A Linear Formulation of the Graph Edit Distance for Graph Recognition

Derek Justice and Alfred Hero
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109

Abstract

An efficient graph matching algorithm based on optimizing the graph edit distance is presented. The graph edit distance is expressed as a linear function of a permutation matrix and a sequence of edit matrices which represent graph edit operations. This allows the development of a linear program that is solved using an interior point method. The linear optimization produces a continuous analog to the permutation matrix that is used as a weight matrix for an instance of the well-known assignment problem. The assignment problem is solved as usual with the Hungarian method to produce a permutation matrix. A standard recognition problem of matching a sample input graph to a database of known prototype graphs is presented as an application of the new method. The costs associated with various edit operations are chosen using a minimum variance criterion applied to pairwise distances between nearest neighbors in the database of prototypes. The new approach is shown to provide significant reduction in classification ambiguity.

Index Terms

Graph algorithms, Similarity measures, Structural Pattern Recognition, Graphs and Networks, Linear Programming, Continuation (homotopy) methods

I. Introduction

Graphs provide convenient structures for representing objects when relational properties are of interest. Such representations are frequently useful in machine vision applications, where the problem may be to recognize specific objects within an image. In this case, the image is processed to generate a representative graph based on structural characteristics, such a region adjacency graph or a line adjacency graph [1]. This representative graph is then compared to a database of prototype or model graphs in order to identify and classify the object of interest. Face identification [2] and symbol recognition [3] are among the problems in machine vision where graphs have been utilized recently. In this context, a reliable and speedy method for graph matching in the presence of noise is important.

The problem of graph matching is to find a mapping from the nodes of a graph G_1 to another graph G_2 that makes the two graphs as similar as possible in some sense. Frequently used similarity metrics include the size of the largest subgraph common to both graphs and the norm of a difference between permuted adjacency matrix representations of the two graphs. A more recently developed metric based on a similarity score between graph vertices is discussed in [4].

The graph edit distance is a convenient and logical similarity metric in the presence of errors [5], [6], [7]. It is a natural extension of the string edit distance for string processing [8]. The basic idea is to define graph edit operations such as insertion or deletion of a node or vertex along with costs associated with each operation. The graph edit distance between two graphs is then just the cost associated with the least costly series of edit operations needed to transform one graph into another. Because the edit operations correspond to ’errors’ between the two graphs, this approach is often referred to as error correcting graph matching. Furthermore, some other similarity metrics can be equated to a graph edit distance with a certain cost function [6].

The graph edit distance is parameterized by a set of edit costs. The flexibility provided by these costs can be very useful in the context of a standard recognition problem of matching a sample input graph to a database of known prototype graphs [9]. If chosen appropriately, the costs can capture the essential features that characterize differences among the prototype graphs. Recently, methods for choosing these costs that are best from a recognition point of view have been presented. In [10], the EM algorithm is applied to assumed Gaussian mixture models for edit events in order to choose costs that enforce similarity (or dissimilarity) between specific pairs of graphs in a training set. In addition to the complexity of the parameterization involved, this algorithm is exponential in the number of nodes.

The (weighted) graph matching problem was shown in [11] to be neither NP-complete nor to have an efficient algorithm, so consequently there are a wide variety of algorithms that attempt to tackle this problem. Optimal algorithms inevitably resort to enumeration and searching [12], [13], [5]. Because there are an exponential number of enumerations, these methods are only useful for small graphs (i.e. graphs not having much more than ten nodes).

Many suboptimal approaches exist that may be effectively applied to matching of larger graphs [14], [15], [16], [17], [18], [19]. The adjacency matrix eigendecomposition approach of [15] gives fast suboptimal results, however it is only applicable to adjacency matrices with no repeated eigenvalues. Graphs with a low degree of connnectivity will often have adjacency matrices with multiple zero eigenvalues. Heuristics are used in the graduated assignment type methods of [16], [18] to significantly reduce the exponential complexity of the original problem. These methods can be applied to very large graphs; however they require several tuning parameters to which the performance of the algorithm is quite sensitive. Unfortunately, no systematic method for choosing these parameters is provided. The linear programming approach of [14] gives good results in a reasonable amount of time ($O(n^6L)$) in the presence of noise. The authors of [14] use the linear program to minimize a matrix norm similarity metric. Speedy algorithms for exact graph matching (the case where a vertex mapping exists that causes the graph structures to exactly coincide) are provided in [19], [17].

In this paper, we will provide a linear formulation of the graph edit distance so that error correcting graph matching may be performed by solving a linear program. Edit operations are introduced in the form of slack variables. A cost structure is then added in a linear fashion to multiply the edit operations and thus give a graph edit distance. A linear program is constructed to solve for the vertex mapping and edit operations that minimize the graph edit distance specified by a cost structure. Although a continuous optimization is performed, the vertex mapping and edit operations obviously reside in a discrete space. The Hungarian method for weighted bipartite graph matching [20] is therefore applied to the result of the linear program to produce a permutation mapping.

