Graph Matching by Relaxation of Fuzzy Assignments

Swarup Medasani, Raghu Krishnapuram, and YoungSik Choi

***Abstract*—Graphs are very powerful and widely used representational tools in computer applications. In this paper, we present a relaxation approach to (sub)graph matching based on a fuzzy assignment matrix. The algorithm has a computational complexity of $O(n^2m^2)$ where n and m are the number of nodes in the two graphs being matched, and can perform both exact and inexact matching. To illustrate the performance of the algorithm, we summarize the results obtained for more than 12 000 pairs of graphs of varying types (weighted graphs, attributed graphs, and noisy graphs). We also compare our results with those obtained using the Graduated Assignment algorithm.**

***Index Terms*—Graph isomorphism, graph matching, inexact graph matching, subgraph matching.**

I. Introduction

GRAPHS of various types have been widely used as representational tools in many applications. For example, object recognition can be accomplished by establishing a correspondence between the graph representing the object and the graphs representing the models in the database. Graphs have also been used for knowledge representation and scene description. The process of determining the correspondence between graphs is called graph matching. For real-time applications, fast and efficient graph matching is mandatory.

One of the most commonly used graph structures for modeling objects is the attributed relational graph (ARG). An ARG is a relational structure, which consists of a set of nodes connected by edges. The nodes and edges have attributes associated with them. Each node represents a component of the object being modeled, and the properties of the component are assigned as attributes to the node. The edges of an ARG represent the structural relationships between the components. A formal definition of ARGs is given in [1].

A graph $A = (V_A, E_A)$ is a collection of nodes and edges. V_A is the set of nodes and E_A is the set of edges. An edge connecting nodes u and v is denoted by (u, v), and belongs to the set E_A. If the edges and nodes are labeled, we obtain labeled graphs. The problem of comparing graphs can be classified as either exact graph matching or inexact graph matching. Graph isomorphism usually entails finding an exact match between two graphs, while subgraph isomorphism means finding an exact match between a graph and a subgraph of another graph. The exact graph-matching or graph-isomorphism problem can be stated as follows. Given two graphs $A = (V_A, E_A)$ and $B = (V_B, E_B)$, find $f\colon V_A \to V_B$ such that $(u, v) \in E_A$ iff $(f(u), f(v)) \in E_B$. The subgraph isomorphism problem can be formulated as follows. Find $V_B' \subseteq V_B$ and $E_B' \subseteq E_B$, and a mapping function $f\colon V_A \to V_B'$ such that $(u, v) \in E_A$ iff $[f(u), f(v)] \in E_B'$. In the same vein, the inexact graph matching problem can be framed as follows. If labels are assigned to nodes and edges from two sets N and E, find $V_B' \subseteq V_B, E_B' \subseteq E_B$, and a mapping function $f\colon V_A \to V_B'$ such that the number of nodes in V_A and V_B' with matching labels is $\geq l$, and the number of edges in E_B' such that $(u, v) \in E_A$ if $(f(u), f(v)) \in E_B'$ is $\geq m$. Inexact graph matching can be further generalized to the case where the labels in N and E are not crisp. In this case, inexact graph matching means comparing graphs based on the overall structure and the compatibility of node and edge labels. In this paper, we consider the exact as well as inexact subgraph matching problem.

The computational complexity of graph isomorphism is still an open question, i.e., whether it belongs to P or NP [2], [3]. However, the problem of subgraph isomorphism and inexact graph matching is known to be NP-complete [2]. There are three basic approaches to graph matching [4]. The first approach is based on group-theoretic concepts and aims at classifying the adjacency matrices into permutation groups. Unfortunately, this approach is not practical due to a large overhead. The second approach employs a state-space search. In this approach, a state-space is constructed and searched for a solution. One of the best known methods is the depth-first backtracking search [5]. The state-space search methods have a high computational complexity, i.e., $O(k^3l^2)$, where, k, l are the number of edges in the two graphs. The third approach uses concepts from nonlinear optimization. Relaxation labeling methods are the most commonly used among the nonlinear optimization methods [6]–[8]. Since these methods do not search the entire state-space, their computational complexity is quite low, namely $O(kl)$.

The graduated assignment (GA) algorithm [4] is an example of the relaxation approach. The GA technique can potentially provide good (sub)optimal solutions for problems that use a "match matrix" to denote correspondence between two groups of objects. Let graph A have n nodes and graph B have m nodes. Without loss of generality, we assume that $n \geq m$. In the graph-matching application, if the elements of the match matrix are binary, then a "1" in the ith row and jth column means that node i in graph A matches node j in graph B. In other words, the match matrix is an assignment matrix in which the sum of the elements in each column is 1, and the sum of elements in each row is ≤ 1. In GA as well as our approach, we do not restrict the elements of the matrix to be binary. In other words, we use a fuzzy assignment matrix. A new energy-minimization framework for graph isomorphism based on an equivalent maximum, clique formulation was recently proposed by [9]. Some other approaches to graph isomorphism use neural networks [10], [11], genetic algorithms [13], Lagrange relaxation [12], [14], continuous edit [15].

Manuscript received November 2, 1999; revised April 24, 2000. This work was supported by a Grant from *Korea Telecom*'s MTRL (Multimedia Technology Research Laboratory).

S. Medasani is with HRL Laboratories, LLC, Malibu, CA 90265 USA (e-mail: smedasani@hrl.com).

R. Krishnapuram was at the Colorado School of Mines, Golden, CO. He is now with IBM India Research Lab Indian Institute of Technology, New Delhi, 110016 India.

Y. Choi is with Korea Telecom, MTRL Seoul, 137-792 Korea.

Publisher Item Identifier S 1063-6706(01)01360-1.

In this paper, we present a *new* approach to graph matching by relaxation of fuzzy assignments. It is similar in principle to Gold and Rangarajan's graduated assignment algorithm [4] and Cross and Hancock's dual-step expectation maximization (EM)-based algorithm [16], but only to the extent that all these algorithms use (partial) matching degrees in each step and apply what is known as the "alternating optimization" technique. GA tries to match graphs edgewise, while our algorithm tries to match the graphs nodewise. The dual-step EM approach solves the problems of finding correspondences and projection transformation between points by maximizing the data-likelihood over the space of correspondence matches and transformation parameters. A relational consistency measure [17], [18] based on supercliques is used to assign probabilities to putative correspondences. A bipartite graph representing the most probable correspondence match is used to gate the likelihood function for obtaining the transformation parameters. In the maximization step of the EM the correspondence match is updated and is used to update the transformation parameters in turn. The highlights of our approach are: 1) the objective function is simple and easy to interpret; 2) the objective function encodes fuzzy memberships explicitly so that a discrete optimization problem is converted to a continuous one that is amenable to optimization; 3) *exact* update equations can be derived by using techniques commonly employed in fuzzy clustering and optimization theory; and 4) the algorithm is easy to implement. Our experimental results indicate that the matching accuracy of FGM is consistently higher across the board, although GA tends to be faster for sparse graphs.

The rest of the paper is organized as follows. In Section II, we introduce the fuzzy graph matching (FGM) algorithm and derive the update equations. In Section III, we discuss the compatibility measure used in FGM and present an analysis of the complexity of the algorithm. In Section IV, we present results on exact and inexact subgraph matching using FGM. Finally, in Section V, we present the conclusions.

