graphs (based on the topology) is quite useful. Finding the best match using our similarity measure against a database of more than 40,000 compounds takes a few seconds, and once the current flow vectors have been calculated and stored there is no need to calculate them again as they will remain unchanged for each graph.

II. Related Work

In [7] an approach related to electrical currents in a network of resistors is proposed. This approach tries to solve the problem of finding the connection subgraph that can deliver as many units of electrical current as possible. For this purpose, a graph $\mathcal{G} = (V, E)$ is treated as an electrical network, where edge weights represent conductance ($C(u, v)$ represents the conductance between nodes *u* and *v*), and the vertices represent the nodes of the electrical circuit ($V(u)$ represents the voltage at node u). The voltages at each node of the circuit are calculated by combining Ohm's law and Kirchhoff's current law.

$$\forall u, v : I(u, v) = C(u, v)(V(u) - V(v)) \tag{1}$$

$$\forall v \neq s, t : \sum_u I(u, v) = 0 \tag{2}$$

having *s* as the source node, and *t* as the target node, equations (1) and (2) determine the voltages and currents as the solution to the following linear system:

$$V(u) = \sum_v \frac{V(v)C(u, v)}{C(u)} \quad \forall u \neq s, t \tag{3}$$

$$V(s) = 1, V(t) = 0 \tag{4}$$

$$C(u) = \sum_v C(u, v) \tag{5}$$

solving (3) with boundary conditions (4) will determine the voltages at each node. $C(u)$ represents the total conductance of node u, this is, the sum of all edge weights adjacent to u. Once the current, $I(u, v)$, values are available, the current along a particular path: $\hat{I}(P) \;\; P = \{s, \ldots, t\}$ is defined as the pro-rated current along that path from source to target.

$$\hat{I}(s, u) = I(s, u) \tag{6}$$

$$\hat{I}(s = u_1, \ldots, u_i) = \hat{I}(s = u_1, \ldots, u_{i-1}) \frac{I(u_{i-1}, u_i)}{I_{out}(u_{i-1})} \tag{7}$$

where $I_{out}(u) = \sum_{\{v|u \to v\}} I(u, v)$. This is the total current leaving a node, which is equal to the sum of all currents leaving the node in a downhill stream, where a downhill stream from node *u* to node *v* means that voltage at node *u* is higher than voltage at node *v*, $V(u) > V(v)$. Since the idea of the approach presented in [7] is to find the best connection subgraph, the concept of capture flow $CF(\mathcal{H})$ is introduced. $CF(\mathcal{H})$ of a subgraph $\mathcal{H}$ of $\mathcal{G}$ is the total delivered current, summed over all paths from source *s* to target *t* that belong to $\mathcal{H}$. For our purposes, we are only going to consider single paths in $\mathcal{G}$, not subgraphs of it. The concept of *delivered current* over a path, $\hat{I}(P)$, is very important in the calculation of the current flow vectors in the next section.

III. Current Flow Vectors

Current flow analysis along the shortest and longest geodesics of a graph as presented in [8], provides a method to describe the graph structure such that the information needed to represent the graph is reduced significantly compared to the original size of the graph representation. Once a graph has been described using current flow analysis, its new representation is a n-dimensional vector that stores the current flow along geodesics of different node degrees.

$$\mathcal{G} = (V, E) \qquad \Delta(\mathcal{G}) = \max_{v \in V} \; deg(v) \tag{8}$$

$$f : \mathcal{G} \longrightarrow \mathbb{R}^{2\Delta(\mathcal{G})}, \; 2\Delta(\mathcal{G}) \; is \; an \; upper \; bound. \tag{9}$$

As we can observe from (9), the current flows vector is represented by function f , which transform the input graph $\mathcal{G}$ to a $2\Delta(\mathcal{G})$-dimensional vector, where $\Delta(\mathcal{G})$ represents the highest node degree among all vertices in the graph. The actual dimension of the vector is twice the size of the maximum degree. This is due to the analysis of the current flows along shortest and longest geodesics of the graph. The size of the vector based on the highest vertex degree actually represents an upper bound of the final vector size, this is because for a given graph, some geodesics of an specific node degree may not exist.