We also present a standard recognition problem [9] that demonstrates the utility of the new method. Suppose there is a database of prototype graphs to which a sample graph is to be matched as in the motivating machine vision example. We provide a method for choosing the edit costs that is purely nonparametric and has polynomial runtime (in the size of the graph). The edit costs are chosen as those that minimize the variance of pair wise distances between nearest neighbor prototypes. The motivation behind this cost selection is that a metric which maximally separates nearest neighbors in the database should be more discriminating in graph classification tasks. This computation involves converting all prototype graphs to a canonical form, which requires at most $\frac{n^2-n}{2}$ graph matchings, and tabulating the edits between canonical prototypes. These are provided as inputs to a single quadratic program to solve for the optimal edit costs.

The algorithm for graph matching and cost selection is tested on several databases of fifteen prototype random graphs. Within each database, fifteen different sample graphs are generated by slightly perturbing each of the prototypes. Each sample graph is matched to every prototype in the database, and a classification ambiguity index is computed. The new method is shown to provide significant reduction in classification ambiguity.

This paper is organized as follows. Section II presents the bulk of the theory. Within Section II, notation is described first, and then we present the linear formulation of the graph edit distance. This is followed by the development of a linear program to minimize the graph edit distance, and finally a description of edit cost selection for a graph recognition problem. Section III presents simulation results of the new method applied to the recognition problem, and Section IV provides some concluding remarks.

II. Theory

Before formally defining and outlining a solution to the graph edit optimization problem, we will first define some general notation associated with graphs and then develop the linear formalism of the graph edit distance.

A. Notation

We will consider undirected, unmarked, simple graphs $G(V, E, f)$ defined by a set of vertices V, a set of edges E, and an incidence relation f. For convenience, the vertices and edges are labeled by the natural numbers so that $V = \{1, 2, \ldots, |V|\}$ and $E = \{1, 2, \ldots, |E|\}$. The incidence relation associates a pair of unordered endpoint vertices with each edge so that $f : E \to V \times V$.

The adjacency matrix representation A of a graph G will be used frequently and is defined below.

$$A_{ij} = \begin{cases} 1 & \text{if } \exists e \in E \text{ such that } f(e) = \{i, j\} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Note that since the graphs are undirected, $A = A^T$. Furthermore, it is clear that since A is binary, all connected edges have equal costs.

B. Linear Formulation of the Graph Edit Distance

It is desired to transform any undirected simple graph into any other undirected simple graph by performing a series of graph edit operations. Clearly the following operations taken together can accomplish any such transformation: edge deletion (ed), edge insertion (ei), vertex deletion (vd), and vertex insertion (vi). Substitution of edge or vertex labels obviously does not change the structure of the graph, so these are not considered (or equivalently, label subsitutions carry zero cost). Let $G(V, E, f)$ be a graph that is transformed to $G'(V', E', f')$ after performing an edit operation. The edit operations may then be formally defined as in Def. 1. The effect of each operation on the adjacency matrix representation of the edited graph A' is given in parentheses. Note that in vertex edits, a rank ordering of the vertices is assumed without loss of generality such that the vertex to be edited (inserted or deleted) is always represented by the last row and column of the adjacency matrix.

Definition 1 *Graph Edit Operations (Corresponding Adjacency Matrix Edits)*

1) **edge deletion***: Edge e with $f(e) = \{i, j\}$ for some vertices i and j is deleted by assigning $f'(e) = \phi$ (ϕ is the empty set) and assigning $E' = E - e$. ($A'_{ij} = 0$, $A'_{ji} = 0$)*
2) **edge insertion***: Edge e with $f(e) = \phi$ is inserted by assigning $f'(e) = \{i, j\}$ for some vertices i and j and assigning $E' = E \cup e$. ($A'_{ij} = 1$, $A'_{ji} = 1$)*
3) **vertex deletion***: Vertex i is deleted by assigning $f'(e) = \phi$ for all e such that $i \in f(e)$ and assigning $V' = V - i$. ($A' = M_{NN}$ where $A \in \Re^{N \times N}$ and M_{ij} denotes the minor formed by removing row i and column j from matrix A)*
4) **vertex insertion***: Vertex i is inserted by assigning $V' = V \cup i$.* $\left(A' = \begin{bmatrix} A & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \right)$

Now suppose that c_{--} denotes the cost of a specific edit operation that is required to transform a graph X into another graph Y. The graph edit distance may then be defined:

$$d(X, Y) = \sum_{e \in E_X - E_Y} c_{ed}(e) + \sum_{e \in E_Y - E_X} c_{ei}(e) + \sum_{v \in V_X - V_Y} c_{vd}(v) + \sum_{v \in V_Y - V_X} c_{vi}(v) \tag{2}$$

