Subgraph Error-Correcting Isomorphisms for Syntactic Pattern Recognition

WEN-HSIANG TSAI, MEMBER, IEEE, AND KING-SUN FU, FELLOW, IEEE

Abstract—**The structure-preserved error-correcting graph isomorphism proposed by Tsai and Fu [1] for matching patterns represented by attributed relational graphs is extended to the case of subgraphs. The resulting subgraph error-correcting isomorphism, which includes the structure-preserved error-correcting graph isomorphism as a special case, is useful for recognizing partially viewed or structurally distorted patterns. After formulating a subgraph error-correcting isomorphism as a state-space tree-search problem, heuristic information useful for speeding up the search is suggested and an ordered-search algorithm is proposed for finding an optimal subgraph error-correcting isomorphism.**

I. Introduction

THIS PAPER will try to extend the structure-preserved error correcting isomorphism of attributed relational graphs proposed by Tsai and Fu [1] for pattern analysis. Their work is first briefly reviewed, followed by a discussion on the limitation of the structure-preserved error-correcting graph isomorphism to more practical applications which motivates this study of subgraph error-correcting isomorphisms, to be proposed in the following sections.

Attributed relational graphs are defined [1] for representing both structural and semantic information contained in given patterns. In an attributed relational graph, the underlying graph consisting of unlabeled nodes and branches represents the global structure of the pattern. The labels of the nodes and the branches specify the local structures and the local properties of the primitives and relations in the pattern. Each label of a node or a branch is a pair (s, x) where s, called a *syntactic symbol*, is used to denote the structure of the primitive or the relation represented by the node or the branch, and x, called a *semantic vector*, consists of a set of numerical and/or logical attributes of the primitive or the relation. Possible primitive structures may be a line segment or a small square region with the length of the segment or the area of the region as the attributes, respectively. Possible relation structures may be "above," "left," etc., with the distance between the two related primitives as the relation attribute.

A pattern deformational model is also proposed [1] to model a kind of so-called structure-preserved graph deformations, which transform *pure* or noise-free patterns into *observed* or deformed patterns by locally corrupting only the structures and the properties of the primitives and the relations without changing the global structures of both the pure and the observed patterns. In terms of relational-graphic terminologies, a structure-preserved deformation only changes the labels of some nodes or branches in a relational graph, but does not delete any nodes or branches. Based on this structure-preserved deformational model, the probability or density value of an observed pattern deformed from a pure pattern is then computed in terms of the deformation probability or density values of the primitives, and relations in the observed pattern under some assumptions on the interdependency of primitive and relation deformations.

To recognize an observed pattern which is transformed from a pure pattern through a structure-preserved deformation, one approach is to match the relational graph of the former with that of the latter. This is a graph isomorphism problem. But conventional graph isomorphism procedures are *discrete* and *exact* in nature [4]–[7], and can only be used for matching two symbolically identical graphs without attributes. For matching attributed relational graphs whose labels of nodes and branches may be different, structure-preserved error-correcting graph isomorphisms are then defined [1]. The pattern deformation probability or density value of an observed pattern from a pure pattern is used naturally as the likelihood of a structure-preserved error-correcting isomorphism between the two patterns, which specifies the goodness of the matching defined by the error-correcting isomorphism. Since there may exist more than one isomorphism between two relational graphs, the structure-preserved error-correcting isomorphism with the maximum likelihood is suggested as the desired matching of the two graphs.

To determine the maximum-likelihood structure-preserved error-correcting isomorphism between two relational graphs, the graph isomorphism problem can be formulated as a state-space tree-search problem [3], [4]. Since the negative logarithms of the primitive and relation deformation probability or density values can be used as the path costs for the search of an optimal path in the state-space tree, blind search methods can be avoided, and a uniform-cost search method can be applied [3]. Tsai and Fu [1] furthermore propose a more efficient ordered-search algorithm for finding the maximum-likelihood structure-preserved error-correcting graph isomorphism after suggesting several ways of using heuristic information to speed

Manuscript received June 4, 1981; revised August 26, 1982. This work was supported in part by ONR Contract N00014-79-C-0574 and NSF Grant ECS 78-16970.

W. H. Tsai is with the Institute of Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China.

K. Fu is with the School of Electrical Engineering, Purdue University, Electrical Engineering Building, West Lafayette, IN 47907.

0018-9472/83/0100-0048$01.00

up the search procedure. Finally they propose a decision rule for classifying unknown patterns, based on the maximum likelihood computed by the ordered-search algorithm after the maximum-likelihood error-correcting isomorphism is determined.

From the previous review of Tsai and Fu's work, we see a limitation of their approach. That is the *structure-preserved* property of the proposed error-correcting isomorphism which is based on the structure-preserved pattern deformational model. The resulting pattern analysis capability, though powerful enough for a variety of practical problems, is insufficient for more general applications. For example when the observed patterns are just partial views or distorted parts of some pure patterns, the resulting attributed relational graphs of the former will just be the subgraphs of those of the latter. In such cases subgraph isomorphisms are needed for recognizing the observed patterns. On the other hand subgraph isomorphisms also find quite a few applications other than pattern analysis [8]–[11].

Inexact subgraph matching of weighted relational graphs has been investigated by Shapiro and Haralick [12] and Kitchen [13] using the relaxation method, and by Ghahraman *et al.* [14] in terms of the subgraphs of the Cartesian graph product. We propose subgraph error-correcting isomorphisms of probabilistic attributed relational graphs as an extension of structure-preserved error-correcting graph isomorphisms, with the latter being the special cases of the former. The extension is based on a modification of the structure-preserved graph deformational model to include the deletions of primitives or relations in an observed pattern. The state-space formulation for the search of structure-preserved graph isomorphisms is also modified for the search of subgraph isomorphisms. Corresponding heuristic information contained in the graphs is extracted for speeding up the search. An ordered-search algorithm for finding maximum-likelihood subgraph error-correcting isomorphisms is finally proposed, accompanied by an illustrative example.

II. A Subgraph Deformational Model for Attributed Relational Graphs

In this section the definition of attributed relational graphs is reviewed [1], [2]. Subgraph deformations are then defined as an extension of structure-preserved graph deformations. After decomposing a pattern deformation into primitive and relation deformations, primitive and relation deletions and substitutions are then investigated and their occurrence probability or density values are defined. Pattern deformation probability or density values are finally computed in terms of these primitive and relation deformation probability or density values.

A. Definitions of Attributed Relational Graphs and Subgraph Deformations

Definition 1: An attributed relational graph (ARG) over $V_N \cup V_B$ is a 4-tuple $\omega = (N, B, \mu, \epsilon)$ where

N	is a finite nonempty set of elements called nodes;
$B \subset N \times N$	is a set of distinct ordered pairs of distinct elements in N called branches;
V_N	is a finite nonempty set of node labels (as primitive descriptions), each of which is denoted as a pair (s, x) as explained in the Introduction;
V_B	is a set of branch labels (as relation descriptions), each of which is also denoted as a pair (s, x) as explained in the Introduction;
$\mu: N \to V_N$	is a function called node interpreter;
$\epsilon: B \to V_B$	is a function called branch interpreter;
$G_\omega = (N, B)$,	which denotes the unlabelled graph obtained from ω by deleting all the node and branch labels, is called the underlying graph of ω.