II. FGM Algorithm

The proposed fuzzy graph matching algorithm [19] uses ideas from relaxation labeling and fuzzy set theory to solve the subgraph isomorphism problem. The algorithm can handle exact as well as inexact subgraph matching. The objective function of FGM is inspired by the assignment prototype (AP) [20], fuzzy c-means FCM [21], and GA [4] algorithms. Let A and B denote the two graphs being matched with vertex sets V_A and V_B, respectively. The complexity of FGM algorithm is $O(n^2m^2)$, where $n = |V_A|$ and $m = |V_B|$. The FGM algorithm uses a membership matrix $\mathbf{U} = [u_{ij}]$ where u_{ij} represents the relative degree to which node $i \in V_A$ matches the node $j \in V_B$, i.e., $\mathbf{U}$ is the fuzzy assignment matrix.

The objective function used for the FGM algorithm is

$$J(\mathbf{U}, \mathbf{C}) = \sum_{\substack{i=1 \\ (i,j)\neq(n+1,m+1)}}^{n+1} \sum_{j=1}^{m+1} u_{ij}^2 f(c_{ij}) + \eta \sum_{\substack{i=1 \\ (i,j)\neq(n+1,m+1)}}^{n+1} \sum_{j=1}^{m+1} u_{ij}(1-u_{ij}). \qquad (1)$$

In (1), η is a constant that controls the relative influence of the two terms in the minimization process, c_{ij} represents the absolute compatibility between nodes $i \in V_A$, $j \in V_B$ (given the fuzzy assignments $\mathbf{U}$), taking into account the attributes of the edges incident on nodes i and j and those of the neighboring nodes of i and j. In other words, $\mathbf{C} = [c_{ij}]$ is the compatibility matrix. The function $f()$ a decreasing function that converts c_{ij} to a kind of "dissimilarity." In this paper, we use $f(c_{ij}) = \exp(-\beta c_{ij})$, where β is a control parameter. This function was chosen mainly due to its simplicity. Our experiments indicate that the choice of this function is not critical. In Section III, we provide a more detailed discussion on how c_{ij}s can be chosen. As mentioned earlier, the compatibilities c_{ij} depend on $\mathbf{U}$. Similarly, the assignments $\mathbf{U}$ depend on the compatibilities $\mathbf{C}$. We update $\mathbf{U}$ and $\mathbf{C}$ in an alternating fashion, giving rise to a relaxation process. To accomplish robust matching, we introduce dummy nodes in each of the graphs being compared. Node $n+1$ in graph A and node $m+1$ in graph B represent dummy nodes. These dummy nodes [4] are similar to slack variables that are used to deal with inequality constraints in optimization problems. When a particular node in graph A does not match any of the nodes in graph B, it can be assigned to the dummy node of graph B, and vice versa. The dummy node enables us to minimize the objective function J subject to the following constraints:

$$\left.\begin{array}{ll} \sum_{j=1}^{m+1} u_{ij} = 1, & \text{for } i = 1, \ldots, n \\ \sum_{i=1}^{n+1} u_{ij} = 1, & \text{for } j = 1, \ldots, m \\ u_{ij} \geq 0 & \forall\, i \text{ and } j \end{array}\right\}. \qquad (2)$$

The first term in (1) is minimized if the matching degrees u_{ij} are high whenever the compatibilities c_{ij} are high. However, ideally we want $u_{ij} \in \{0, 1\}$. To accomplish this goal, we add the second (entropy) term in (1) which tries to push the values of u_{ij} toward either zero or one.

To minimize (1) subject to (2), we use Lagrange multipliers. The objective function then takes the form

$$\begin{aligned} J(\mathbf{U}, \mathbf{C}) = & \sum_{\substack{i=1 \\ (i,j)\neq(n+1,m+1)}}^{n+1} \sum_{j=1}^{m+1} u_{ij}^2 f(c_{ij}) \\ & + \eta \sum_{\substack{i=1 \\ (i,j)\neq(n+1,m+1)}}^{n+1} \sum_{j=1}^{m+1} u_{ij}(1-u_{ij}) \\ & - \sum_{i=1}^{n} \lambda_i \left(\sum_{j=1}^{m+1} u_{ij} - 1 \right) \\ & - \sum_{j=1}^{m} \mu_j \left(\sum_{i=1}^{n+1} u_{ij} - 1 \right) \\ & - \sum_{\substack{i=1 \\ (i,j)\neq(n+1,m+1)}}^{n+1} \sum_{j=1}^{m+1} \gamma_{ij} u_{ij}. \end{aligned} \qquad (3)$$

In (3), λ_i, μ_j, γ_{ij} are the Lagrange multipliers that handle the three sets of constraints in (2). Taking the partial derivative of J in (3) with respect to u_{pq}, we obtain

$$\frac{\partial J}{\partial u_{pq}} = \begin{cases} 2u_{pq}f(c_{pq}) + \eta(1-2u_{pq}) - \lambda_p - \mu_q - \gamma_{pq} \\ \quad \text{for } p = 1 \text{ to } n,\ q = 1 \text{ to } m \\ 2u_{pq}f(c_{pq}) + \eta(1-2u_{pq}) - \lambda_p - \gamma_{pq} \\ \quad \text{for } p = 1 \text{ to } n,\ q = m+1 \\ 2u_{pq}f(c_{pq}) + \eta(1-2u_{pq}) - \mu_q - \gamma_{pq} \\ \quad \text{for } p = n+1,\ q = 1 \text{ to } m. \end{cases} \tag{4}$$

Taking into account the objective function and the constraints of the problem, and letting $I_v = \{(i, j) | i = 1 \text{ to } n+1, j = 1 \text{ to } m+1, \text{ and } (i, j) \neq (n+1, m+1)\}$, we can write the following Karush–Kuhn–Tucker conditions for $(p, q) \in I_v$:

$$\gamma_{pq} \geq 0 \tag{5}$$
$$\frac{\partial J}{\partial u_{pq}} = 0 \tag{6}$$
$$\gamma_{pq} u_{pq} = 0. \tag{7}$$

Equations (4) and (6) can be combined into

$$u_{pq} = \begin{cases} \dfrac{\lambda_p + \mu_q + \gamma_{pq} - \eta}{2(f(c_{pq}) - \eta)} & p = 1 \text{ to } n,\ q = 1 \text{ to } m \\ \dfrac{\mu_q + \gamma_{pq} - \eta}{2(f(c_{pq}) - \eta)} & p = n+1,\ q = 1 \text{ to } m \\ \dfrac{\lambda_p + \gamma_{pq} - \eta}{2(f(c_{pq}) - \eta)} & p = 1 \text{ to } n,\ q = m+1. \end{cases} \tag{8}$$

Since (5) and (7) must hold, we have two possible cases depending on whether some of the values of γ_{pq} are zero or not.