Assume X and Y are both adjacency matrix representations. With the goal in mind of developing a linear optimization approach to compute $d(X, Y)$, define the 'edit matrices' S and T as

$$\begin{aligned} S_{ij} &= (Y_{ij} - X_{ij})(Y_{ij} > X_{ij}) \\ T_{ij} &= (X_{ij} - Y_{ij})(X_{ij} > Y_{ij}) \end{aligned} \tag{3}$$

where $a > b$ is equal to unity if the statement is true and zero if it is false. If vertex deletions or insertions occur, then the smaller of $\{X, Y\}$ is zero padded so that they have the same dimension. S and T are appropriately referred to as 'edit matrices' because $S_{ij} = 1$ indicates an edge was inserted between vertices i and j in going from X to Y and similarly, $T_{ij} = 1$ indicates an edge was deleted between vertices i and j in going from X to Y. The following constraint equation follows immediately from the definitions in Eq. (3):

$$X - Y + S - T = 0 \tag{4}$$

Furthermore, the graph edit distance in Eq. (2) may be re-expressed as

$$d(X,Y) = \frac{1}{2}\sum_{i,j}\left(c_{ed}(e_{ij})T_{ij} + c_{ei}(e_{ij})S_{ij}\right) + \sum_{v \in V_X - V_Y} c_{vd}(v) + \sum_{v \in V_Y - V_X} c_{vi}(v) \tag{5}$$

Where e_{ij} corresponds to the edge connecting vertices i and j. We wish to express the vertex edits in terms of the edit matrices in order to make the problem fully linear. Without loss of generality, we will allow either vertex deletions or vertex insertions to occur in a given sequence of edit operations, but not both. This is a reasonable assumption, since it allows one to proceed as follows: add vertices to the smaller of the two graphs or delete vertices from the larger one until both graphs have the same number of vertices. Finally edit the edges to make the graphs match.

For illustration, consider the case where vertices are deleted. Suppose N_X is the number of vertices in graph X and N_Y is the number of vertices in graph Y. If a vertex deletion has occurred in going from X to Y, then $N_X > N_Y$. Let $v \in V_X - V_Y$ be a deleted vertex and $\gamma(v)$ be the number of other vertices to which it is connected, i.e. vertex degree. The following relation between the elements of the edit matrix T and the vertex degree holds.

$$\sum_j T_{vj} = \sum_i T_{iv} = \gamma(v) \tag{6}$$

The above relationship allows the graph edit distance in Eq. (5) to be re-expressed as

$$d(X,Y) = \frac{1}{2}\sum_{i,j}\left(c_{ed}(e_{ij})T_{ij} + c_{ei}(e_{ij})S_{ij}\right) + \frac{1}{2}\sum_{i \in V_X - V_Y} \frac{1}{\gamma(i)} c_{vd}(i) \sum_j \left(T_{ij} + T_{ji}\right) \tag{7}$$

An expression similar to Eq. (7) may be written in the case of vertex insertion. In order to simplify further, define the following cost matrices $C^{x\gamma}$ and C^{x}, where $x \in \{d, i\}$.

$$C^{x\gamma} = \frac{1}{2}\times$$

$$\begin{pmatrix} 0 & c_{ex}(e_{12}) & \cdots & c_{ex}(e_{1N_s}) & c_{ex}(e_{1(N_s+1)}) + c_{\gamma x}(N_s+1) & \cdots & c_{ex}(e_{1N_l}) + c_{\gamma x}(N_l) \\ \ddots & 0 & \cdots & c_{ex}(e_{2N_s}) & c_{ex}(e_{2(N_s+1)}) + c_{\gamma x}(N_s+1) & \cdots & c_{ex}(e_{2N_l}) + c_{\gamma x}(N_l) \\ \ddots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \ddots & \ddots & \ddots & 0 & c_{ex}(e_{N_s(N_s+1)}) + c_{\gamma x}(N_s+1) & \cdots & c_{ex}(e_{N_sN_l}) + c_{\gamma x}(N_l) \\ \ddots & \ddots & \ddots & \ddots & 0 & \cdots & c_{ex}(e_{(N_s+1)N_l}) + c_{\gamma x}(N_l) \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \end{pmatrix}$$

$$C^{x} = \frac{1}{2}\begin{pmatrix} 0 & c_{ex}(e_{12}) & \cdots & c_{ex}(e_{1N}) \\ \ddots & 0 & \cdots & c_{ex}(e_{2N}) \\ \ddots & \ddots & \vdots & \vdots \\ \ddots & \ddots & \ddots & 0 \end{pmatrix} \tag{8}$$