In the following sections we will describe the steps needed to perform the transformation from a graph representation $\mathcal{G} = (V, E)$, to a vectorial representation $\mathcal{G} = \mathbb{R}^n$. Section III.A describes the process of geodesic selection, also called *group-N* generation. In Section III.B, we describe the steps needed to calculate the current flow along each of the selected geodesics. Nodal analysis is used as shown in section II in order to calculate the pro-rated current along geodesics.

A. *group-N Generation*

Current flow analysis requires the selection of voltage source *s* and target ground *t* nodes. Once the selection of these nodes has been performed, boundary conditions as described in (4) can be applied to solve the linear system in (3) to find the voltages and currents of the circuit. Path selection is made using shortest paths (geodesics) along the graph. Different source and target nodes will provide different paths, hence different current flows along each path. Each current flow along a particular path will capture different characteristics of the topology of the graph as different connections and flows along each path will be different from each other. The idea then is to select a representative number of paths that will capture as much information as possible about the topology of the graph using current flows. To this end, two different sets of paths are defined: shortest geodesics and longest geodesics.

Since different graphs will render different geodesics, we need a way to pair them when comparing them. A good way to describe the characteristics of a geodesic is based on the node degree of its source and target nodes. In order to provide a standard framework of comparison between current flow along geodesics of different graphs, the selection of geodesics is limited to those in which the source and target nodes have the same node degree.

The concept ***group-N*** makes reference to the group of geodesics that share the same degree ***N*** in their source and target nodes. Each *group-N* will have two sets of geodesics: shortest geodesics and longest geodesics. As noted before, by selecting a representative number of paths along the graph we are providing a way to capture as much information as possible about the topology of the graph. Fig. 1 and Table I provide an example of a graph and its corresponding *group-N* shortest, and longest geodesics.

Table I
group-N GEODESICS FOR GRAPH IN FIG. 1

group-N	Shortest Geodesics	Longest Geodesics
group-1	(8,10)	(8,10)
group-2	(1,2),(1,3)	(1,7)
group-3	(4,5),(4,9),(5,6)	(4,6),(5,9),(6,9)

Geodesics in group-1 are those where their source and target nodes are of degree 1, in this case, nodes 8 and 10. The same applies for other group-Ns. As we can see, multiple geodesics of the same length of the same *group-N* can be generated using different source and target nodes. Current flows along these geodesics are averaged to produce a single current flow value for each *group-N* set.

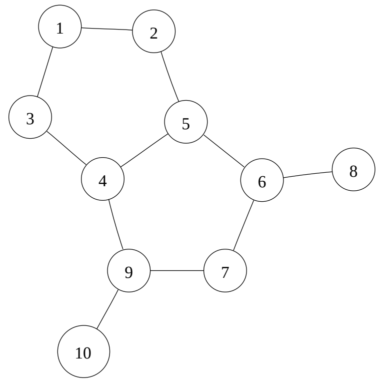

Figure 1. A simple graph used in Table 1 for *group-N* generation

As we mentioned before, the selection of the geodesics is done using a single-source shortest path algorithm from source to target. In this case we are using Dijkstra's algorithm [12]. It is important to note here that edge weights in the graph represent the cost (or resistance) of going from one node to another. This annotation is important in the sense that while calculating the current flows using equations (1), (2), (3), and (4) the value of the edge weights represent the conductance between the nodes rather than resistance. Therefore a conductance equal to the reciprocal of the edge weight is used while performing the nodal analysis.

B. *Current flow along group-N geodesics*

Now that we have described how to generate the *group-N* sets for a given graph, we can proceed with the current flow calculation along each of the geodesics. Calculating currents is done using nodal analysis as described in section II. Each geodesic can be described as a path from source to target. $P = (s, \ldots, t)$. As noted by equation (4) voltage values for source and target nodes are initialized to $V(s) = 1$ and $V(t) = 0$. In this scenario, the source will act as the voltage source and the target as a ground. In the case that the graph has some nodes of degree 1 that are *not* the source nor the target, these nodes are considered to be grounds as well, hence the voltage at these nodes is $V(u) = 0$.