Definition 2: Let $\omega = (N, B, \mu, \epsilon)$ over $V_N \cup V_B$ be the ARG for a given pure pattern, called a pure ARG, and $\omega' = (N', B', \mu', \epsilon')$ over $V_{N'} \cup V_{B'}$ be the ARG for one of the observed patterns deformed from ω, called an observed ARG. When $N' \subset N$ and $B' \subset B$, i.e., when $G_{\omega'}$ is just a subgraph of G_ω, then we say that there exists a subgraph deformation from ω into ω'.

Note that a subgraph deformation includes a *structure-preserved graph deformation* [1] as a special case, which occurs when $B' = B$ and $N' = N$. In a subgraph deformation, not only the underlying global graphic structure of ω is affected because $N' \subset N$ and $B' \subset B$, but the labels of ω, which specify both the local structures and the local properties contained in ω, are also subject to changes because in general $\mu' \neq \mu$ and $\epsilon' \neq \epsilon$. More specifically if node α' in N' corresponds to node α in N, then it is possible that the label $\mu'(\alpha')$ is not identical to the label $\mu(\alpha)$. Similarly for the corresponding branches γ' in B' and γ in B, it may be true that $\epsilon'(\gamma') \neq \epsilon(\gamma)$. On the other hand since $N' \subset N$ and $B' \subset B$, some nodes and branches in ω may be deleted when ω is transformed into ω'. This means that some primitives or relations in an observed pattern may be missing when the observed pattern is compared with its corresponding pure pattern. In the rest of this paper, for discussion convenience, the notations ω and ω', previously used to denote ARG's, will also be used to denote the patterns represented by the ARG's. Similarly the label of a node or a branch will also be used to denote the primitive or the relation itself represented by the node or the branch, respectively. For simplicity we will also call either a primitive or a relation a *terminal.*

B. Primitive Deformations

To investigate the details of a pattern deformation, Tsai and Fu [1], [2] decompose a pattern deformation into a set of primitive and relation deformations. In this paper we discuss primitive and relation deformations from the viewpoint of subgraph deformations. Primitive deformations

are investigated in this section, followed by relation deformations in the next section.

Let $a = (s, x)$ be a pure primitive in a pure pattern ω, and let the deformation from ω into one of its observed version ω' be a subgraph deformation. Then a may be deleted or it may be transformed into an observed primitive $c = (t, z)$, when ω is deformed into ω'. Either of these two cases will be called a *primitive deformation*. When a is deleted, we denote

$$a \xrightarrow[\text{deletion}]{r_D(\lambda|a)} \lambda$$

where λ, a null symbol, is used to represent the deletion of a terminal, and $r_D(\lambda|a) \neq 0$ is the probability for a to be deleted, called the *deletion* probability of a. For simplicity $r_D(\lambda|a)$ is also denoted as $r_D(a)$. For the other case, where a is substituted by c, we denote

$$a \xrightarrow[\text{substitution}]{r_S(c|a)} c$$

where $r_S(c|a) \neq 0$ is the probability or density value for a to be transformed into c, called the *substitution probability* or *density* of a by c. Either $r_D(\lambda|a)$ or $r_S(c|a)$ will be called the *deformation probability* or density of a. Let $D_S(a)$ denote the set of all possible observed versions of a. By convention [15], [16] a pure primitive is assumed to be a possible observed version of itself, so $a \in D_S(a)$ and $r_S(a|a)$ is not always equal to one. Also, we use $r_S(a)$ to denote $\sum_{c \in D_S(a)} r_S(c|a)$ when z in all $c = (t, z)$ is discrete, or $\int_{c \in D_S(a)} r_S(c|a)$ when z in all $c = (t, z)$ is continuous. Then we have the equality

$$r_D(a) + r_S(a) = 1.$$

Notice that in the above discussion we have made implicitly an assumption that each primitive is deformed independently of any other terminals in ω.

On the other hand, following Tsai and Fu [1], [2], we can further decompose a substitution of $a = (s, x)$ by $c = (t, z)$ into two steps as follows:

$$a = (s, x) \xrightarrow[\substack{\text{syntactic}\\ \text{substitution}}]{p(t|s)} b = (t, y) \xrightarrow[\substack{\text{semantic}\\ \text{substitution}}]{q(z|t, s)} c = (t, z),$$

↑	↑	↑
pure primitive	semi-pure primitive	observed primitive

where the first step, a *syntactic substitution*, transforms the syntactic symbol s of a into another symbol t with discrete probability $p(t|s)$, called *syntactic substitution probability*, with y being a representative semantic vector for t. The second step, a *semantic substitution*, further transforms y into an observed semantic vector z with probability or density $q(z|t, s)$, called *semantic substitution probability* or *density*. Therefore the substitution probability or density of a by c can be computed as

$$r_S(c|a) = p(t|s)q(z|t, s).$$

Note that $\int_z q(z|t, s)\, dz = 1$ when all z are continuous, or $\sum_z q(z|t, s) = 1$ when all z are discrete, and that $r_S(a) = \sum_t p(t|s)$. (For illustrative examples see [16].)

C. Relation Deformations

As to relation deformations the situations are much more complicated, because a relation deformation usually will be affected by the deformations induced on the two primitives at the ends of the relation. For example a relation obviously should be deleted if either or both of its two end primitives are deleted. Other cases exist and will be discussed subsequently.

Let $e = (u, x)$ be a pure relation between two pure primitives a and b all in a pure pattern ω. As in [1], [2] we assume in the following discussions that e is deformed independently of any other terminals (primitives or relations) except its two end primitives a and b; it deforms according to how a, b are deformed in ω', an observed pattern of ω. Totally, three different kinds of deformations induced on e can be identified, each of which will be called a *relation deformation*:

1) e is deleted in ω' due to the deletion of either or both of its two end primitives a and b. By taking $a \xrightarrow[\text{deletion}]{} \lambda$ or/and $b \xrightarrow[\text{deletion}]{} \lambda$ as a condition, the conditional probability for e to be deleted obviously is one because no relation will ever exist when either or both of the end primitives are missing. For this case, we denote

$$e \xrightarrow[\text{deletion}]{1.0} \lambda.$$

2) e is deleted in ω' while its two end primitives a and b are transformed into observed primitives c and d, respectively. For this case we denote

$$e \xrightarrow[\text{deletion}]{r_D(\lambda|e, c, d)} \lambda,$$

where $r_D(\lambda|e, c, d)$ is the *deletion probability of e under the conditions* $a \xrightarrow[\text{substitution}]{} c$, $b \xrightarrow[\text{substitution}]{} d$. For simplicity we also use $r_D(e|c, d)$ to denote $r_D(\lambda|e, c, d)$.