Where $N_s = min(N_X, N_Y)$, $N_l = max(N_X, N_Y)$, and $c_{\gamma x}(i) = \frac{1}{\gamma(i)} c_{vx}(i)$. Also, since the graphs of interest are undirected, the matrices in Eq. (8) are symmetric. With these defined, the graph edit distance may be written quite compactly:

$$d(X,Y) = \begin{cases} \sum_{i,j} \left(C^{i} * S + C^{d\gamma} * T\right)_{ij} & \text{if } N_X > N_Y \\ \sum_{i,j} \left(C^{i\gamma} * S + C^{d} * T\right)_{ij} & \text{if } N_X < N_Y \\ \sum_{i,j} \left(C^{i} * S + C^{d} * T\right)_{ij} & \text{if } N_X = N_Y \end{cases} \tag{9}$$

where $A * B$ indicates the element by element multiplication of matrices A and B having equal dimension.

C. Graph Matching by Linear Optimization

We now state the graph matching problem formally using the linear formulation of the graph edit distance. If A_1 and A_2 are the adjacency matrix representations of two graphs to be matched (appropriately zero padded if they are originally different sizes), then the graph matching problem is to find a permutation matrix P and edit matrices S and T such that the graph edit distance $d(A_1, PA_2P^T)$ is minimized. Clearly, this is equivalent to finding P, S, and T that minimize $d(A_1P, PA_2)$ since right multiplying by a permutation matrix will simply reorder the sum of edit operation costs.

The problem will be formulated as a linear optimization problem in standard form [20] to minimize the graph edit distance of Eq. (9) subject to the natural constraint of Eq. (4) and an additional constraint that should be added. Since P is a permutation matrix, its row and column sums should be equal to unity. Since A_1, A_2, and P are all of size N after zero padding, the following constraint is also imposed.

$$\begin{aligned} P\mathbf{1}_N &= \mathbf{1}_N \\ \mathbf{1}_N^T P &= \mathbf{1}_N^T \end{aligned} \tag{10}$$

In order to restate the linear optimization in standard form, unknown matrices in the problem will be vectorized, i.e. have their columns 'peeled' off in order and arranged in a vector. Let

$\tilde{A} = VEC(A)$ represent the vector corresponding to matrix A as shown below.

$$\tilde{A} = VEC(A) = \begin{pmatrix} A_{11} \\ A_{21} \\ \vdots \\ A_{N1} \\ A_{12} \\ A_{22} \\ \vdots \\ A_{N2} \\ \vdots \\ A_{1N} \\ A_{2N} \\ \vdots \\ A_{NN} \end{pmatrix} \tag{11}$$

The kronecker product (denoted by $\otimes$) is useful for simplifying notation when VEC operations are involved. Vectorizing the graph edit distance minimization problem as well as the constraint equations allows the linear program that solves for the optimal vertex mapping (P) and edit matrices (S, T) to be written as follows.

$$\begin{aligned} &\underset{v}{min} \;\; c^T v \\ &\text{such that} \;\; Av = b \\ &\text{and} \;\; 0 \leq v_i \leq 1 \end{aligned} \tag{12}$$

Where

$$v = \begin{pmatrix} \tilde{P} \\ \tilde{S} \\ \tilde{T} \end{pmatrix}$$

$$c^T = \begin{cases} \begin{pmatrix} \mathbf{0}_{N^2}^T & (\tilde{C}^i)^T & (\tilde{C}^{d\gamma})^T \end{pmatrix} & \text{if } N_1 > N_2 \\ \begin{pmatrix} \mathbf{0}_{N^2}^T & (\tilde{C}^{i\gamma})^T & (\tilde{C}^{d})^T \end{pmatrix} & \text{if } N_1 < N_2 \\ \begin{pmatrix} \mathbf{0}_{N^2}^T & (\tilde{C}^i)^T & (\tilde{C}^{d})^T \end{pmatrix} & \text{if } N_1 = N_2 \end{cases} \tag{13}$$

$$A = \begin{pmatrix} I_N \otimes A_1 - A_2^T \otimes I_N & I_{N^2} & -I_{N^2} \\ I_N \otimes \mathbf{1}_N^T & 0 & 0 \\ \mathbf{1}_N^T \otimes I_N & 0 & 0 \end{pmatrix}$$

$$b = \begin{pmatrix} \mathbf{0}_{N^2} \\ \mathbf{1}_N \\ \mathbf{1}_N \end{pmatrix}$$