Once the voltage for source node and ground nodes has been specified we can proceed to solve a system of linear equations with *n* variables, where *n* is equal to the total number of vertices of the graph minus source and grounds nodes. This system can be reduced to solving an eigenvector calculation of the form:

$$\begin{bmatrix} A & B \\ 0 & I \end{bmatrix} V = V \quad (10)$$

$$A = \left\{ a_{ij} = \frac{w_{ij}}{w_j} \right\} \quad (11)$$

where matrix ***A*** represents the relationship between the nodes based on their connection weights. ***B*** represents the boundary conditions for *s, t,* and other ground nodes. And ***I*** is the identity matrix. The solution to the system of equations represents the voltages at each of the *n* nodes. With the voltages for each node, we can now calculate the current for each edge using equation (1). Once we have calculated the current along each edge, we need to calculate the current flow along the geodesic. This is done using equations (6) and (7). The current flow is a pro-rated amount from source to target based on the total current along each node in the path and the total current leaving each of the nodes along the path.

Certain considerations need to be taken in order to assure that nodal analysis will yield useful results. On one hand, for disjoint graph representations where certain sections of the graph are not connected to each other, we need to exclude the nodes where there is no path from the source to the node. This can be accomplished by using BFS (Breath-First Search) [12]. This will prevent calculating currents along non-existing connections in the circuit.

On the other hand, certain topology configurations of the graph, in particular where too many ground nodes (nodes of degree 1) are present, and closed rings (cycles) provide alternative routes from the current to flow from source to target avoiding the extra ground nodes, display the potential for an odd distribution of voltages along the geodesic; i.e. voltages along the nodes of the path will not always be in descending configuration from source to target, causing the current flow calculation to yield negative results. To avoid this scenario, we opted to exclude the node pair that causes this behavior from the pro-rated calculation of the current flow as presented in equation (7). This situation is analogue to short-circuiting an electrical network. Fig. 2 shows an example of an scenario where a path from source node *s = 21 O* to target node *t = 13 N*, flows in a downhill stream (as defined in Section II) until it reaches node *9 C*. Since node *9 C* is connected to a ground node (degree of node *12 O* is 1), the current flows down to this node. Current also flows through a closed ring to reach target node *13 N* through node *8 C*. As we can see, since the current flows from *8 C* to *9 C*, if we try to calculate the current along the path (grayed out nodes), we would get negative results. As noted, when a scenario like this one arises, we opted for ignoring the portion of the current between nodes *8 C* and *9 C*, and short-circuit the network from *4 C* to *8 C*.

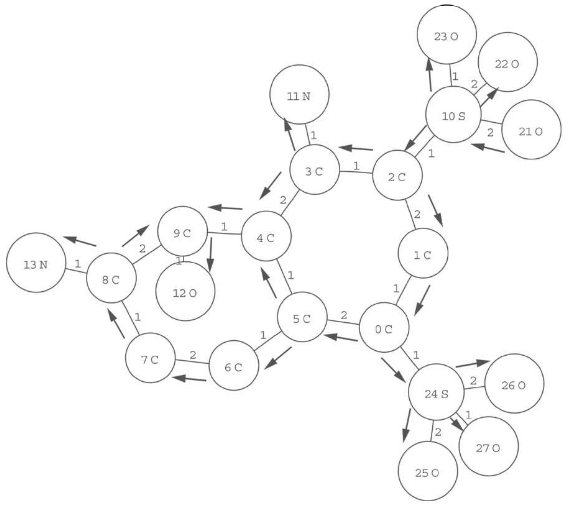

Figure 2. Current flow along path causing short-circuit

Having calculated the current flow along each geodesic we can calculate the current flow for each *group-N* set. For each set (shortest and longest) we average the current of all geodesics of the same degree. This is, for group-Ns with more than one geodesic of the same length we calculate an average current flow. Having calculated all of the group-Ns current flows for shortest and longest geodesics, we can produce our vectorial representation of graph $\mathcal{G}$.

$$\mathcal{G} = \mathbb{R}^{2D} \quad D = \max_{n \in groupN} deg(groupN) \quad (12)$$

The vectorial representation of the graph if defined by:

$$\mathcal{G} = [SN_1, \ldots, SN_D, LN_1, \ldots, LN_D] \quad (13)$$

Where SN_i is the current flow value for the shortest geodesic(s) from *group-i.* Similar, LN_i is the current flow value for the longest geodesic(s) from *group-i.* If a particular degree is not represented in the group-Ns, a value of 0 is assigned to the current flow along that group.