3) e is transformed into an observed relation g in ω' while both of its two end primitives a and b are transformed into c and d, respectively. For this case we denote

$$e \xrightarrow[\text{substitution}]{r_S(g|e, c, d)} g,$$

where $r_S(g|e, c, d)$ is the *substitution probability* or *density* of e by g under the conditions $a \xrightarrow[\text{substitution}]{} c$, $b \xrightarrow[\text{substitution}]{} d$. Similar to the case of primitive substitutions, e can also be substituted by itself—that is, e is assumed to be an observed version of itself. Each of the conditional probability or density values defined previously will be called the *relation deformation probability* or *density* of e. Let $D_S(e|c, d)$ denote the set of all possible observed versions of e under the condition that the two end primitives a and b of e are substituted by c and d, respectively, in ω'. If we use $r_S(e|c, d)$ to denote $\sum_{g \in D_S(e|c, d)} r_S(g|e, c, d)$ for discrete z in all $g = (v, z)$, or $\int_{g \in D_S(e|c, d)} r_S(g|e, c, d)$ for

continuous z in all $g = (v, z)$, then we have the following equality:

$$r_D(e|c, d) + r_S(e|c, d) = 1.$$

Again, a substitution of $e = (u, x)$ by $g = (v, z)$ can be decomposed into two steps [1] as follows:

$$e = (u, x) \xrightarrow[\text{syntactic substitution}]{p(v|u, c, d)} f = (v, y) \xrightarrow[\text{semantic substitution}]{q(z|v, u, c, d)} g = (v, z)$$

$$\uparrow \qquad\qquad \uparrow \qquad\qquad \uparrow$$

pure relation — semipure relation — observed relation

where the interpretations of all notations are analogous to those for the primitive substitution except that the probability or density values p and q now depend further on the observed primitives c and d at the two ends of g. The substitution probability or density of e by g therefore is

$$r_S(g|e, c, d) = p(v|u, c, d)q(z|v, u, c, d).$$

Similar to the case of primitive substitutions, we have $\int_z q(z|v, u, c, d)\, dz = 1$ for continuous z, or $\sum_z q(z|v, u, c, d) = 1$ for discrete z and finally, $r_S(e|c, d) = \sum_v p(v|u, c, d)$.

D. Pattern Deformation Probabilities or Densities

We are now ready to compute the deformation probability or density $P(\omega'|\omega)$ of an observed pattern ω' from a pure pattern ω with $\omega = (N, B, \mu, \epsilon)$ over $V_N \cup V_B$ and $\omega' = (N', B', \mu', \epsilon')$ over $V_N \cup V_{B'}$, where

$$N' = \{\alpha'_i | i = 1, 2, \cdots, n_{N'}\},$$

$$N = \{\alpha_j | j = 1, 2, \cdots, n_N\},$$

$$B' = \{\gamma'_k | \gamma'_k = (\alpha'_{k_1}, \alpha'_{k_2}), \quad k = 1, 2, \cdots, n_{B'}\},$$

$$B = \{\gamma_l | \gamma_l = (\alpha_{l_1}, \alpha_{l_2}), \quad l = 1, 2, \cdots, n_B\},$$

$$V_{N'} = \{a'_i | a'_i = \mu'(\alpha'_i) = (s'_i, x'_i), \quad i = 1, 2, \cdots, n_{N'}\},$$

$$V_N = \{a_j | a_j = \mu(\alpha_j) = (s_j, x_j), \quad j = 1, 2, \cdots, n_N\},$$

$$V_{B'} = \{e'_k | e'_k = \epsilon'(\gamma'_k) = (u'_k, z'_k), \quad k = 1, 2, \cdots, n_{B'}\},$$

$$V_B = \{e_l | e_l = \epsilon(\gamma_l) = (u_l, z_l), \quad l = 1, 2, \cdots, n_B\}.$$

In the above notations note that $n_{N'} \leqslant n_N$ and $n_{B'} \leqslant n_B$ because $N' \subset N$ and $B' \subset B$. Also notice that $a_j = \mu(\alpha_j)$ is the pure primitive on node α_j and that $a_{l_1} = \mu(\alpha_{l_1})$ and $a_{l_2} = \mu(\alpha_{l_2})$ are two pure primitives at the ends of pure relation $e_l = \epsilon(\gamma_l)$, where $\gamma_l(\alpha_{l_1}, \alpha_{l_2})$. For discussion convenience we define the following notations:

$$M_S = \{j | \alpha_j \in N \text{ and } a_j \xrightarrow[\text{substitution}]{} a'_{i_j}\},$$

$$M_D = \{j | \alpha_j \in N \text{ and } a_j \xrightarrow[\text{deletion}]{} \lambda\},$$

$$A_{SS} = \{l | \gamma_l = (\alpha_{l_1}, \alpha_{l_2}) \in B, \text{ and } e_l \xrightarrow[\text{substitution}]{} e'_{k_l} \text{ while } \cdot a_{l_1} \xrightarrow[\text{substitution}]{} a'_{k_{l,1}}, a_{l_2} \xrightarrow[\text{substitution}]{} a'_{k_{l,2}}\},$$

$$A_{SD} = \{l | \gamma_l = (\alpha_{l_1}, \alpha_{l_2}) \in B, \text{ and } e_l \xrightarrow[\text{deletion}]{} \lambda \text{ while } \cdot a_{l_1} \xrightarrow[\text{substitution}]{} a'_{k_{l,1}}, a_{l_2} \xrightarrow[\text{substitution}]{} a'_{k_{l,2}}\},$$

$$A_{DD} = \{l | \gamma_l = (\alpha_{l_1}, \alpha_{l_2}) \in B, \text{ and } e_l \xrightarrow[\text{deletion}]{} \lambda \text{ because } \cdot \alpha_{l_1} \xrightarrow[\text{deletion}]{} \lambda \text{ or/and } \alpha_{l_2} \xrightarrow[\text{deletion}]{} \lambda\},$$

where M_S specifies the set of indices of the nodes in ω whose corresponding primitives are substituted in ω', and other sets are similarly interpreted. Note that $M_S \cup M_D = \{i | 1 \leqslant i \leqslant n_N\}$ and that $A_{SS} \cup A_{SD} \cup A_{DD} = \{i | 1 \leqslant i \leqslant n_B\}$. Also let $a \xrightarrow[\text{p.d.}]{} [a']$ denote a primitive deformation where $[a']$ is either a label a' for a primitive substitution or a null symbol λ for a primitive deletion, and let $e \xrightarrow[\text{r.d.}]{} [e']$ denote a relation deformation with $[e']$ being similarly interpreted. Then recalling the assumptions we made previously regarding the interdependency of primitive and relation deformations, we can compute $P(\omega'|\omega)$ as follows:

$$\begin{aligned}
P(\omega'|\omega) &= P\left\{\left(a_j \xrightarrow[\text{p.d.}]{} [a'_{i_j}], j = 1, 2, \cdots, n_N\right), \left(e_l \xrightarrow[\text{r.d.}]{} [e'_{k_l}], l = 1, 2, \cdots, n_B\right)\right\} \\
&= P\left\{a_j \xrightarrow[\text{p.d.}]{} [a'_{i_j}], \quad j = 1, 2, \cdots, n_N\right\} \\
&\quad \cdot P\left\{e_l \xrightarrow[\text{r.d.}]{} [e'_{k_l}], \quad l = 1, 2, \cdots, n_B | a_j \xrightarrow[\text{p.d.}]{} [a'_{i_j}], \quad j = 1, 2, \cdots, n_N\right\} \\
&= \left[P\{a_j \xrightarrow[\text{subst.}]{} a'_{i_j} \text{ for all } j \in M_S\} \cdot P\{a_j \xrightarrow[\text{del.}]{} \lambda \text{ for all } j \in M_D\}\right] \\
&\quad \cdot \left[P\{e_l \xrightarrow[\text{subst.}]{} e'_{k_l} \text{ for all } l \in A_{SS} | a_{l_1} \xrightarrow[\text{subst.}]{} a'_{k_{l,1}}, a_{l_2} \xrightarrow[\text{subst.}]{} a'_{k_{l,2}}\} \right. \\
&\quad \cdot P\{e_l \xrightarrow[\text{del.}]{} \lambda \text{ for all } \in A_{SD} | a_{l_1} \xrightarrow[\text{subst.}]{} a'_{k_{l,1}}, a_{l_2} \xrightarrow[\text{subst.}]{} a'_{k_{l,2}}\} \\
&\quad \left. \cdot P\{e \xrightarrow[\text{del.}]{} \lambda \text{ for all } l \in A_{DD} | a_{l_1} \xrightarrow[\text{del.}]{} \lambda \text{ or } / \text{ and } a_{l_2} \xrightarrow[\text{del.}]{} \lambda\}\right] \\
&= \left[\prod_{j \in M_S} r_S(a'_{i_j}|a_j) \cdot \prod_{j \in M_D} r_D(\lambda|a_j)\right] \\
&\quad \cdot \left[\prod_{l \in A_{SS}} r_S(e'_{k_l}|e_l, a'_{k_{l,1}}, a'_{k_{l,2}}) \right. \\
&\quad \left. \cdot \prod_{l \in A_{SD}} r_D(\lambda|e_l, a'_{k_{l,1}}, a'_{k_{l,2}}) \cdot \prod_{l \in A_{DD}} (1.0)\right] \\
&= \prod_{j \in M_S} r_S(a'_{i_j}|a_j) \cdot \prod_{j \in M_D} r_D(\lambda|a_j) \\
&\quad \cdot \prod_{l \in A_{SS}} r_S(e'_{k_l}|e_l, a'_{k_{l,1}}, a'_{k_{l,2}}) \\
&\quad \cdot \prod_{l \in A_{SD}} r_D(\lambda|e_l, a'_{k_{l,1}}, a'_{k_{l,2}}),
\end{aligned}$$