The linear program in Eq. (12) is solved using an interior point method [21]. Note that the optimal vertex mapping P^* that minimizes the linear program may not be a 0-1 permutation matrix. It may give fractional weights for each $V_1 \rightarrow V_2$ mapping. The permutation matrix chosen is the one that maximizes the sum of these possibly fractional weights. The resulting problem is known as the assignment problem in combinatorial optimization and can be solved by the Hungarian method in $O(n^3)$ operations [20]. For completeness, this optimization for a permutation matrix P with weight matrix P^* computed via the linear program of Eq. (12) is specified by

$$\begin{aligned} P = \underset{X\in\{0,1\}^{N\times N}}{arg.max} \quad & \textstyle\sum_{i,j} P_{ij}^* X_{ij} \\ \text{such that: } \; & \mathbf{1}^T X = \mathbf{1}^T \;\text{ and }\; X\mathbf{1} = \mathbf{1} \end{aligned} \tag{14}$$

The L^1 norm adjacency matrix matching criterion used in [14] turns out to be a special case of this more general formulation. This criterion is equivalent to a graph edit distance minimization with costs $c_{ei}(e) = c_{ed}(e) = 1$ for all $e \in E$ and $c_{vi}(v) = c_{vd}(v) = 0$ for all $v \in V$.

D. Edit Cost Selection for Graph Recognition

The expression for the graph edit distance given in Eq. (9) is parameterized by the edit costs and vertex connectivities contained in the matrices C^x and $C^{x\gamma}$ of Eq. (8). We propose an empirical method for selecting these parameters suitable for a recognition problem. Suppose there is a set of prototype graphs $\{G_i\}_{i=1}^N$, and we classify a sample graph G_o by selecting the prototype that most closely matches it. It is easier to classify a sample with a greater degree of confidence if the prototypes are separated by sufficiently large and relatively uniform graph edit distances. If all prototypes are separated by very large graph edit distances (say for unity costs), then little will be gained by carrying out an intricate computation to determine the edit costs. However, in the case that the prototypes are very similar and perhaps unevenly distributed (with respect to the graph edit distance), proper selection of edit costs is helpful.

In order to uniformly distribute the prototypes, we select a set of edit costs that minimizes the variance of distances between pairs of nearest neighbor prototypes–that is ’nearest’ with respect to a graph edit distance with unity cost structure. For the notion of uniformity to be meaningful, it is clear that the graph edit distance must be symmetric, thus forcing $c_{ei}(e) = c_{ed}(e)$ and $c_{vi}(v) = c_{vd}(v)$ for all edges e and vertices v. The edit costs are computed by first converting all prototypes to a sort of ’canonical form’ by matching them to a reference prototype using unit edit costs. This amounts to computing the optimal permutation P_i for each protoype G_i and permuting each adjacency matrix to match the reference.

After converting the graphs to this canonical representation, we tabulate the edits necessary to match each graph with its n nearest neighbors. We consider each nearest neighbor pair only once. For example if G_i has G_j as one of its n nearest neighbors and G_j has G_i as one of its n nearest neighbors, then the edits necessary to match G_i to G_j are tabulated only once. Let $\{N_j\}_{j=1}^K$ be a set of binary vectors that indicates whether a specific edit is necessary to match the j^{th} pair of nearest neighbor prototypes, where K is the number of distinct nearest neighbor pairs.

If c is a vector containing the corresponding edit costs, then the graph edit distance between the j^{th} pair is given by $d_j = N_j^T c$. Using this notation, the variance of pair wise distances is then given by

$$\sigma_d^2 = \frac{1}{K}\sum_{i=1}^{K}\left(N_i^T c - \frac{1}{K}\sum_{j=1}^{K} N_j^T c\right)^2 \tag{15}$$

Eq. (15) may be expanded to give the objective function $\Psi(c)$ as

$$\Psi(c) = K\sigma_d^2 = c^T\left[\sum_{i=1}^{K}\left(N_i - \frac{1}{K}\sum_{j=1}^{K} N_j\right)\left(N_i - \frac{1}{K}\sum_{j=1}^{K} N_j\right)^T\right] c \equiv c^T Q c \tag{16}$$

We also require a minimum allowable distance d_{min} between any two nearest neighbors, and that costs be nonnegative. So if we define the matrix $M^T = \begin{bmatrix} N_1 & N_2 & \cdots & N_K \end{bmatrix}$, then the optimal costs are given by the quadratic program:

$$\begin{aligned} &\underset{c}{min}\ \ c^T Q c \\ &\text{such that}\ \ Mc \geq d_{min}\mathbf{1} \\ &\text{and}\ \ 0 \leq c_i \end{aligned} \tag{17}$$

A projection method may be used to solve this quadratic program; we use an active set method which falls under this broader category [22]. We now turn our attention to some practical issues associated with implementing this method for edit cost selection. In order to assure the problem in Eq. (17) is well posed, we only consider edits that are not dependent–that is edits that do not always occur in tandem. For example, whenever a vertex is edited so are all edges connected to it; it follows that these edits are dependent. Dependent edits are assigned the same optimal cost. Similarly, some edits may not occur at all among the pairs of prototypes. Such edits have no bearing on the distribution of the prototypes, so they are simply assigned unit cost.