Case 1: Here we assume that $\gamma_{pq} = 0$ for $(p, q) \in I_v$. Using the m column constraints and the n row constraints [see (2)], we obtain

$$\sum_{p=1}^{n} \frac{\lambda_p}{2D_{pq}} + \sum_{p=1}^{n+1} \frac{\mu_q}{2D_{pq}} - \sum_{p=1}^{n+1} \frac{\eta}{2D_{pq}} = 1 \qquad q = 1 \text{ to } m \tag{9}$$

and

$$\sum_{q=1}^{m+1} \frac{\lambda_p}{2D_{pq}} + \sum_{q=1}^{m} \frac{\mu_q}{2D_{pq}} - \sum_{q=1}^{m+1} \frac{\eta}{2D_{pq}} = 1, \qquad p = 1 \text{ to } n \tag{10}$$

where $D_{pq} = (f(c_{pq}) - \eta)$. Equations (9) and (10) form a consistent system of $n+m$ linear equations that can be easily solved. Equations (9) and (10) can be written as

$$\begin{bmatrix} \mathbf{K} & \vdots & \mathbf{R} & \vdots & \mathbf{x} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{R}^T & \vdots & \mathbf{L} & \vdots & \mathbf{y} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\mu} \\ 1 \end{bmatrix} = \mathbf{1} \tag{11}$$

where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n]^T$, $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_m]^T$ are the Lagrange multipliers in (3) in vector form

$$\mathbf{K} = \frac{1}{2}\,\text{diag}\left[\sum_{q=1}^{m+1} \frac{1}{D_{1q}}, \sum_{q=1}^{m+1} \frac{1}{D_{2q}}, \ldots, \sum_{q=1}^{m+1} \frac{1}{D_{nq}}\right]$$
$$\mathbf{L} = \frac{1}{2}\,\text{diag}\left[\sum_{p=1}^{n+1} \frac{1}{D_{p1}}, \sum_{p=1}^{n+1} \frac{1}{D_{p2}}, \ldots, \sum_{p=1}^{n+1} \frac{1}{D_{pm}}\right]$$
$$\mathbf{R} = [r_{ij}],\ r_{ij} = \frac{1}{2D_{ij}}, \qquad i = 1, \ldots, n,\ j = 1, \ldots, m$$

and b

$$\mathbf{x} = -\frac{\eta}{2}\left[\sum_{q=1}^{m+1} 1/D_{1q}, \ldots, \sum_{q=1}^{m+1} 1/D_{nq}\right]^T$$
$$\mathbf{y} = -\frac{\eta}{2}\left[\sum_{p=1}^{n+1} 1/D_{p1}, \ldots, \sum_{p=1}^{m+1} 1/D_{pm}\right]^T. \tag{12}$$

We can solve the $(n+m)$ linear equations for $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ as follows. From (11) we have

$$\mathbf{K}\boldsymbol{\lambda} + \mathbf{R}\boldsymbol{\mu} + \mathbf{x} = \mathbf{1} \tag{13}$$

and

$$\mathbf{R}^T\boldsymbol{\lambda} + \mathbf{L}\boldsymbol{\mu} + \mathbf{y} = \mathbf{1}. \tag{14}$$

From (13), we have

$$\boldsymbol{\lambda} = \mathbf{K}^{-1}(\mathbf{1} - \mathbf{x} - \mathbf{R}\boldsymbol{\mu}). \tag{15}$$

Substituting (15) into (14), we obtain

$$\boldsymbol{\mu} = \mathbf{P}^{-1}\left[\mathbf{1} - \mathbf{y} - \mathbf{R}^T\mathbf{K}^{-1}\mathbf{1} - \mathbf{R}^T\mathbf{K}^{-1}\mathbf{x}\right] \tag{16}$$

where $\mathbf{P} = [\mathbf{L} - \mathbf{R}^T\mathbf{K}^{-1}\mathbf{R}]$.

Case 2: In this case, we assume that $\gamma_{pq} \geq 0$ for at least some $(p, q) \in I_v$. Let us define two sets S and $\overline{S}$, where $S = \{(i, j) | u_{ij} > 0, (i, j) \in I_v\}$ and $\overline{S} = \{(i, j) | u_{ij} = 0, (i, j) \in I_v\}$. It follows from (5) and (7) that for all $(i, j) \in S$, $\gamma_{ij} = 0$ and for all $(i, j) \in \overline{S}$, $u_{ij} = 0$, and $\gamma_{ij} > 0$. Let us also define δ_{ij} as follows

$$\delta_{ij} = \begin{cases} 1 & \text{if } (i, j) \in S \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Note that δ_{ij} is not the Kronecker delta. Since some of the u_{pq}'s are zero, the m column constraints can be written as $\sum_{p=1}^{n+1} \delta_{pq} u_{pq} = 1$, for $q = 1, \ldots, m$. Similarly, the n row constraints can be written as $\sum_{q=1}^{m+1} \delta_{pq} u_{pq} = 1$, for $p = 1, \ldots, n$. Using the constraints along with (8) we obtain

$$\sum_{p=1}^{n} \frac{\delta_{pq}\lambda_p}{2D_{pq}} + \sum_{p=1}^{n+1} \frac{\delta_{pq}\mu_q}{2D_{pq}} + \sum_{p=1}^{n+1} \frac{\delta_{pq}\gamma_{pq}}{2D_{pq}} - \sum_{p=1}^{n+1} \frac{\delta_{pq}\eta}{2D_{pq}} = 1, \qquad q = 1 \text{ to } m \tag{18}$$

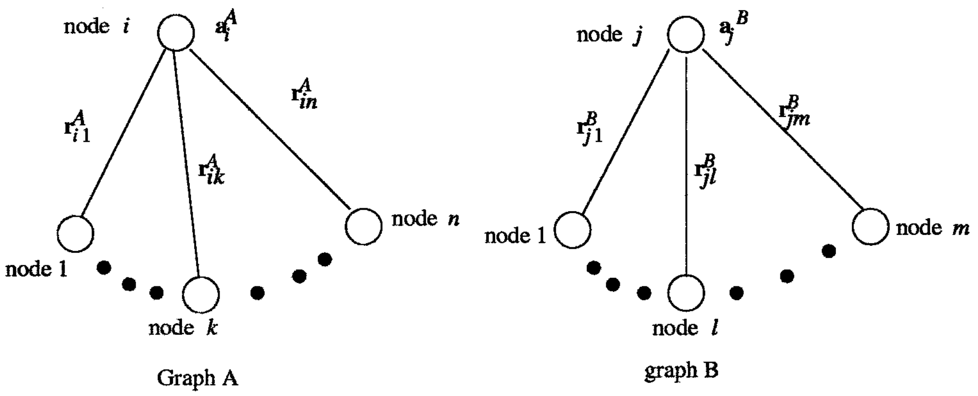

Fig. 1. Computation of compatibility between nodes.

and

$$\sum_{q=1}^{m+1} \frac{\delta_{pq}\lambda_p}{2D_{pq}} + \sum_{q=1}^{m} \frac{\delta_{pq}\mu_q}{2D_{pq}} + \sum_{q=1}^{m+1} \frac{\delta_{pq}\gamma_{pq}}{2D_{pq}} - \sum_{q=1}^{m+1} \frac{\delta_{pq}\eta}{2D_{pq}} = 1, \qquad p = 1 \text{ to } n. \tag{19}$$