where "del." and "subst." are abbreviations for deletion

and substitution, respectively. If syntactic and semantic substitution probability or density values are available, $r_S(a'_{i_j}|a_j)$ may be substituted by $p(s'_{i_j}|s_j) \cdot q(x'_{i_j}|s'_{i_j}, s_j)$, and $r_S(e'_{k_l}|e_l, a'_{k_{l,1}}, a'_{k_{l,2}})$ by $p(u'_{k_l}|u_l, a'_{k_{l,1}}, a'_{k_{l,2}}) \cdot q(z'_{k_l}|u'_{k_l}, u_l, a'_{k_{l,1}}, a'_{k_{l,2}})$, according to the notations defined previously. Note that in some cases decomposition of a pure primitive or relation substitution into two steps may not be obvious, then the total substitution probability or density values, $r_S(a'_{i_j}|a_j)$ and $r_S(e'_{k_l}|e_l, a'_{k_{l,1}}, a'_{k_{l,2}})$, should be inferred and used directly. $P(\omega'|\omega)$ as usual will be called the *pattern deformation probability or density of* ω' *from* ω [1].

III. Subgraph Error-Correcting Isomorphisms of Attributed Relational Graphs

As pointed out in the Introduction graph isomorphisms are useful for pattern matching. In this section we define subgraph error-correcting isomorphisms for matching ARG's. The error-correcting capability of the defined isomorphisms allows two terminals with nonidentical labels to be matched. In addition attributes contained in the primitives and relations expressed in terms of semantic vectors can also be handled. These two features are the advantages of error-correcting isomorphisms over conventional graph isomorphism [1], [2]. An illustrative example is included.

A. Definition of Subgraph Error-Correcting Isomorphisms

To facilitate the description of the definition to be proposed, we review or define the following notations, using all symbols given in Section II-D.

1) $D_S(a_j) = \{a'_{i_j} | a_j \xrightarrow[\text{subst.}]{} a'_{i_j}\}$ which, as mentioned in Section II-B, is the set of all possible observed versions in ω' of primitive a_j in ω.
2) $D_S(e_l|a'_{k_{l,1}}, a'_{k_{l,2}}) = \{e'_{k_l} | e_l \xrightarrow[\text{subst.}]{} e'_{k_l}$ while $a_{l_1} \xrightarrow[\text{subst.}]{} a'_{k_{l,1}}, a_{l_2} \xrightarrow[\text{subst.}]{} a'_{k_{l,2}}\}$ which is the set of all possible observed versions in ω' of relation e in ω, constrained by the two end primitive substitutions, as mentioned in Section II-C.
3) $$D(a_j) = \begin{cases} D_S(a_j), & \text{if } r_D(\lambda|a_j) = 0, \\ D_S(a_j) \cup \{\lambda\}, & \text{if } r_D(\lambda|a_j) \neq 0. \end{cases}$$

   a_j and its corresponding node α_j are called *deletable* if $\lambda \in D(a_j)$.

4) $$D(e_l|a'_{k_{l,1}}, a'_{k_{l,2}}) = \begin{cases} D_S(e_l|a'_{k_{l,1}}, a'_{k_{l,2}}) \\ \qquad \text{if } r_D(\lambda|e_l, a'_{k_{l,1}}, a'_{k_{l,2}}) = 0 \\ D_S(e_l|a'_{k_{l,1}}, a'_{k_{l,2}}) \cup \{\lambda\} \\ \qquad \text{if } r_D(\lambda|e_l, a'_{k_{l,1}}, a'_{k_{l,2}}) \neq 0. \end{cases}$$

   e_l and its corresponding branch γ_l are called *deletable* if $\lambda \in D(e_l|a'_{k_{l,1}}, a'_{k_{l,2}})$.

Definition 3: Let $\omega' = (N', B', \mu', \epsilon')$ over $V_{N'} \cup V_{B'}$ be an observed ARG, and $\omega = (N, B, \mu, \epsilon)$ over $V_N \cup V_B$ be a pure ARG. A function $h: N' \to N$ is called a subgraph error-correcting isomorphism (SGECI) from ω' to ω, denoted as $h: \omega' \to \omega$, if the following conditions are satisfied:

1) h is one-to-one, i.e., for any two nodes $\alpha'_1, \alpha'_2 \in N'$, $h(\alpha'_1) = h(\alpha'_2)$ implies $\alpha'_1 = \alpha'_2$. Note that h is not onto when $N' \neq N$ because $N' \subset N$.
2) Let $M_S^h = \{j | \alpha_j \in N, h(\alpha') = \alpha_j$ for some $\alpha' \in N'\}$ which, similar to M_S defined in Section II-D, is the set of the indices of all the nodes in the range of function h. Let $M_D^h = \{j | j = 1, 2, \cdots, n_N\} - M_S^h$ which, similar to M_D, is the set of the indices of the nodes in ω which are not mapped to by any node in ω'. We also define the inverse function h^{-1} of h such that $h^{-1}(\alpha_j) = \alpha'_{i_j}$ if and only if $h(\alpha'_{i_j}) = \alpha_j$ for all $j \in M_S^h$. Then,
   a) for each $j \in M_S^h$, $\mu'(h^{-1}(\alpha_j)) \in D(\mu(\alpha_j))$ should be true; and
   b) for each $j \in M_D^h$, $\lambda \in D(\mu(\alpha_j))$ should be true.
3) Let $A_{SS}^h = \{l | \gamma_l = (\alpha_{l_1}, \alpha_{l_2}) \in B, l_1 \in M_S^h, l_2 \in M_S^h$, and $h^{-1}(\gamma_l) = (h^{-1}(\alpha_{l_1}), h^{-1}(\alpha_{l_2})) \in B'\}$ which, similar to A_{SS} defined previously, is the set of the indices of all the branches in ω whose corresponding branches in ω' are existing. Let $A_{SD}^h = \{l | \gamma_l = (\alpha_{l_1}, \alpha_{l_2}) \in B, l_1 \in M_S^h, l_2 \in M_S^h$, but $h^{-1}(\gamma_l) = (h^{-1}(\alpha_{l_1}), h^{-1}(\alpha_{l_2})) \notin B'\}$ which, similar to A_{SD}, is the set of the indices of all the branches in ω whose corresponding branches in ω' do not exist. Then,
   a) for each $l \in A_{SS}^h$, $\epsilon'(h^{-1}(\gamma_l)) \in D(\epsilon(\gamma_l)|\mu'(h^{-1}(\alpha_{l_1})), \mu'(h^{-1}(\alpha_{l_2})))$ should be true; and
   b) for each $l \in A_{SD}^h$, $\lambda \in D(\epsilon(\gamma_l)|\mu'(h^{-1}(\alpha_{l_1})), \mu'(h^{-1}(\alpha_{l_2})))$ should be true.