IV. Similarity Measure Based on Current Flow Vectors

The representation of the graph topology as a n-dimensional vector allows us to define a similarity measure between two graphs as a numeric value in the range of [0..1].

$$\mathbf{k} : \mathcal{G} \times \mathcal{G} \to \mathbb{R} \quad (14)$$

We define the similarity measure ***k*** between graphs $\mathcal{G}1$ and $\mathcal{G}2$ as:

$$S = \sum_{d=1}^{D} \frac{|\mathcal{G}1[SN_d] - \mathcal{G}2[SN_d]|}{\mathcal{G}1[SN_d] + \mathcal{G}2[SN_d]}$$

$$L = \sum_{d=1}^{D} \frac{|\mathcal{G}1[LN_d] - \mathcal{G}2[LN_d]|}{\mathcal{G}1[LN_d] + \mathcal{G}2[LN_d]}$$

$$k(\mathcal{G}1, \mathcal{G}2) = 1 - \frac{S + L}{2D} \tag{15}$$

$$D = \max(maxdeg(\mathcal{G}1), maxdeg(\mathcal{G}2)) \tag{16}$$

The value of ***k*** is a real number in [0..1], where the closer the value is to 1, the more similarities are shared between the current flows vector of both graphs. Equation (15) can be described as computing the differences between each pair of group-N geodesics from graphs $\mathcal{G}1$ and $\mathcal{G}2$. The first summation S compares the shortest geodesics from both graphs, while the second summation L compares the longest geodesics from both graphs. As mentioned before, function ***k*** can be used as a graph kernel. A positive definite Gram matrix ***K*** can be constructed from function ***k***, given than k is always positive and symmetric, function k can be referred as *positive definite (pd) kernel [10].*

V. Implementation Details

During the implementation of our prototype we developed two main programs: *cf-vectors* and *cf-compare*. ***cf-vectors*** is the program used to generate the vectorial representation of the graphs using current flow analysis as described in section II. And ***cf-compare*** is the program used to compare two set of graphs and generate a similarity value between each pair of graphs among both sets as described in section IV. In the course of our development we defined two file formats to be used by our programs. These are: graph dataset file (.gds) and current flow vectors dataset file (.cfd). The graph dataset file stores a set of directed graphs as described in the the next abstract grammar:

Graph Dataset : ***Graph+***
Graph: BEGIN graph_name graph_class
    vertices
    edges
  END
vertices: {v vertex_id vertex_label}+
edges: {e from to edge_weight}*

Graph Dataset File (.gds)

The current flow vectors dataset file stores the vectorial representation of the graphs as described in the next abstract grammar:

Graph Dataset : ***Graph+***
Graph: graph_name graph_class **[*S*]*** **[*L*]***
S: S:degree:current_flow_value
L: L:degree:current_flow_value

Current Flow Vectors Dataset File (.cfd)

The current flow vectors dataset format allows for a sparse representation of the current flows. As noted before, not all *group-N* degrees will be present in a graph. Only those degrees present in the graph need to be stored in the *.cfd* files.

cf-vectors receives as a parameter a .gds file, and returns as output a .cfd file. Once the translation from the graph representation to the vectorial representation of the graph has been performed using *cf-vectors*, there is no need to perform this step again on the same dataset.

cf-compare provides several options to compare two .cfd files. The two datasets to be compared are called: query dataset, and base dataset. The query dataset is usually a smaller dataset that we want to compare against our base dataset. Since the number of results that can be obtained from comparing the query dataset to the base dataset is equal to the number of graphs in the query dataset multiplied by the number of graphs in the base dataset, *cf-compare* provides the ability to limit the number of results to avoid generating huge output files. These options are ***-n*** and ***-t***. Option *-n* allows the user to define the top N results to be generated. Option -*t* allows the user to define a value from 0 to 1, this value represents a threshold for the similarity measure, meaning that only graphs where the similarity measure is equal or greater than the provided threshold value would be returned. Results are stored in a text file that shows the name of the graph being compared, followed by the graphs that met the criteria provided by the user (either top N, above or equal to threshold, or all base graphs) in descending order based on the similarity value (closest matches are listed first), this helps to identify the closest matches in a more efficient manner. In case the results are needed for classification purposes, *cf-compare* can provide counts based on the class labels in the base dataset. Option ***-c*** allows the user to request class counts to be included. Class counts will be generated in a separate file from the results, showing the total number of graphs from each class that met the criteria provided by the user.