It is also important to consider the canonical graph representation used to tabulate edits. It is convenient to use the prototype with the most vertices as the reference graph. This allows the connectivities of all edited vertices ($\gamma(v)$) to be noted directly. We also observe that this canonical representation is only an approximation, albeit a good one for fairly similar graphs. Ideally, the tabulated edits should be equivalent (to within a permutation) regardless of which prototype is selected as the reference. This will certainly not be the case for prototype sets that vary wildly; however, as argued earlier, edit cost selection is not vital in this case.

III. Simulation Results

The matching scheme described was simulated in the context of random graph recognition. Three sets of fifteen prototype random graphs each were generated through a rejection sampling process to assure relatively similar structure of all protoypes within a set. The protoypes range in size from 15 to 18 vertices with 41 to 55 edges. The adjacency matrix representations of all prototypes are shown in Fig. 1. Here, a black square represents a '1' in the adjacency matrix and a white square represents '0'. See Section II-A for a description of the adjacency matrix representation.

The minimum variance procedure described in Section II-D was used to select the edit costs for each prototype set with a minimum allowable nearest neighbor distance constraint $d_{min} = 5$. The six nearest neighbors of each protoype were used in the variance computation. Pair wise distance histograms between nearest neighbors are shown in Fig. 2. Histograms are shown for a unity cost structure along with the costs computed from the minimum variance criterion. It is clear that the minimum variance costs (labeled MinVar) result in histograms that more nearly resemble a uniform distribution, thus indicating the MinVar costs are more suitable for graph classification as previously argued. Note that in Set C, there is one pair of nearest neighbors that falls below the minimum distance constraint of $d_{min} = 5$. This is not a contradiction because the minimum distance constraint is imposed on the matchings between canonical representations as described in Section II-D. It turns out that there is in fact a closer matching between this pair than suggested by the canonical representation.

After computing the optimal edit costs, a series of sample input graphs were generated for classification. For each graph in the protoype set, a sample graph corresponding to the protoype was created by applying four random edits to the protoype. This resulted in fifteen sample graphs to classify for each prototype set. Each sample graph was matched to all fifteen protoypes within a set using both unity and MinVar costs. A sample graph is recognized as a noisy version of whichever prototype it is closest to using the graph edit distance as a metric. A 'classifier ratio'