Following the notation used previously we define or review the following notations:

$$h^{-1}(\alpha_j) = \alpha'_{i_j}, h^{-1}(\alpha_{l_1}) = \alpha'_{k_{l,1}},$$
$$h^{-1}(\alpha_{l_2}) = \alpha'_{k_{l,2}}, \gamma'_{k_l} = (\alpha'_{k_{l,1}}, \alpha'_{k_{l,2}}),$$
$$a_j = \mu(\alpha_j), a'_{i_j} = \mu'(\alpha'_{i_j}),$$
$$a'_{k_{l,1}} = \mu'(\alpha'_{k_{l,1}}), a'_{k_{l,2}} = \mu'(\alpha'_{k_{l,2}}),$$
$$e_l = \epsilon(\gamma_l), e'_{k_l} = \epsilon'(\gamma'_{k_l}).$$

Then it is easy to derive for an SGECI $h: \omega' \to \omega$ its corresponding pattern deformation probability or density of ω' from ω as

$$P_h(\omega'|\omega) = \prod_{j \in M_S^h} r_S(a'_{i_j}|a_j) \cdot \prod_{j \in M_D^h} r_D(\lambda|a_j) \cdot \prod_{l \in A_{SS}^h} r_S(e'_{k_l}|e_l, a'_{k_{l,1}}, a'_{k_{l,2}}) \cdot \prod_{l \in A_{SD}^h} r_D(\lambda|e_l, a'_{k_{l,1}}, a'_{k_{l,2}}),$$

which specifies a measure of goodness for the matching defined by the SGECI h, and will be called the *likelihood* of h.

Now given two ARG's ω' and ω, since there may exist several SGECI's from ω' to ω due to a variety of node and branch mappings from ω' to ω, it is desirable to determine an SGECI $h_0: \omega' \to \omega$ such that

$$P_{h_0}(\omega'|\omega) = \max_h P_h(\omega'|\omega).$$

h_0 will be called the *maximum-likelihood SGECI* (MLSGECI). The pattern matching problem now is reduced to the search of an MLSGECI among all possible SGECI's from ω' to ω. This is the topic of Section IV.

B. An Illustrative Example

To illustrate the definition of SGECI, let Figs. 1 and 2 be a pure graph $\omega = (N, B, \mu, \epsilon)$ and an observed graph $\omega' = (N', B', \mu', \epsilon')$, respectively, where

$$\begin{aligned} N' &= \{\alpha'_1, \alpha'_2, \alpha'_3\}, & N &= \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}, \\ B' &= \{\gamma'_1 = (\alpha'_1, \alpha'_2), & \gamma'_2 &= (\alpha'_1, \alpha'_2)\}, \\ B &= \{\gamma_1 = (\alpha_2, \alpha_1), & \gamma_2 &= (\alpha_4, \alpha_2), \gamma_3 = (\alpha_4, \alpha_1), \\ & \gamma_4 = (\alpha_3, \alpha_2), & \gamma_5 &= (\alpha_4, \alpha_3)\}. \end{aligned}$$

All labels $a'_i = \mu'(\alpha'_i)$, $a_j = \mu(\alpha_j)$, $e'_k = \epsilon'(\gamma'_k)$, and $e_l = \epsilon(\gamma_l)$ are as shown in the figures. Suppose that we have the following deformations and probabilities:

$D(a_1) = D(c_1) = \{c, a, \lambda\}$[1]
with $r_S(c|c_1) = 0.6$, $r_S(a|c_1) = 0.3$, $r_D(\lambda|c_1) = 0.1$;
$D(a_2) = D(c_2) = \{c, a, b\}$[1]
with $r_S(c|c_2) = 0.7$, $r_S(a|c_2) = 0.2$, $r_S(b|c_2) = 0.1$;
$D(a_3) = D(b) = \{b, a, \lambda\}$
with $r_S(b|b) = 0.7$, $r_S(a|b) = 0.1$, $r_D(\lambda|b) = 0.2$;
$D(a_4) = D(a) = \{a\}$
with $r_S(a|a) = 1.0$;
$D(e_1|c, c) = D(x|c, c) = \{x, \lambda\}$
with $r_S(x|x, c, c) = 0.9$, $r_D(\lambda|x, c, c) = 0.1$;
$D(e_1|b, c) = D(x|b, c) = \{x, \lambda\}$
with $r_S(x|x, b, c) = 0.8$, $r_D(\lambda|x, b, c) = 0.2$;
$D(e_2|a, c) = D(z|a, c) = \{z\}$
with $r_S(z|z, a, c) = 1.0$
$D(e_2|a, b) = D(z|a, b) = \{z, y\}$
with $r_S(z|z, a, b) = 0.8$, $r_S(y|z, a, b) = 0.2$;
$D(e_3|a, c) = D(z|a, c) = \{z\}$
with $r_S(z|z, a, c) = 1.0$;
$D(e_4|b, c) = D(z|b, c) = \{z, \lambda\}$
with $r_S(z|z, b, c) = 0.9$, $r_D(\lambda|z, b, c) = 0.1$;
$D(e_4|b, b) = D(z|b, b) = \{z\}$
with $r_S(z|z, b, b) = 1.0$;
$D(e_5|a, b) = D(y|a, b) = \{y, x\}$
with $r_S(y|y, a, b) = 0.9$, $r_S(x|y, a, b) = 0.1$

[1]Subscriptions 1,2 in c_1 and c_2 are used to indicate that they are on different nodes so that $r_S(c_1|c_2) \neq r_S(c_2|c_2)$.

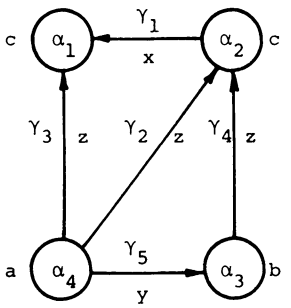

Fig. 1. Pure graph ω.

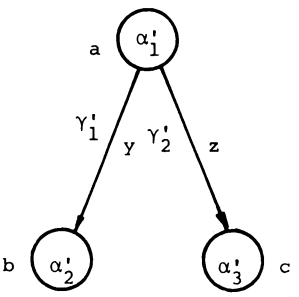

Fig. 2. Observed graph ω'.

We want to show the following one-to-one function h is an SGECI from ω' to ω:

$$h: \begin{cases} \alpha'_1 \to \alpha_4 \\ \alpha'_2 \to \alpha_3 \\ \alpha'_3 \to \alpha_2 \end{cases}.$$

Here we have

$M_S^h = \{2, 3, 4\}$, $M_D^h = \{1\}$,

$A_{SS}^h = \{2, 5\}$ because γ_2 and γ_5 correspond to γ'_2 and γ'_1,

$A_{SD}^h = \{4\}$ because γ_4 corresponds to (α'_2, α'_3) which is missing in ω'.