Other tools were developed as part of this research in order to facilitate the analysis and visualization of the results. These tools are: *sdf2gds* and *gds2dot.* ***sdf2gds*** converts an *.sdf* file also known as Structures Data File which is a common file format developed by Molecular Design Limited to handle a list of molecular structures with associated properties [13] into a .gds file, which is the format expected by *cf-vectors*. ***gds2dot*** exports each graph in the graph dataset file to separate .dot files for each graph. *.dot* files as defined by [14] are used by Graphviz as its input format. Graphviz is a popular open source suite of tools developed by AT&T research labs for graph visualization.

VI. Experimental Results

In order to test the our prototype, we applied our program to the NCI-HIV dataset of chemical compounds [11]. This dataset contains 42,689 molecules, 423 of which are active (CA), 1081 are moderately active (CM), and 41,185 are inactive (CI). This dataset has been used in the empirical evaluation of several graph mining techniques [15], [16], [17], [18]. The first step of our experiments was to convert the NCI-HIV dataset from .sdf to .gds. We used *sdf2gds* for this purpose. Once we had our graph dataset file, the next step was creating the current flow vectors file. Using *cf-vectors* on the NCI-HIV.gds file took a little over 10 minutes. With the NCI-HIV.cfd file at hand, we were ready to compare some graphs against our *base dataset*. The next step in our research was to find out how the similarity measure worked at finding isomorphisms. We compared our base dataset against itself with a threshold of 1.0. Due to the sheer size of the dataset we cannot visually verify each of the results, but after a random verification of several compounds, each and every one of those that were visually verified were perfect isomorphisms (excluding the same graph compare to itself). This of course, by no means allows us to present a sound statement about the efficiency of our prototype, but it is encouraging to see the excellent results achieved when finding isomorphisms.

Comparing a single graph against the whole 42,689 takes a few seconds. Here is an example of the results obtained for compound 669337CI shown in Fig. 3:

Figure 4 shows the closest match for compound 669337CI, which in this case is an isomorphic graph, compound 669335CI, with a similarity value of 1.0.

Figure 5 shows the next best match, compound 676178CI. The similarity value for this compound was 0.99606. As we can observe there are some small differences in the topology of both graphs that account for the differences in the current flows.

In order to evaluate the predictive power of current flow vectors on the NCI-HIV dataset we compare the results against the frequent subgraph discovery algorithm (FSG). In [16], [17], M. Deshpande et al. investigated