Since $\delta_{pq}\gamma_{pq}$ is always zero, the third term is eliminated from (18) and (19). Equations (18) and (19) can then be written as

$$\begin{bmatrix} \mathbf{K}' & \vdots & \mathbf{R}' & \vdots & \mathbf{x}' \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ (\mathbf{R})'^T & \vdots & \mathbf{L}' & \vdots & \mathbf{y}' \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\mu} \\ 1 \end{bmatrix} = \mathbf{1} \tag{20}$$

where

$$\mathbf{K}' = \frac{1}{2}\operatorname{diag}\left[\sum_{q=1}^{m+1} \frac{\delta_{1q}}{D_{1q}}, \sum_{q=1}^{m+1} \frac{\delta_{2q}}{D_{2q}}, \ldots, \sum_{q=1}^{m+1} \frac{\delta_{nq}}{D_{nq}}\right]$$
$$\mathbf{L}' = \frac{1}{2}\operatorname{diag}\left[\sum_{p=1}^{n+1} \frac{\delta_{p1}}{D_{p1}}, \sum_{p=1}^{n+1} \frac{\delta_{p2}}{D_{p2}}, \ldots, \sum_{p=1}^{n+1} \frac{\delta_{pm}}{D_{pm}}\right]$$
$$\mathbf{R}' = [r_{ij}], \quad r_{ij} = \frac{1}{2D_{ij}}\delta_{ij}$$

and

$$\mathbf{x}' = -\frac{\eta}{2}\left[\sum_{q=1}^{m+1} \delta_{1q}/D_{1q}, \ldots, \sum_{q=1}^{m+1} \delta_{nq}/D_{nq}\right]^T$$
$$\mathbf{y}' = -\frac{\eta}{2}\left[\sum_{p=1}^{n+1} \delta_{p1}/D_{p1}, \ldots, \sum_{p=1}^{m+1} \delta_{pm}/D_{pm}\right]^T. \tag{21}$$

As in case 1, we can solve the $(n+m)$ linear equations in (20) for $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ as follows:

$$\boldsymbol{\lambda} = \mathbf{K}'^{-1}(\mathbf{1} - \mathbf{x}' - \mathbf{R}'\boldsymbol{\mu}) \tag{22}$$

and

$$\boldsymbol{\mu} = \mathbf{P}'^{-1}\left[\mathbf{1} - \mathbf{y}' - \mathbf{R}'^T\mathbf{K}'^{-1}\mathbf{1} - \mathbf{R}'^T\mathbf{K}'^{-1}\mathbf{x}'\right] \tag{23}$$

where $\mathbf{P}' = [\mathbf{L}' - \mathbf{R}'^T\mathbf{K}'^{-1}\mathbf{R}']$.

Once $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are computed, we can use c_{ij} to update memberships u_{ij} using (8). The FGM algorithm is presented below.

```
FGM Algorithm
Initialize compatibilities $\mathbf{C} = [c_{pq}]$;
REPEAT
  Compute $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ using equations (15) and (16);
  Compute memberships $\mathbf{U} = [u_{pq}]$ using (8) with $\gamma_{pq}$ set to zero;
    If any $u_{pq}$ is negative
    {
      Initialize sets $S$ and $\overline{S}$, where $S = \{(i, j) | u_{ij} > 0, (i, j) \in I_v\}$ and $\overline{S} = \{(i, j) | u_{ij} = 0, (i, j) \in I_v\}$;
      Find $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ using (22) and (23);
      For $u_{pq} \in S$ use (8) with $\gamma_{pq}$ set to zero;
      Set $u_{pq} \in \overline{S}$ to zero;
    }
    Update $\mathbf{C} = [c_{pq}]$ using (24);
UNTIL (memberships stabilize).
Use the Sinkhorn technique [22] to further crispify the $\mathbf{U}$ matrix;
```

III. Compatibility

The compatibility c_{ij} is a quantitative measure of the (absolute) degree of match between node $i \in V_A$ and node $j \in V_B$, given the current fuzzy assignment matrix $\mathbf{U}$. For example, if $c_{ij} \in [0, 1]$, then $c_{ij} = 1$ indicates complete compatibility of the nodes and $c_{ij} = 0$ indicates no compatibility. In this paper, we define the compatibility c_{ij} as

$$c_{ij} = w_{ij}^{0.5} \sum_{\substack{k=1 \\ (k,l)\neq(i,j)}}^{n+1} \sum_{l=1}^{m+1} \frac{m_{kl}m'_{kl}}{n_j^B} \qquad i = 1, \ldots, n+1, \text{ and } j = 1, \ldots, m+1 \tag{24}$$

where w_{ij} is the degree of match between (the attributes of) node $i \in V_A$ and node $j \in V_B$, $m_{kl} \in [0, 1]$ is the matching score between the edge $(i, k) \in E_A$ and edge $(j, l) \in E_B$, $\mathbf{M} = [m_{kl}]$, $\mathbf{M}' = [m'_{kl}]$ is the crisp assignment matrix closest to $\mathbf{M}$ satisfying the constraints in (2) for $i = 1, \ldots, n+1$ and $j = 1, \ldots, m+1$, and n_j^B is a normalization factor equal to the number of edges (with nonzero weights or attribute values) that are incident on node $j \in V_B$. Note that $\mathbf{M}'$ acts as a filter so that each edge in graph B that is incident on node j (except the one from the dummy node) contributes to c_{ij} only once. Also, w_{ij} is raised to the power 0.5 for enhancement purposes. Fig. 1 illustrates the notation used.

To compute $\mathbf{M}'$, we can apply the Sinkhorn technique [22], which repeatedly normalizes the rows and columns to give the solution. The complexity of this technique is $O(nm)$. However, this method sometimes takes too many iterations to converge. There are also other standard algorithms for the assignment problem that can be used here as well [23]. Rather than implement these, we used a rather straightforward method. We identify the largest element in $\mathbf{M}$, set it equal to 1, and zero out the remaining elements of the row and column corresponding to the largest element. We repeat this process m more times. This method has a somewhat higher complexity of $O(nm^2)$, but does not impose a noticeable computational burden in practice. We refer to this method as the greedy algorithm.

The degree of match m_{kl} can be defined in many ways, and the choice of the definition is application-dependent. In this paper, we deal with weighted graphs and attributed graphs. In what follows, we consider these cases separately.

In the case of weighted graphs, the nodes have no attributes, but each edge has a weight associated with it. We denote the weight associated with edge (i, k) in graph A by r_{ik}^A, where $r_{ik}^A \in [0, 1]$. Similarly, the weight associated with edge (j, l) in graph B is denoted by r_{jl}^B. Since the nodes have no attributes, we define the matching degree w_{ij} between the (attributes of) node $i \in V_A$ and node $j \in V_B$ as

$$w_{ij} = \begin{cases} 1, & \text{if } n_j^B \le n_i^A \text{ and } i \ne n+1 \text{ and } j \ne m+1 \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

where n_i^A is the number of edges with nonzero weights incident on node i of graph A, and n_j^B is the number of edges with nonzero weights incident on node j of graph B. We initialize the compatibilities as $c_{ij} = \exp\{-w_{ij}\}$, and use (24) in later iterations (for updating). For the exact matching case (i.e., when no noise is added to the edge weights of graph B), we define

$$m_{kl} = u_{kl}^{0.5} \min\left(w_{kl}, (1 - (r_{ik}^A - r_{jl}^B))^{1/4}\right) \quad \text{if } k \ne n+1 \text{ and } l \ne m+1. \tag{26}$$

For the inexact case (i.e., when noise is added to the edge weights of graph B), the weights w_{kl} are defined as in (25), and m_{kl} is given by (26). However, the compatibilities c_{ij} are computed using (24) only for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. The remaining compatibilities are not computed, but the following formulas are used to compute $f(c_{ij})$ directly. [Note that the algorithm requites only $f(c_{ij})$, and not c_{ij}.]

$$f(c_{n+1,j}) = \left[1 - \min_{1 \le k \le n} f(c_{kj})\right]^2$$

and

$$f(c_{i,m+1}) = \left[1 - \min_{1 \le k \le m} f(c_{ik})\right]^2. \tag{27}$$

The reason for using the modified updating technique, shown in (27), for the compatibilities involving dummy nodes is as follows: Equations (24) and (25) give a compatibility of 0 when one of the nodes involved is a dummy node. This forces the nodes to seek matches among the nondummy nodes. In the inexact case, since good (exact) matches are not possible, this leads to a poor performance. A better alternative is to use the dummy node to the degree that a reasonable match is not possible among the nondummy nodes. This is what (27) accomplishes. This method can also be used in the exact case, but we found that the performance is satisfactory even otherwise.