Now, we check conditions 2a), 2b), 3a), and 3b) in order:

2a) for $j = 2$, $\mu'(h^{-1}(\alpha_2)) = \mu'(\alpha'_3) = c \in D(\mu(\alpha_2)) = D(a_2) = \{c, a, b\}$;
for $j = 3$, $\mu'(h^{-1}(\alpha_3)) = \mu'(\alpha'_2) = b \in D(\mu(\alpha_3)) = D(a_3) = \{b, a, \lambda\}$;
for $j = 4$, $\mu'(h^{-1}(\alpha_4)) = \mu'(\alpha'_1) = a \in D(\mu(\alpha_4)) = D(a_4) = \{a\}$;

2b) for $j = 1$, $\lambda \in D(\mu(\alpha_1)) = D(a_1) = \{c, a, \lambda\}$;

3a) for $l = 2$, $\epsilon'(h^{-1}(\gamma_2)) = \epsilon'((h^{-1}(8\alpha_4), n^{-1}(\alpha_2)) = \epsilon'((\alpha'_1, \alpha'_3)) = \epsilon'(\gamma'_2) = z \in D(\epsilon(\gamma_2)|\mu'(h^{-1}(\alpha_4)), \mu'(h^{-1}(\alpha_2))) = D(e_2|a, c) = \{z\}$;
for $l = 5$, $\epsilon'(h^{-1}(\gamma_5)) = \epsilon'((h^{-1}(\alpha_4), h^{-1}(\alpha_3)) = \epsilon'((\alpha'_1, \alpha'_2)) = \epsilon'(\gamma'_1) = y \in D(\epsilon(\gamma_5)|\mu'(h^{-1}(\alpha_4)), \mu'(h^{-1}(\alpha_3))) = D(e_5|a, b) = \{y, x\}$;

3b) for $l = 4$, $\lambda \in D(\epsilon(\gamma_4)|\mu'(h^{-1}(\alpha_3)), \mu'(h^{-1}(\alpha_2))) = D(e_4|b, c) = \{z, \lambda\}$.

Therefore h is an SGECI from ω' to ω. It is now easy to

compute the likelihood $P_h(\omega'|\omega)$ of h as

$$\begin{aligned} P_h(\omega'|\omega) &= [r_S(c|c_2) \cdot r_S(b|b) \cdot r_S(a|a)] \cdot [r_D(\lambda|c_1)] \\ &\quad \cdot [r_S(z|z, a, c) \cdot r_S(y|y, a, b)] \\ &\quad \cdot [r_D(\lambda|z, b, c)] \\ &= (0.7 \times 0.7 \times 1.0) \times (0.1) \\ &\quad \times (1.0 \times 0.9) \times (0.1) \\ &= 0.00441. \end{aligned}$$

Actually it can be shown that there exists another SGECI h' from ω' to ω:

$$h': \begin{cases} \alpha_1' \to \alpha_4 \\ \alpha_2' \to \alpha_2 \\ \alpha_3' \to \alpha_1 \end{cases}$$

with a smaller likelihood

$$\begin{aligned} P_{h'}(\omega'|\omega) &= [r_{SS}(c|c_1) \cdot r_S(b|c_2) \cdot r_S(a|a)] \cdot [r_D(\lambda|b)] \\ &\quad \cdot [r_S(y|z, a, b) \cdot r_S(z|z, a, c)] \\ &\quad \cdot [r_D(\lambda|x, b, c)] \\ &= (0.6 \times 0.1 \times 1.0) \times (0.2) \\ &\quad \times (0.2 \times 1.0) \times (0.2) \\ &= 0.00048. \end{aligned}$$

IV. Determination of a MLSGECI as a State-Space Search Problem

In this section we discuss the search of a MLSGECI from an observed ARG ω' to a pure ARG ω. An error-correcting graph isomorphism (ECI) problem has been formulated by Tsai and Fu [1] as a state-space tree-search problem [3]. The negative logarithms of terminal deformation probability or density values are interpreted as path costs in the tree-search procedure, and a uniform-cost algorithm can be applied for finding a MLSGECI. But Tsai and Fu [1] further propose a more efficient ordered-search algorithm for finding a structure-preserved ECI after proposing a set of estimation rules for extracting heuristic information from the terminal deformation probability or density values to speed up the search. Here we extend that algorithm for the purpose of determining a MLSGECI from ω' to ω. The essence of an ordered-search procedure is first reviewed [3]. A SGECI problem is again formulated as a state-space search problem. New estimation rules for extracting heuristic information to speed up the search is proposed, followed by the ordered-search algorithm for determining an MLSGECI from an observed ARG ω to a pure ARG ω.

A. Ordered-Search Algorithms for Finding ECI's

In a state-space tree search problem each state description is called a *node*. Applicable *operators* are defined and applied to nodes to obtain their *successors*. If a sequence of operators leads a *start node* to one of the *goal nodes*, a corresponding *path* through the state-space will then be defined and is called the *solution path* of the search problem. The process of generating all successors of a node is called *expanding the node*. Then an ordered-search algorithm uses an evaluation function to order nodes for expansion [3]. For a node N_i in the state space, an evaluation function is usually taken as

$$\hat{f}(N_i) = \hat{g}_1(N_i) + \hat{g}_2(N_i),$$

where $\hat{g}_1(N_i)$ is the minimum total path cost from the start node to N_i computed during the search, and $\hat{g}_2(N_i)$ is a *consistent lower-bounded estimate*, using any heuristic information available, of $g_2(N_i)$ which is the optimal path cost from N_i to a goal node. Then as long as $\hat{g}_2(N_i) \leqslant g_2(N_i)$ for all N_i, a corresponding ordered-search algorithm will expand fewer nodes, compared with a search algorithm using no heuristic information (i.e., $g_2(N_i)$ set 0) such as a uniform-cost search algorithm. The ordered-search algorithm will still be guaranteed to find a minimum cost solution path in the state-space search.

For the case of finding MLSGECI from an observed ARG ω' to a pure ARG ω during matching the nodes and branches in ω' with those in ω, it is natural to consider the negative logarithms of all the corresponding primitive and relation probability or density values as partial path costs. Both $\hat{g}_1$ and $\hat{g}_2$ values therefore can be computed from these negative logarithm values. An ordered-search algorithm then becomes possible for finding a MLSGECI from ω' to ω. Before proposing such an algorithm, we have to formulate the search of an SGECI as a state-space tree-search problem.

B. State-Space Formulation for Finding SGECI's

To find a SGECI from $\omega' = (N', B', \mu', \epsilon')$ over $V_{N'} \cup V_{B'}$ to $\omega = (N, B, \mu, \epsilon)$ over $V_N \cup V_B$, we first add to N' a set of null nodes (denoted as λ) so that the augmented N', denoted as N'', has the same number (n_N) of nodes as N has. Note that $n_{N'} \leqslant n_N$ since $N' \subset N$. Next we number the nodes in N'' in such a way that the first $n_{N'}$ nodes are all those in N', and the rest are all the null nodes added. That is $\alpha_i' \in N'$ for all $1 \leqslant i \leqslant n_{N'}$ and $\alpha_i' = \lambda$ for all $n_{N'} + 1 \leqslant i \leqslant n_N$. We also number those nodes in N in an arbitrary order. The state-space formulation for finding a SGECI $h: \omega' \to \omega$ is as follows.