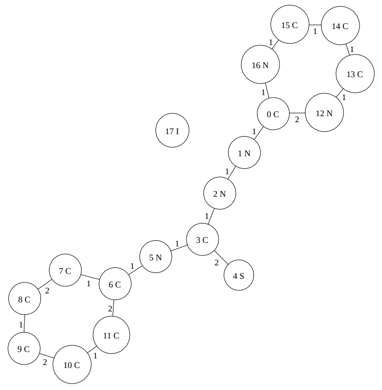

Figure 3. compound 669337CI

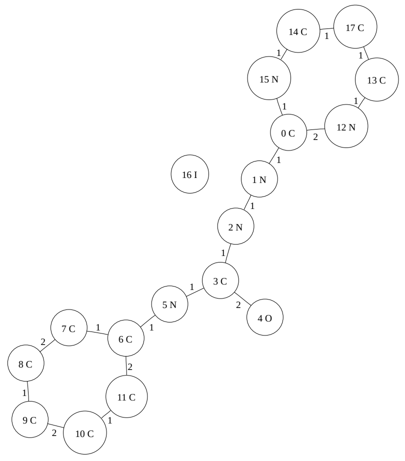

Figure 4. compound 669335CI : 1.0000

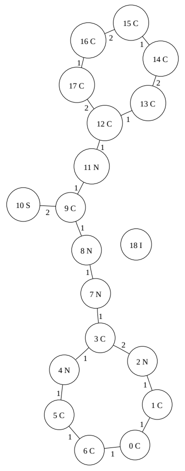

Figure 5. compound 676178CI : 0.99606

Table II
CURRENT FLOW VECTORS RESULTS ON NCI-HIV DATASET

Dataset	Classifier	Options	AUC-CF	AUC-FSG (cost 1.0)	Difference
E11_Top36	Neural Network	-L 0.3 -M 0.2 -N 150 -V 0 -S 0 -E 20 -H 2	0.781 (±0.022)	0.774	0.00672253
E21_Top57	Neural Network	-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H 4	0.715 (±0.013)	0.742	-0.027511452
E31_Top10	Neural Network	-L 0.3 -M 0.2 -N 150 -V 0 -S 0 -E 20 -H 4	0.865 (±0.022)	0.839	0.026053254

Table III
STATISTICAL SIGNIFICANCE OF THE RESULTS

Class. Problem	AUC-CF	AUC-FSG (cost 1.0)	Mean Diff	STDev	Non-Paired T-test	Confidence Level
(1) CA vs CM	0.781	0.774	0.007	0.022	0.503089628	68% (win)
(2) CA+CM vs CI	0.715	0.742	-0.027	0.013	-3.283903724	98% (loss)
(3) CA vs CI	0.865	0.839	0.026	0.022	1.868618617	93% (win)

the predictive power of the FSG algorithm using support vector machines (SVM) as the classification technique. The use of SVM enabled them to associate a higher cost for the miss-classification of positive instances.

Three different classification problems were defined in [16] and [17]: (1) distinguish CA from CM, (2) distinguish CA+CM from CI, and (3) distinguish CA from CI. We compare the results of current flow vectors for each of these classification problems. The first step in our experiment was to generate the current flow vectors for all 42689 compounds. This step took approximately 10 minutes. The second step was to compare each of the compounds against ALL of the other compounds in the dataset using our k(G1,G2) function (excluding the compound compared to itself). The comparison took a little over 6 hours. The results of the second step were the similarity values for each of the compounds in the dataset compared to all the other compounds. Given the size of the dataset we only stored the 100 closest matches for each compound. Having the top 100 matches for each compound we generated 100 different datasets for each classification problem. Each dataset, which we will refer to as EM_topN, with M = [1..3] and N=[1..100], contains 6 attributes (CA, CM, CI, MaxCA, MaxCM, MaxCI) and the class (C1 or C2). For example, the dataset called E1_top10 stores in the first three attributes the number of compounds per class (CA, CM, CI) within the top 10; the other three attributes store the higher similarity value (value returned by function k) for each class within the top 10 (MaxCA, MaxCM, MaxCI). The final field in the E1_top10 dataset contains the actual class for each compound; this is used for supervised learning. The prefix E1 indicates that the dataset contains the compounds for classification problem (1) CA vs. CM. In this case the class value in the dataset will be C1 for CA, and C2 for CM. For classification problem (2), C1 represents CA and CM joined together while C2 represents CI. With the 300 datasets at hand, we used a variety of classification algorithms such as naïve Bayes, back-propagation neural networks, and support vector machines in order to attempt to classify the compounds based on the six attributes defined. Each of the algorithms were tested with several parameters in order to find the best set of settings for each algorithm. This was done using Weka 3.5.6. In order to determine the best combination of dataset/algorithm/settings we performed a 5x2 cross-validation with an F-test. The winner on all three classification problems was a back-propagation neural network.

In [16], [17], each classification problem was evaluated using a five-fold cross-validation and ROC curves. In order to determine statistical significance when comparing the results of current flows vectors against FSG we obtained an area under the curve (AUC) average over a five-fold cross-validation. Since we do not have the variance of the AUC for the FSG results we will assume the same sampled pool variance as the one for current flow vectors. Tables II and III show the results for each of the three classification problems comparing current flows to FSG. As we can observe from the results, current flow vectors performed better on classification problems (1) and (3). In classification problem (2) FSG outperformed current flow vectors. Based on the analysis of statistical significance we can see that in classification problem (1) current flow vectors performed better but only with a 68% level of confidence that there is statistical significance. On the other hand, performance on classification problem (3) showed statistical significance at 93% favoring the results obtained when using current flow vectors. In classification problem (2) FSG outperformed current flows and it is clear given the high level of confidence, 98%, that the results for this particular problem were better than current flow vectors.