In the case of attributed graphs, node $i \in V_A$ is assumed to have n_a attributes associated with it, and each edge $(i, k) \in E_A$ has n_r attribute associated with it. Each node and edge attribute has a value in [0, 1]. The values of the n_a attributes of node i of graph A are denoted by $\mathbf{a}_i^A = (a_{i1}^A, \ldots, a_{in_a}^A)$. Similarly, the values of the n_r attributes of edge (i, k) of graph A are denoted by $\mathbf{r}_{ik}^A = (r_{ik1}^A, \ldots, r_{ikn_r}^A)$. The matching degree w_{ij} between the (attributes of) node $i \in V_A$ and node $j \in V_B$ is defined as

$$w_{ij} = \begin{cases} 1 - \max\limits_{1 \le p \le n_a} |a_{ip}^A - a_{jp}^B|, & \text{if } i \ne n+1 \text{ and} \\ & j \ne m+1 \\ 0, & \text{otherwise.} \end{cases} \tag{28}$$

Again, we initialize the compatibilities as $c_{ij} = \exp\{-w_{ij}\}$, and use (24) in later iterations. However, with attributed graphs (for both exact and inexact cases), we define

$$m_{kl} = u_{kl}^{0.5} \min\left(w_{kl}, 1 - \max_{1 \le p \le n_a} \rho_{ijkl,p}^{AB}\right) \quad \text{if } k \ne n+1 \text{ and } l \ne m+1 \tag{29}$$

where

$$\rho_{ijkl,p}^{AB} = \begin{cases} 1, & \text{if } r_{ikp}^A = 0 \text{ or } r_{jlp}^B = 0 \\ |r_{ikp}^A - r_{jlp}^B|, & \text{otherwise.} \end{cases} \tag{30}$$

The FGM algorithm requires the specification of η and β. The value of η was always chosen to be $e^{-\beta}$. The algorithm is quite insensitive to a range of β values. However, for faster convergence, the value of β was determined as follows. For the weighted graphs, we used $\beta = 3.5 + (n - 20)/10$ for the exact matching case, where n is the number of nodes. For the inexact matching case, we used $\beta = 20$ if connectivity ≤ 50, and $\beta = 40$ otherwise. In the case of attributed graphs, we used $\beta = 2.5$.

The computational complexity of the FGM algorithm can be shown to be $O(n^2m^2)$. The $\boldsymbol{\lambda}, \boldsymbol{\mu}$ computation step requires the inversion of $\mathbf{P}$ (which is an $m \times m$ matrix) and $\mathbf{K}$ (which is a diagonal matrix). In addition, two matrix vector multiplications are required [see (15) and (16)]. Thus, we have a complexity of $O(m^3 + n + m^2 + n^2)$. To compute the memberships u_{pq}, we have $O(nm)$ computations [see (8)]. If any of the memberships are negative, we have to repeat the process of $\boldsymbol{\lambda}, \boldsymbol{\mu}$ computation and then recompute $\mathbf{U}$. Computing compatibilities is the most expensive part of the FGM algorithm. Computation of the $\mathbf{M}$ matrix is $O(nm)$, finding the $\mathbf{M}'$ matrix is $O(nm)$ [$O(nm^2)$ if the greedy algorithm is used]. Thus computing c_{ij} for all i and j using (24) is $O(n^2m^2)$. Therefore, the overall complexity of the algorithm is $O(n^2m^2)$, where m is the number of nodes in the smaller graph and n is the number of nodes in the larger one. The complexity of the GA algorithm is $O(pq)$, where p is the number of edges in the first graph and q is the number of edges in the second graph. Thus, the two complexities are

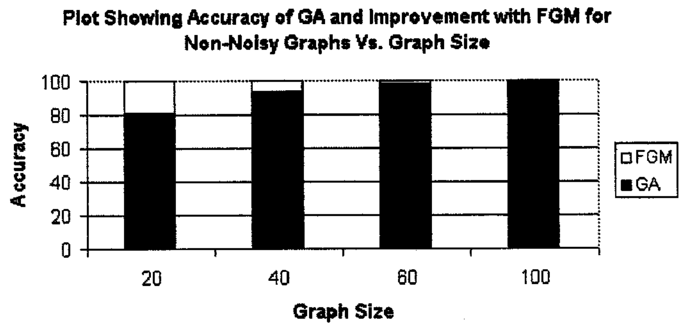

Fig. 2. Comparison of matching accuracies of FGM and GA: nonnoisy case.

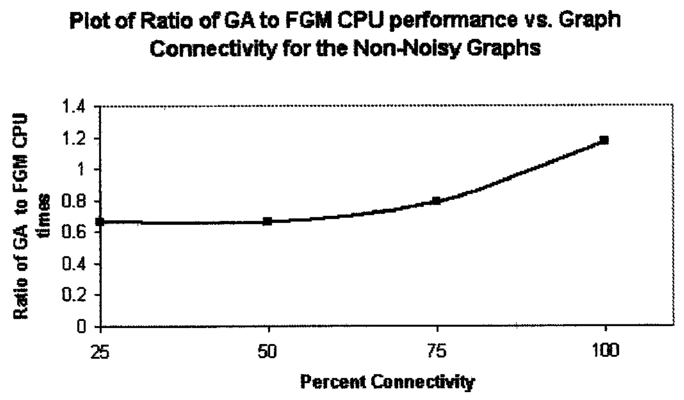

Fig. 3. Comparison of CPU times of FGM and GA in the nonnoisy case.

TABLE I
COMPARISON OF FGM AND GA FOR NONNOISY GRAPHS, $n = 20$. AVERAGE NUMBER OF MISMATCHED NODES, NUMBER OF CORRECT MATCHES, AND CPU TIME IN MIN

	cn=25%		*cn*=50%		*cn*=75%		*cn*=100%	
	FGM	GA	FGM	GA	FGM	GA	FGM	GA
del=25	0,100	4.8,87	0,100	1,98	0,100	0,100	0,100	0,100
	0.02	0.02	0.02	0.02	0.02	0.03	0.03	0.03
del=50	0,100	3.8,34	0,100	3.7,69	0,100	4.4,71	0,100	5,81
	0.01	0.01	0.03	0.02	0.02	0.02	0.02	0.02

similar. (Note that in our model, the graphs are complete, even though the strengths of relations between edges may be zero.)