C. State-Space Formulation of a SGECI Problem

1) *State Description*: A state is described by a collection M of 2-tuples (i, j), each of which denotes a pair of *matched* nodes $\alpha_i' \in N''$ and $\alpha_j \in N$ found so far. When $n_{N'} + 1 \leqslant i \leqslant n_N$, α_i' is λ as defined, which means that α_j is matched with no node in N', or that the primitive $\mu(\alpha_j)$ in ω is deleted when ω is deformed into ω'. The initial state is $M = \varnothing$, the empty set.

2) *Operators*: Let $M_1 = \{i | (i, j) \in M\}$ and $M_2 = \{j | (i, j) \in M\}$, then an operator performs the following

actions to a state M. Pick up a node $\alpha'_k \in N''$ with $k \notin M_1$, and a node $\alpha_l \in N$ with $l \notin M_2$, and form a 2-tuple (k, l). Check the *validity* of (k, l) with the following conditions which are derived from the definition of SGECI (Definition 3):

(a) When $1 \leqslant k \leqslant n_N$, (i.e., when $\alpha'_k \in N'$),
  ⟨1⟩ it should be true that $\mu'(\alpha'_k) \in D(\mu(\alpha_l))$, i.e., $\mu'(\alpha'_k)$ should be an observed version of $\mu(\alpha_l)$ and
  ⟨2⟩ for each $(i, j) \in M$, if $1 \leqslant i \leqslant n_{N'}$, then the following applies.
    (i) If $\gamma'_m = (\alpha'_k, \alpha'_i) \in B'$ (or $\gamma'_m = (\alpha'_i, \alpha'_k \in B')$, it should be true that $\gamma_n = (\alpha_l, \alpha_j) \in B$ (or $\gamma_n = (\alpha_j, \alpha_l) \in B$) and $\epsilon'(\gamma'_m) \in D(\epsilon(\gamma_n)|\mu'(\alpha'_k), \mu'(\alpha'_i))$ (or $\epsilon'(\gamma'_m) \in D(\epsilon(\gamma_n)|\mu'(\alpha'_i), \mu'(\alpha'_k)))$, i.e., $\epsilon'(\gamma'_m)$ should be an observed version of $\epsilon(\gamma_n)$.
    (ii) If $\gamma'_m = (\alpha'_k, \alpha'_i) \notin B'$ (or $\gamma'_m = (\alpha'_i, \alpha'_k) \notin B'$), but $\gamma_n = (\alpha_l, \alpha_j) \in B$ (or $\gamma_n = (\alpha_j, \alpha_l) \in B$), then it should be true that $\lambda \in D(\epsilon(\gamma_n)|\mu'(\alpha'_k), \mu'(\alpha'_i))$ (or $\lambda \in D(\epsilon(\gamma_n)|\mu'(\alpha'_i), \mu'(\alpha'_k)))$, i.e., γ_n should be deletable.

(b) When $n_{N'} + 1 \leqslant k \leqslant n_N$ (i.e., when α'_k is a null node), it should be true that $\lambda \in D(\mu(\alpha_l))$, i.e., α_l should be deletable.

Add (k, l) to M if it is valid. Otherwise try any other 2-tuple (k', l').

3) *The Goal State*: A state M is a goal state if its corresponding M_1 includes the indices of all the nodes in N''. With this formulation the evaluation function for an ordered-search algorithm now can be written as

$$\hat{f}(N_M) = \hat{g}_1(N_M) + \hat{g}_2(N_M)$$

where N_M is a node in the state space with a state description M. To facilitate the computation of $\hat{g}_1(N_M)$ in the algorithm, we define the *cost for adding a valid 2-tuple* (k, l) *to* M, $c(k, l)$, as follows.

(a) When $1 \leqslant k \leqslant n_{N'}$, let

$$R_1 = \{\gamma'_m | \gamma'_m = (\alpha'_k, \alpha'_i) \in B', i \in M_1, \text{and } 1 \leqslant i \leqslant n_{N'}\},$$

$$R_2 = \{\gamma'_m | \gamma'_m = (\alpha'_i, \alpha'_k) \in B', i \in M_1, \text{and } 1 \leqslant i \leqslant n_{N'}\},$$

$$R_3 = \{\gamma_n | \gamma'_m = (\alpha'_k, \alpha'_i) \notin B', i \in M_1, \text{and } 1 \leqslant i \leqslant n_{N'}, \text{ but } \gamma_n = (\alpha_l, \alpha_j) \in B, \text{with } (i, j) \in M\},$$

$$R_4 = \{\gamma_n | \gamma'_m = (\alpha'_i, \alpha'_k) \notin B', i \in M_1, \text{and } 1 \leqslant i \leqslant n_{N'}, \text{ but } \gamma_n = (\alpha_j, \alpha_l) \in B, \text{with } (i, j) \in M\}.$$

R_1 and R_2 together denote the set of all those branches in B', each of which connects α'_k and a matched node α'_i in N'. Note that each branch γ'_m in R_1 and R_2 is corresponded to by a branch γ_n in B. Also R_3 and R_4 above together denote the set of all those branches in B, each of which connects α_l and a matched node α_j in N but corresponds to a nonexisting branch in B'. Then we define

$$\begin{aligned} c(k, l) = & -\ln r_S(\mu'(\alpha'_k)|\mu(\alpha_l)) \\ & + \sum_{\gamma'_m \in R_1} [-\ln r_S(\epsilon'(\gamma'_m)|\epsilon(\gamma_n), \mu'(\alpha'_k), \mu'(\alpha'_i))] \\ & + \sum_{\gamma'_m \in R_2} [-\ln r_S(\epsilon'(\gamma'_m)|\epsilon(\gamma_n), \mu'(\alpha'_i), \mu'(\alpha'_k))] \\ & + \sum_{\gamma_n \in R_3} [-\ln r_D(\lambda|\epsilon(\gamma_n), \mu'(\alpha'_k), \mu'(\alpha'_i))] \\ & + \sum_{\gamma_n \in R_4} [-\ln r_D(\lambda|\epsilon(\gamma_n), \mu'(\alpha'_i), \mu'(\alpha'_k))]. \end{aligned}$$

(b) When $n_{N'} + 1 \leqslant k \leqslant n_N$, we define

$$c(k, l) = -\ln r_D(\lambda|\mu(\alpha_l)).$$

Then for a node N_M in the state space with $M = \{(i_1, j_1), (i_2, j_2), \cdots, (i_L, j_L)\}$ where $(i_L, j_L) = (k, l)$ is the most recently added 2-tuple to M, $\hat{g}_1(N_M)$ is set as

$$\hat{g}_1(N_M) = \sum_{k=1}^{L} c(i_k, j_k),$$

which is computed during the search in the following way. First set $\hat{g}_1(N_{M_0}) = 0$ at the start node $N_{M_0} = \varnothing$. Then whenever a 2-tuple (k, l) is added to M at node N_M, add $c(k, l)$ to $\hat{g}_1(N_M)$.

D. Heuristic Information for Speeding up the Search

As to the computation of $\hat{g}_2(N_M)$ which, in the case of finding an MLSGECI here, is a consistent estimate of the total cost $g_2(N_M)$ for matching all the remaining nodes in ω' and ω after α'_k and α_l are matched (i.e., after (k, l) is added to M), quite an amount of heuristic information useful for estimating $g_2(N_M)$ is found to lie in two partial graphs, one in ω' consisting of all the nodes related to α'_k, and the other in ω consisting of all the nodes related to α_l. The following estimation rules for computing $g_2(N_M)$ reveals this point.