VII. Conclusions

Current flow analysis as a tool for error-tolerant graph matching holds the potential to be a very powerful approach for structural graph similarity. This technique can prove being very valuable in datasets were the topological information of the graphs, rather than vertex labels, hold the information stored in the graph. Examples of such graph datasets are fingerprint matching and handwriting recognition. As shown in our empirical results, current flow analysis emerges as a promising technique to detect graph isomorphisms, even on datasets where the vertex label information is important, as seems likely in the case of chemical compounds. Similar graph structures could provide hints about the chemical composition, and current flow similarity could yield good results to find similar compounds. Future work analyzing other datasets and further exploration of the classification capabilities of the current flow similarity measure is needed in order to develop the full potential of this promising technique. Further improvements to the current flow vectors technique such as inclusion of vertex information within the current flow calculation should increase the predictive power of this technique.

References

[1] B. T. Messmer and H. Bunke, "A new algorithm for error-tolerant subgraph isomorphism detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 5, pp. 493–504, 1998.

[2] M. Neuhaus and H. Bunke, "Edit distance-based kernel functions for structural pattern classification," *Pattern Recogn.*, vol. 39, no. 10, pp. 1852–1863, 2006.

[3] H. Bunke and K. Shearer, "A graph distance metric based on the maximal common subgraph," *Pattern Recogn. Lett.*, vol. 19, no. 3-4, pp. 255–259, 1998.

[4] D. Justice, "A binary linear programming formulation of the graph edit distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 8, pp. 1200–1214, 2006, fellow-Alfred Hero.

[5] M.-L. Fernández and G. Valiente, "A graph distance metric combining maximum common subgraph and minimum common supergraph," *Pattern Recogn. Lett.*, vol. 22, no. 6-7, pp. 753–758, 2001.

[6] W. D. Wallis, P. Shoubridge, M. Kraetz, and D. Ray, "Graph distances using graph union," *Pattern Recogn. Lett.*, vol. 22, no. 6-7, pp. 701–704, 2001.

[7] C. Faloutsos, K. S. McCurley, and A. Tomkins, "Fast discovery of connection subgraphs," in *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2004, pp. 118–127.

[8] L. O. Hall and A. Hildoer, "Compound matching using current flows," 2006.

[9] P. G. Doyle and J. L. Snell, "Random walks and electric networks," *Mathematical Association America*, vol. 22, 1984.

[10] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[11] DTP, "AID2DA99 42,689 2d structures with aids test data as of october 1999, in sdf format." Downloaded from http://cactus.nci.nih.gov/ncidb/download.html, Oct. 1999.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[13] A. Dalby, J. G. Nourse, W. D. Hounshell, A. K. I. Gushurst, D. L. Grier, B. A. Leland, and J. Laufer, "Description of several chemical structure file formats used by computer programs developed at molecular design limited," *Journal of Chemical Information and Computer Sciences*, vol. 32, no. 3, pp. 244–255, 1992.

[14] E. Gansner, E. Koutsofios, and S. North, "Drawing graphs with `dot`," AT&T Bell Laboratories, Murray Hill, NJ, USA, Technical Report, Feb. 2002. [Online]. Available: http://www.research.att.com/sw/tools/graphviz/dotguide.pdf

[15] C. Borgelt and M. R. Berthold, "Mining molecular fragments: Finding relevant substructures of molecules," in *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*. Washington, DC, USA: IEEE Computer Society, 2002, p. 51.

[16] M. Deshpande, M. Kuramochi, and G. Karypis, "Automated approaches for classifying structures," in *BIOKDD*, 2002, pp. 11–18.

[17] ——, "Frequent sub-structure-based approaches for classifying chemical compounds," in *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2003, p. 35.

[18] S. Kramer, L. D. Raedt, and C. Helma, "Molecular feature mining in hiv data," in *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2001, pp. 136–143.