IV. EXPERIMENTAL RESULTS

The FGM algorithm was tested on both weighted and attributed relational graphs of varying sizes. For both types of graphs, we tested the exact matching case as well as inexact matching case. In the inexact matching case, the weights (or attributes) of the smaller graph are perturbed by adding noise, so that exact matching is not possible. The results from more than 12 000 experiments are summarized here. The results on weighted graphs are compared with results obtained using the GA algorithm. The code for GA was obtained from the authors.

We randomly generated weighted and attributed relational graphs for our experiments. The sizes of the graphs varied from 20 nodes to 100 nodes. In particular, we used graphs with 20, 40, 60, and 100 nodes. The number of edges with nonzero weights (or nonzero attribute values) were constrained by a user-specified connectivity level (cn), which was specified as either 25%, 50%, 75%, or 100%. A connectivity of 50% implies that only 50% of all possible edges will have nonzero weights (attribute values). In the case of weighted graphs, the nodes were unlabeled and the edge strengths were chosen randomly from the [0, 1] interval. For attributed relational graphs, there were five attributes for each node and each edge, and these attributes were assigned random values in the [0,1] interval. Once the random graph of the specified size was generated,

TABLE II
COMPARISON OF FGM AND GA FOR NONNOISY GRAPHS, $n = 40$. AERAGE NUMBER OF MISMATCHED NODES, NUMBER OF CORRECT MATCHES, AND CPU TIME IN MIN

	cn=25%		*cn*=50%		*cn*=75%		*cn*=100%	
	FGM	GA	FGM	GA	FGM	GA	FGM	GA
del=25	0,100	0,100	0,100	0,100	0,100	0,100	0,100	0,100
	0.11	0.08	0.23	0.15	0.41	0.25	0.39	0.37
del=50	0,100	8.1,64	0,100	12.6,91	0,100	1,93	0,100	15,95
	0.09	0.06	0.14	0.1	0.23	0.17	0.24	0.27

TABLE III
COMPARISON OF FGM AND GA FOR NONNOISY GRAPHS, $n = 60$. AVERAGE NUMBER OF MISMATCHED NODES, NUMBER OF CORRECT MATCHES, AND CPU TIME IN MIN

	cn=25%		*cn*=50%		*cn*=75%		*cn*=100%	
	FGM	GA	FGM	GA	FGM	GA	FGM	GA
del=25	0,100	0,100	0,100	0,100	0,100	0,100	0,100	0,100
	0.57	0.26	1.28	0.75	2.9	1.45	2.2	2.6
del=50	0,100	10.1,87	0,100	20,97	0,100	1,99	0,100	0,100
	0.27	0.18	0.7	0.46	1.8	1.0	1.3	1.6

TABLE IV
COMPARISON OF FGM AND GA FOR NONNOISY GRAGHS, $n = 100$. AVERAGE NUMBER OF MISMATCHED NODES, NUMBER OF CORRECT MATCHES, AND CPU TIME IN MIN

	cn=25%		*cn*=50%		*cn*=75%		*cn*=100%	
	FGM	GA	FGM	GA	FGM	GA	FGM	GA
del=25	0,100	0,100	0,100	0,100	0,100	0,100	0,100	0,100
	4.75	1.5	11.08	5.5	26.2	12.6	19.9	22.7
del=50	0,100	21.6,92	0,100	0,100	0,100	0,100	0,100	0,100
	2.2	0.94	6.7	3.2	8.8	8.0	8.1	14.6

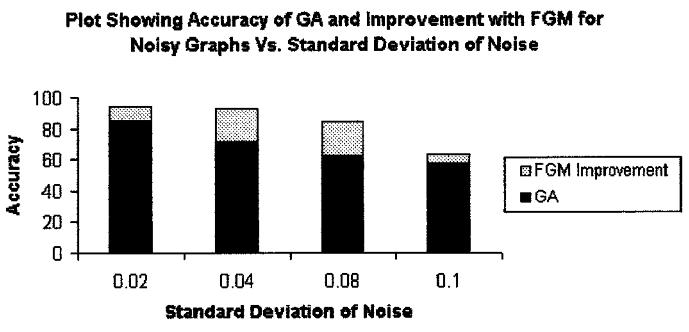

Fig. 4. Comparison of matching accuracies of FGM and GA: noisy case.

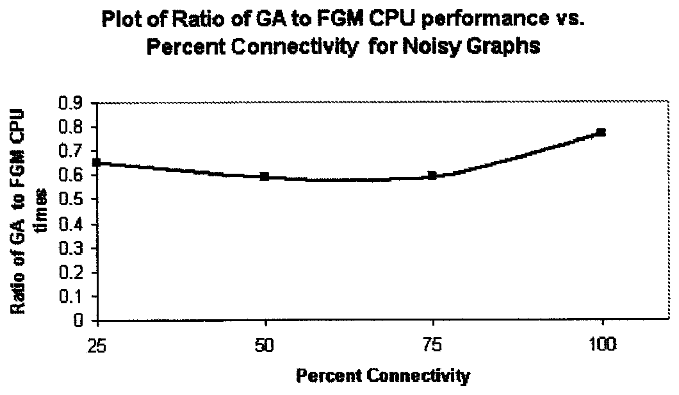

Fig. 5. Comparison of CPU times of FGM and GA in the noisy case.

TABLE V
COMPARISON OF FGM AND GA FOR NOISY GRAPHS, $n = 40, \sigma = 0.02$. AVERAGE NUMBER OF MISMATCHED NODES, NUMBER OF CORRECT MATCHES, AND CPU TIME IN MIN

	cn=25%		*cn*=50%		*cn*=75%		*cn*=100%	
	FGM	GA	FGM	GA	FGM	GA	FGM	GA
del=25	1,94	5.8,92	0,100	0,100	0,100	0,100	0,100	0,100
	0.12	0.08	0.39	0.15	0.6	0.25	0.53	0.38
del=50	1.4,60	5.2,40	0,100	5.3,78	0,100	4.8,74	0,100	4.3,90
	0.08	0.07	0.16	0.1	0.25	0.17	0.22	0.26

TABLE VI
COMPARISON OF FGM AND GA FOR NOISY GRAPHS, $n = 40, \sigma = 0.04$. AVERAGE NUMBER OF MISMATCHED NODES, NUMBER OF CORRECT MATCHES, AND CPU TIME IN MIN

	cn=25%		*cn*=50%		*cn*=75%		*cn*=100%	
	FGM	GA	FGM	GA	FGM	GA	FGM	GA
del=25	0,100	6.2,70	0,100	14.5,96	0,100	14.5,98	0,100	0,100
	0.12	0.09	0.24	0.17	0.38	0.28	0.54	0.43
del=50	1.5,52	6.4,20	1.4,76	4.9,44	0,100	6.7,50	0,100	7.2,88
	0.09	0.07	0.17	0.11	0.25	0.19	0.35	0.27