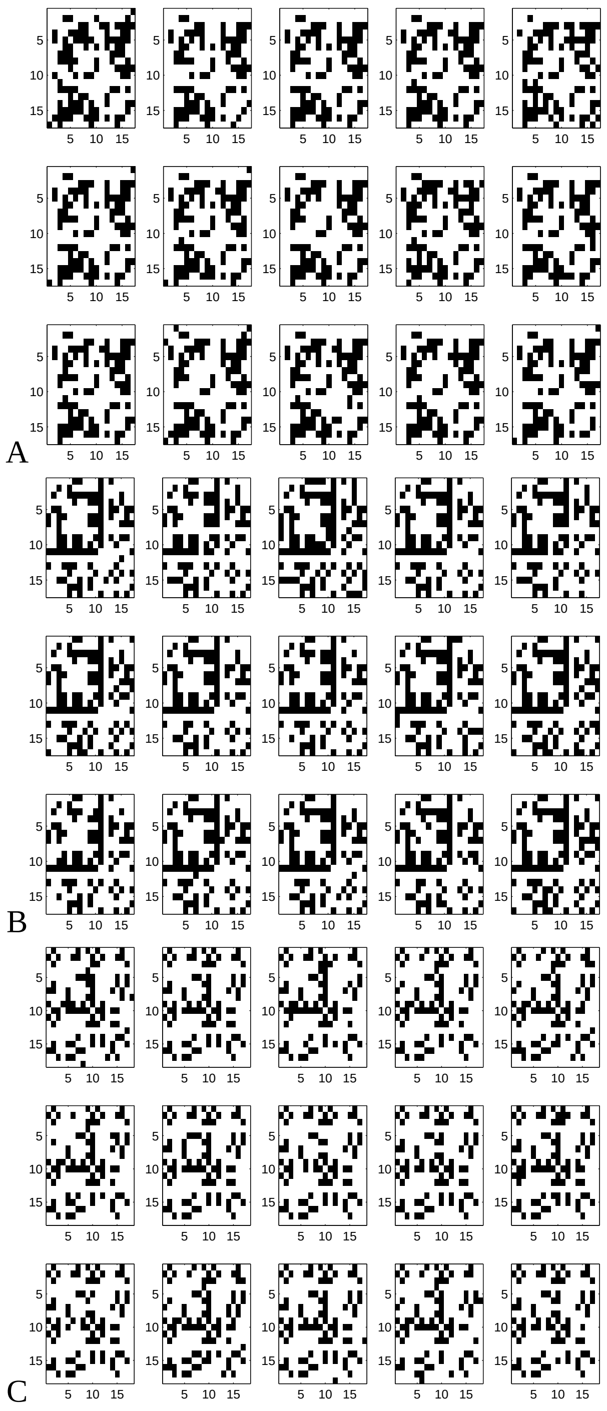
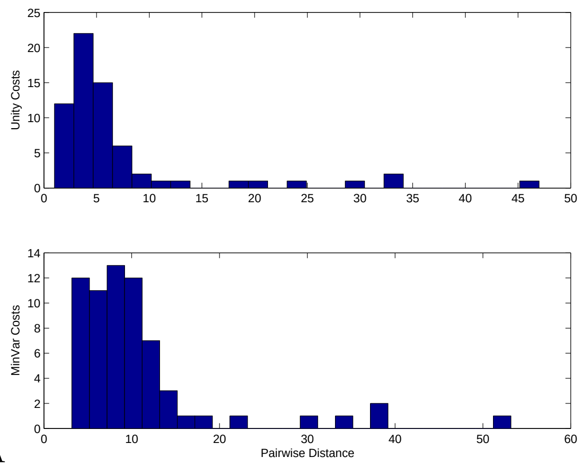
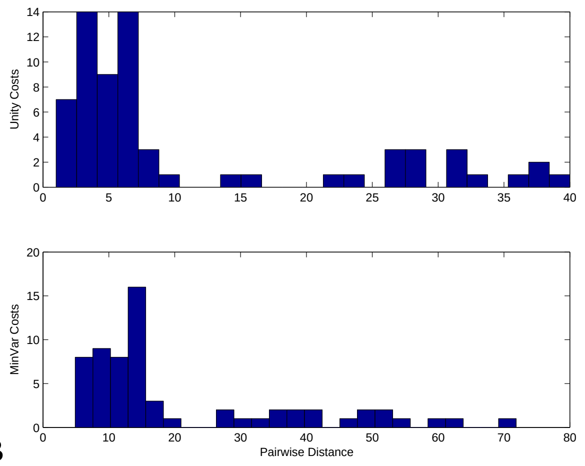


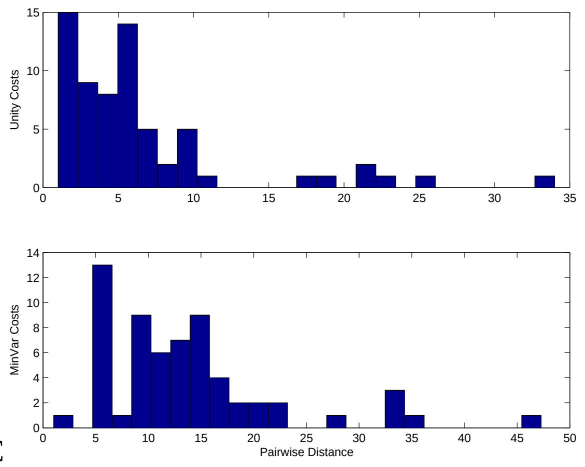
Fig. 1. The adjacency matrix representations of the three protoype sets of fifteen random graphs each. The protoypes range in size from 15 to 18 vertices with 41 to 55 edges. Here, a black square represents a ’1’ in the adjacency matrix and a white square represents ’0’. See Section II-A for a description of the adjacency matrix representation.



A



B



C

Fig. 2. Pair wise distance histograms between nearest neighbors for each of the three protoype sets. Histograms are shown for a unity cost structure (above) along with the costs computed from the minimum variance criterion (below). The minimum variance costs (labeled MinVar) result in histograms that more nearly resemble a uniform distribution, thus indicating the MinVar costs are more suitable for graph classification. Note that in Set C, there is one pair of nearest neighbors that falls below the minimum distance constraint of $d_{min} = 5$. This is not a contradiction because the minimum distance constraint is imposed on the matchings between canonical representations as described in Section II-D. It turns out that there is in fact a closer matching between this pair than suggested by the canonical representation.

	Set A	Set B	Set C
CR via Unity Costs	0.62 (±0.23)	0.70 (±0.15)	0.75 (±0.23)
CR via MinVar Costs	0.55 (±0.22)	0.45 (±0.18)	0.67 (±0.59)
Percent Decrease in Avg.	11%	35%	10%

TABLE I

CLASSIFIER RATIOS COMPUTED VIA EQ. (18) AND AVERAGED OVER THE FIFTEEN SAMPLE GRAPHS FOR EACH SET. THE STANDARD DEVIATIONS OF AVERAGED CR'S ARE GIVEN IN PARENTHESES.