TABLE VII
COMPARISON OF FGM AND GA FOR NOISY GRAPHS, $n = 40, \sigma = 0.08$. AVERAGE NUMBER OF MISMATCHED NODES, NUMBER OF CORRECT MATCHES, AND CPU TIME IN MIN

	cn=25%		*cn*=50%		*cn*=75%		*cn*=100%	
	FGM	GA	FGM	GA	FGM	GA	FGM	GA
del=25	1,98	2.5,31	1,98	12.3,96	0,100	14.5,98	0,100	0,100
	0.14	0.08	0.29	0.16	0.62	0.26	0.87	0.38
del=50	3.2, 54	6.4, 24	3.3, 62	6.0, 50	5.4, 62	5.1, 40	0, 100	4.1, 52
	0.08	0.06	0.12	0.12	0.25	0.19	0.43	0.31

TABLE VIII
COMPARISON OF FGM AND GA FOR NOISY GRAPHS, $n = 40, \sigma = 0.1$. AVERAGE NUMBER OF MISMATCHED NODES, NUMBER OF CORRECT MATCHES, AND CPU TIME IN MIN

	cn=25%		*cn*=50%		*cn*=75%		*cn*=100%	
	FGM	GA	FGM	GA	FGM	GA	FGM	GA
del=25	3.5,66	7.5,64	0,100	7.8,96	0,100	13.8,94	0,100	14.5,98
	0.29	0.08	0.59	0.17	0.76	0.29	1.08	0.46
del=50	1.5, 24	6.4, 14	1.6, 34	5.2, 30	2.5, 40	4.8, 22	4.3, 42	6.1, 40
	0.11	0.06	0.2	0.11	0.35	0.19	0.29	0.31

the nodes of the graph were permuted. The permuted graph was then modified by deleting randomly selected nodes. The number of nodes to be deleted was specified in terms of a deletion percent (del), which was either 25% or 50%. The true-matching assignments between the original graph and the modified subgraph were also recorded and used as the ground truth when comparing the assignments obtained from FGM and GA algorithms. This method of scoring only compares the modified graph with the original, and may ignore better matches that result from the modification. This is particularly true in the inexact matching case, where the noise added to the weights (attribute values) may cause the best match to be different from the recorded ground truth.

In our first experiment, we selected graphs of sizes 20, 40, 60, and 100 nodes. For each graph size, we used four connectivity levels and two deletion levels, which gives us eight cases. For each case, we generated 100 random graph-pairs. The FGM and GA algorithms were run on the 100 graph-pairs and the results were averaged. We recorded the number of times the graph-pairs were matched perfectly by FGM and GA, as well as the average number of nodes that were mismatched. The average was computed over only those cases in which perfect matching did not occur. The results are presented in Tables I–IV. Each entry in the tables shows the average number of mismatched nodes (in the cases where mismatching occurred), total number of correct matches, and the CPU time in minutes. From the results, we see that FGM results are 100% accurate while the GA fails in several cases. As a summary of the tables, Fig. 2 shows the improvement in accuracy (i.e.,

TABLE IX
FGM RESULTS ON EXACT MATCHING OF ATTRIBUTED GRAPHS. AVERAGE NUMBER OF MISMATCHED NODES, NUMBER OF CORRECT MATCHES, AND CPU TIME IN MIN

	n=20				n=40			
	cn=25%	cn=50%	cn=75%	cn=100%	cn=25%	cn=50%	cn=75%	cn=100%
del=	0,100	0,100	0,100	0,100	0,100	0,100	0,100	0,100
25%	0.01	0.02	0.02	0.02	0.11	0.23	0.36	0.49
del=	0,100	0,100	0,100	0,100	0,100	0,100	0,100	0,100
50%	0.01	0.01	0.01	0.01	0.05	0.09	0.15	0.2

TABLE X
FGM RESULTS ON INEXACT MATCHING OF ATTRIBUTED GRAPHS AT NOISE LEVEL $\sigma = 0.04$. AVERAGE NUMBER OF MISMATCHED NODES, NUMBER OF CORRECT MATCHES, AND CPU TIME IN MIN

	n=20				n=40			
	cn=25%	cn=50%	cn=75%	cn=100%	cn=25%	cn=50%	cn=75%	cn=100%
del=	1,98	0,100	0,100	0,100	0,100	0,100	0,100	0,100
25%	0.01	0.02	0.02	0.03	0.11	0.24	0.28	0.39
del=	0,100	0,100	0,100	0,100	0,100	0,100	0,100	0,100
50%	0.005	0.01	0.01	0.012	0.06	0.09	0.12	0.16

percentage of correct matches) achieved by FGM over that of GA for the nonnoisy case. Fig. 3 depicts a plot of the ratio of CPU time taken by GA to that of FGM as a function of percent connectivity. It can be seen from these figures that for the nonnoisy case, the improvement provided by FGM in matching accuracy is significant for graphs of relatively smaller size. Although the GA algorithm is somewhat quicker than the FGM algorithm in many cases, FGM has an advantage for large and highly connected graphs. The experiments were conducted on a Pentium II 450-MHz workstation and no attempt was made to optimize the code of the FGM algorithm. The GA algorithm is essentially edge-based, and is quite efficient for sparse graphs since it uses special data structures.

The second set of experiments deal with inexact weighted graph matching. The random graph-pairs from the previous experiment were used again. The subgraphs from each graph-pair were perturbed by adding uniform noise. Since this experiment intends to find the efficiency of FGM for inexact matching, we fixed the size of the larger graph to 40 nodes and varied the standard deviation of the noise added to the smaller graph. In particular, we tried $\sigma = 0.02$, 0.04, 0.08, and 0.1. We averaged the results of inexact matching over 100 graph-pairs. The results are presented in Tables V–VIII. From the results, we can see that FGM matching accuracy is always better than or similar to that of GA. The percent of nodes mis-matched is also smaller for FGM. However GA is again somewhat faster than FGM in finding the solution. These results also show that as the connectivity increases, the influence of noise diminishes. This is because matching becomes easier with more constraints. To summarize the tables, Fig. 4 shows the improvement in matching accuracy provided by FGM over GA as a function of standard deviation of noise. It can be seen that the improvement is significant for intermediate-level noise. Fig. 5 shows that the advantage of GA in terms of CPU time diminishes with higher connectivity even in the noisy case.3

The next two sets of experiments deal with attributed relational graphs. Each node and edge in the attributed relational graph is represented by five weights in the interval [0, 1]. The results on attributed graphs of sizes 20 and 40 nodes are given in Table IX. In both cases, the FGM results are 100% accurate and are quite fast. The attributed graphs generated in the previous step were then perturbed by adding noise. The noise was added to the subgraphs, and standard deviations of 0.04, 0.1, and 0.2 were used. The noise added here was higher than in the weighted graph case. The results on the 20- and 40-node graphs for the noise level $\sigma = 0.04$ are presented in Table X. Tables XI and XII contain the results for the noise levels $\sigma = 0.1$ and $\sigma = 0.2$. From the results, we can see that FGM is quite robust and insensitive to increased noise levels. The matching results are good. The version of the GA code we had was not able to handle attributed graphs, and so no comparison was possible.

V. SUMMARY AND CONCLUSION

In this paper, we present a fuzzy approach to exact and inexact (sub)graph matching. As in [4], we use a type of relaxation technique. Fuzzy assignments are used in each iteration to avoid premature convergence to local minima. We update the

TABLE XI
FGM RESULTS ON INEXACT MATCHING OF ATTRIBUTED GRAPHS AT NOISE LEVEL $\sigma = 0.1$. AVERAGE NUMBER OF MISMATCHED NODES, NUMBER OF CORRECT MATCHES, AND CPU TIME IN MIN

	n=20				n=40			
	cn=25%	cn=50%	cn=75%	cn=100%	cn=25%	cn=50%	cn=75%	cn=100%
del=	1,97	0,100	0,100	0,100	0,100	0,100	0,100	0,100
25%	0.008	0.014	0.02	0.024	0.11	0.24	0.35	0.47
del=	1,98	0,100	0,100	0,100	1,97	0,100	0,100	0,100
50%	0.004	0.007	0.008	0.01	0.05	0.096	0.24	0.19

TABLE XII
FGM RESULTS ON INEXACT MATCHING OF ATTRIBUTED GRAPHS AT NOISE LEVEL $\sigma = 0.2$. AVERAGE NUMBER OF MISMATCHED NODES, NUMBER OF CORRECT MATCHES, AND CPU TIME IN MIN

	n=20				n=40			
	cn=25%	cn=50%	cn=75%	cn=100%	cn=25%	cn=50%	cn=75%	cn=100%
del=	1,97	0,100	0,100	0,100	0,100	0,100	0,100	0,100
25%	0.008	0.014	0.02	0.03	0.1	0.19	0.28	0.39
del=	1,92	0,100	1,99	1,98	1,96	0,100	1,99	1,96
50%	0.004	0.006	0.009	0.012	0.05	0.08	0.12	0.16

fuzzy assignments based on fuzzy compatibilities, giving rise to a relaxation procedure. We incorporate robustness into the matching process by adding dummy nodes to the two graphs being matched. The computational complexity of the algorithm is $O(n^2m^2)$.

The GA algorithm solves several assignment problems using a continuation method to obtain a double stochastic match matrix after each iteration. The control parameter (β) of the continuation method is slowly increased to obtain a crisp solution for the match matrix. In the case of FGM, the use of a fuzzy objective function eliminates the use of a continuation method. FGM first uses a greedy method to compute compatibilities between nodes, these compatibilities are then used to solve the assignment problem. These two steps are iterated until convergence. The use of the greedy method and the crisp match matrix ensure that consistency constraints are satisfied. In the GA algorithm, the authors use a first-order expansion to simplify the analysis and derive the update equations. No such approximation needs to be made in the case of FGM. Nevertheless, as in the case of GA, convergence to a global optimum is not guaranteed in the case of FGM. Thus, the only possible way to compare them is through experiments. Our experiments unequivocally indicate that the matching accuracy of FGM is consistently higher for graphs of various sizes and connectivity levels. We also believe that FGM can be speeded up considerably. Rather than implement some of the routines required in the optimization process, we borrowed third-party code for quick testing. Thus our code is not at all optimized. Changes to the code are difficult now due to logistic reasons.

The low computational complexity of the FGM algorithm makes it suitable for several real-time computer applications. FGM is easy to implement, and has a matching accuracy that is higher than the GA algorithm. We are currently looking into ways of improving the speed of the algorithm, and applying it to content-based retrieval of images [24], [25]. Another attractive feature of FGM is that when the algorithm converges, the value of the objective function can be used as a measure of similarity between the two graphs. This is useful in many applications, such as clustering.

ACKNOWLEDGMENT

This work was performed while one of the authors (R. Krishnapuram) was with the Colorado School of Mines, Golden, CO USA.

REFERENCES

[1] M. A. Eshera and K. S. Fu, “An image understanding system using attributed symbolic representation and inexact graph-matching,” *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 8, pp. 604–619, Sept. 1986.
[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.
[3] M. A. Abdulrahim, “Parallel algorithms for labeled graph matching,” Ph.D. dissertation, Colorado School of Mines, Golden, CO, 1998.
[4] S. Gold and A. Rangarajan, “A graduated assignment algorithm for graph matching,” *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 18, pp. 377–387, Apr. 1996.
[5] D. G. Corneil and C. C. Gotlieb, “An efficient algorithm for graph isomorphism,” *J. ACM*, vol. 17, pp. 51–64, 1970.
[6] A. Rosenfeld, R. Hummer, and S. Zucker, “Scene labeling by relaxation operations,” *IEEE Trans. Syst., Man, Cybern.*, vol. 6, pp. 420–433, June 1976.

[7] S. Peleg, “A new probabilistic relaxation scheme,” *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 2, pp. 362–369, July 1980.

[8] O. Faugeras and M. Berthoud, “Improving consistency and reducing ambiguity in stochastic labeling: An optimization approach,” *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 4, pp. 412–424, July 1981.

[9] M. Pelillo, “Replicator equations, maximal cliques, and graph isomorphism,” *Neural Computat.*, vol. 11, no. 8, pp. 1933–1955, 1999.

[10] T. W. Chen and W. C. Lin, “A neural network approach to csg-based 3-d object recognition,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, pp. 719–725, July 1994.

[11] P. Suganthan, E. Teoh, and D. Mital, “Pattern recognition by graph matching using the potts mft neural networks,” *Pattern Recog.*, vol. 28, pp. 997–1009, 1995.

[12] M. Krcmar and A. Dhawan, “Application of gentic algorithms for graph matching,” in *Proc. IEEE Int. Conf. Neural Networks (ICNN)*: IEEE Press, 1994, pp. 3872–3876.

[13] A. Rangarajan and E. Mjolsness, “A Lagrangian relaxation network for graph matching,” *IEEE Trans. Neural Networks*, vol. 7, pp. 1365–1381, 1996.

[14] A. Rangarajan, S. Gold, and E. Mjolsness, “A Lagrangian relaxation network for graph matching,” *IEEE Trans. Neural Networks*, vol. 7, pp. 1041–1060, 1996.

[15] A. Finch, R. Wilson, and E. R. Hancock, “An energy function and continuous edit process for graph matching,” *Neural Computat.*, vol. 10, pp. 1873–1894, 1998.

[16] A. Cross and E. R. Hancock, “Graph matching with a dual-step em algorithm,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 11, pp. 1236–1253, 1998.

[17] R. Wilson and E. R. Hancock, “Structural matching by discrete relaxation,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 634–648, 1997.

[18] A. Finch, R. Wilson, and E. R. Hancock, “Symbolic graph matching with the em algorithm,” *Pattern Recog.*, vol. 31, pp. 1777–1790, 1998.

[19] R. Krishnapuram and S. Medasani, “A fuzzy approach to graph matching,” in *Proc. Int. Fuzzy Syst. Assoc. Congress*, Taipei, Taiwan, August 1999, pp. 1029–1033.

[20] M. P. Windham, “Numerical classification of proximity data with assignment measure,” *J. Classificat.*, vol. 2, pp. 157–172, 1985.

[21] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press, 1981.

[22] R. Sinkhorn, “A relationship between arbitrary positive matrices and doubly stochastic matrices,” *Ann. Math. Statistics*, vol. 35, pp. 876–879, 1964.

[23] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[24] S. Medasani and R. Krishnapuram, “A fuzzy approach to content-based image retrieval,” in *Proc. IEEE Conf. Fuzzy Syst.*, Seoul, Korea, Aug. 1999, pp. 1251–1257.

[25] ——, “Content-based image retrieval using fuzzy attributed relational graphs,” in *Proc. IEEE Conf. Multimedia Syst.*, Florence, Italy, June 1999, pp. 964–968.