(CR) as given in Eq. (18) was computed for each sample graph in order to gauge the level of ambiguity associated with the classification.

$$CR = \frac{d_*}{d_o} \tag{18}$$

Where d_* is the graph edit distance between the sample graph and the prototype from which it was generated, and d_o is the distance between the sample and the nearest incorrect prototype ('incorrect' in that the sample was not generated from this prototype). The lower CR is the less ambiguous the classification. Also, a $CR > 1$ indicates a miss-classification. The average over all fifteen samples are shown in Table I. Note that average CR values computed with the minimum variance costs are at least 10% lower than those computed with unity costs in all three sets.

IV. CONCLUSION

This paper develops a linear formulation of the graph edit distance and extends the linear programming approach to the graph matching problem introduced in [14] to find the vertex mapping and edit operations that minimize this metric. A standard graph recognition problem is presented as an application of the graph matching formalism. The edit costs are chosen using a minimum variance criterion in order to allow better discrimination among the model graphs in the database. This method is shown in simulation to more uniformly separate prototype graphs with similar structure and to more reliably classify sample input graphs than the comparable method in [14], which is a special case of the more general formalism presented here. Since cost selection and matching can both be done in polynomial time, it is a viable alternative to matching and/or classifying larger graphs based on a graph edit distance optimization when more traditional enumeration and search techniques are not practical. Also, since a logical method for selecting the parameters associated with the graph edit distance is provided in the context of the pervasive graph recognition problem, semi-arbitrary selection of tuning parameters is not an issue. We anticipate the results of this paper are applicable in any setting where a graphical model is used.

Acknowledgements

This work was partially supported by a Dept. of EECS Graduate Fellowship to the first author and by the National Science Foundation under ITR contract CCR-0325571.

References

[1] T. Pavlidis, *Structural Pattern Recognition*. New York: Springer-Verlag, 1977.

[2] L. Jianzhuang and L. Tsui, "Graph-based method for face identification from a single 2d line drawing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1106–1119, 2000.

[3] J. Llados, E. Marti, and J. Villanueva, "Symbol recognition by error-tolerant subgraph matching between region adjacency graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1137–1143, 2001.

[4] V. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. V. Dooren, "A measure of similarity between graph vertices: applications to synonym extraction and web searching," *SIAM Review*, vol. 46, no. 4, pp. 647–666, 2004.

[5] W. Tsai and K. Fu, "Error-correcting isomorphisms of attributed relational graphs for pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 757–768, 1979.

[6] H. Bunke, "Error correcting graph matching: on the influence of the underlying cost function," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 917–922, Sept. 1999.

[7] ——, "Recent developments in graph matching," *Proc. 15th Intl. Conf. on Pattern Recognition*, vol. 2, pp. 117–124, Sept. 2000.

[8] R. Wagner and M. Fischer, "The string-to-string correction problem," *Journal of the Association for Computing Machinery*, vol. 21, no. 1, pp. 168–173, 1974.

[9] M. Pavel, *Fundamentals of Pattern Recognition*. New York: Marcel Dekker, 1989.

[10] M. Neuhaus and H. Bunke, "A probabilistic approach to learning costs for graph edit distance," *Proc. 17th Intl. Conf. on Pattern Recognition*, vol. 3, pp. 389–393, 2004.

[11] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W.H. Freeman, 1979.

[12] A. Hlaoui and S. Wang, "A new algorithm for inexact graph matching," *Proc. 16th Intl. Conf. on Pattern Recognition*, vol. 4, pp. 180–183, 2002.

[13] B. Messmer and H. Bunke, "Error-correcting graph isomorphism using decision trees," *Int. Journal of Pattern Recognition and Art. Intelligence*, vol. 12, pp. 721–742, 1998.

[14] H. Almohamad and S. Duffuaa, "A linear programming approach for the weighted graph matching problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 5, pp. 522–525, May 1993.

[15] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, pp. 695–703, Sept. 1988.

[16] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 377–387, Apr. 1996.

[17] L. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 10, pp. 1367–1372, Oct. 2004.

[18] B. van Wyk and M. van Wyk, "A pocs-based graph matching algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1526–1530, Nov. 2004.

[19] B. McKay, "Practical graph isomorphism," *Congressus Numerantium*, vol. 30, pp. 45–87, 1981.

[20] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice Hall, Inc., 1982.

[21] Y. Zhang, “Solving large-scale linear programs by interior-point methods under the matlab environment,” Department of Mathematics and Statistics, University of Maryland, Baltimore, MD, Tech. Rep. TR96-01, July 1995.

[22] P. Gill, W. Murray, and M. Wright, *Practical Optimization*. London, UK: Academic Press, 1981.