Recall $M_1 = \{i | (i, j) \in M\}$, $M_2 = \{j | (i, j) \in M\}$.

Define $M_{31} = \{i | \gamma'_m = (\alpha'_i, \alpha'_k) \in B', i \notin M_1\}$ with n_{31} elements;
$M_{32} = \{i | \gamma'_m = (\alpha'_k, \alpha'_i) \in B', i \notin M_1\}$ with n_{32} elements;
$M_3 = M_{31} \cup M_{32}$ which is the set of the indices of unmatched nodes in ω' related to α'_k;
$M_{41} = \{j | \gamma_n = (\alpha_j, \alpha_l) \in B, j \notin M_2\}$ with n_{41} elements;
$M_{42} = \{j | \gamma_n = (\alpha_l, \alpha_j) \in B, j \notin M_2\}$ with n_{42} elements;
$M_4 = M_{41} \cup M_{42}$ which is the set of the indices of unmatched nodes in ω related to α_l;
$M_{51} = \{j | j \in M_{41}, \gamma_n = (\alpha_j, \alpha_l), \lambda \in D(\mu(\alpha_j))$ and/or $\lambda \in D(\epsilon(\gamma_n)|\cdot, \mu'(\alpha'_k))\}$ with n_{51} elements, where the dot "$\cdot$" means the label of any node α'_i in ω' with $i \notin M_1 \cup M_{31}$;
$M_{52} = \{j | j \in M_{42}, \gamma_n = (\alpha_l, \alpha_j), \lambda \in D(\mu(\alpha_j))$ and/or $\lambda \in D(\epsilon(\gamma_n)|\mu'(\alpha'_k), \cdot)\}$ with n_{52} ele-

ments, where the dot "·" means the label of any node α'_i in ω' with $i \notin M_1 \cup M_{32}$;
$M = M_{51} \cup M_{52}$ which is the set of the indices of deletable nodes related to α_l, or of the nodes the relations between which and α_l are deletable.

The rules are as follows:

(1) If $n_{31} > n_{41}$ or $n_{32} > n_{42}$ then set $\hat{g}_2(N_M) = \infty$. This means that α'_k cannot match α_l because the number of nodes directionally related to α_k in ω' is larger than the number of nodes directionally related to α_l in ω. Otherwise check the next rule.

(2) If $n_{41} - n_{51} > n_{31}$ or $n_{42} - n_{52} > n_{32}$ then set $\hat{g}_2(N_M) = \infty$. This means that α'_k cannot match α_l, because after deleting all deletable nodes directionally related to α_l and those nodes whose relations with α_l are deletable, the number of remaining nodes directionally related to α_l is still larger than the number of nodes directionally related to α'_k in ω'. Otherwise check the next rule.

(3) For each $i \in M_{31}$ try to find at least one $j \in M_{41}$ such that $\mu'(\alpha'_i) \in D(\mu(\alpha_j))$ and $\epsilon'(\gamma'_m) \in D(\epsilon(\gamma_n)|\mu'(\alpha'_i), \mu'(\alpha'_k))$, where $\gamma'_m = (\alpha'_i, \alpha'_k)$ and $\gamma_n = (\alpha_j, \alpha_l)$. Similarly for each $i \in M_{32}$, try to find at least one $j \in M_{42}$ such that $\mu'(\alpha'_i) \in D(\mu(\alpha_j))$ and $\epsilon'(\gamma'_m) \in D(\epsilon(\gamma_n)|\mu'(\alpha'_k), \mu'(\alpha'_i))$, where $\gamma'_m = (\alpha'_k, \alpha'_i)$ and $\gamma_n = (\alpha_l, \alpha_j)$. If no such $j \in M_4$ exists for some $i \in M_3$, then set $\hat{g}_2(N_M) = \infty$. This means that α'_k cannot match α_l because the labels of the neighboring nodes and branches of α_l cannot be substituted by the labels of the neighboring nodes and branches of α'_k.

Rules (1)–(3) above essentially check the possibility for α'_k to match α_l according to the properties of the *subgraph* error-correcting isomorphisms defined previously. Setting $\hat{g}_2(N_M) = \infty$ when the matching is impossible is to stop further tree expansion from node N_M. Actual estimation of $g_2(N_M)$ is to be performed in Rule (4) described after the following observation is pointed out. First each node α'_i related to α'_k in ω' should be matched with a node α_j related to α_l in ω so as to guarantee a successful match of α'_k with α_l. That is each $i \in M_{31}$ should be matched with a $j \in M_{41}$, and each $i \in M_{32}$ should be matched with a $j \in M_{42}$. Then since there are $n_{41} + n_{42}$ nodes related to α_l in ω and there are $n_{31} + n_{32}$ nodes related to α'_k in ω', exactly $(n_{41} + n_{42}) - (n_{31} + n_{32})$ nodes related to α_l should be *removable* from α_l. Here a node α_j related to α_l is said to be removable from α_l, either if α_j is deletable or if the relation between α_j and α_l is deletable *and* there exists at least one *unmatched* node α'_h *not* related to α'_k in ω' (i.e., $h \notin M_1 \cup M_3$) such that α'_h can be matched with α_j in ω. Note that removable nodes from α_l are all contained in M_5.

(4) Let $n'_{31} = n_{41} - n_{31}$ and $n'_{32} = n_{42} - n_{32}$. Add a set of n'_{31} null nodes M'_{31} to M_{31} so that the augmented set $M''_{31} = M_{31} \cup M'_{31}$ has an identical number of nodes as M_{41}. Define M'_{32} and M''_{32} similarly. Let $M''_3 = M''_{31} \cup M''_{32}$. Then try to find a mapping from M''_3 to M_4 in the following ways.

⟨1⟩ For each $i \in M_{31}$ find a $j \in M_{41}$ such that the cost $c'_{11}(\alpha'_i, \alpha_j) = -\ln r_S(\mu'(\alpha'_i)|\mu(\alpha_j)) - \ln r_S(\epsilon'(\gamma'_m)|\epsilon(\gamma_n), \mu'(\alpha'_k), \mu'(\alpha'_i))$ is minimized among all possible j, where $\gamma'_m = (\alpha'_i, \alpha'_k)$, $\gamma_n = (\alpha_j, \alpha_l)$. That is $\min_{j \in M_{41}} c'_{11}(\alpha'_i, \alpha_j)$ is the minimum cost for a node α'_i related to α'_k in ω' to be matched with a node α_j related to α_l in ω.

⟨2⟩ For each $i \in M_{32}$ find a $j \in M_{42}$ such that the cost $c'_{12}(\alpha'_{ii}, \alpha_j) = -\ln r_S(\mu'(\alpha'_i)|\mu(\alpha_j)) - \ln r_S(\epsilon'(\gamma'_m)|\epsilon(\gamma_n), \mu'(\alpha'_k), \mu'(\alpha'_i))$ is minimized among all possible j, where $\gamma'_m = (\alpha'_k, \alpha'_j)$, $\gamma_n = (\alpha_l, \alpha_j)$. $\min_{j \in M_{42}} c'_{12}(\alpha'_i, \alpha_j)$ is interpreted similarly to $\min_{j \in M_{41}} c'_{11}(\alpha'_i, \alpha_j)$.

Note that mappings defined in ⟨1⟩ and ⟨2⟩ may be many-to-one.

⟨3⟩ For each $j \in M_{51}$ define

$$c'''_{211}(\alpha_j) = -\ln r_D(\lambda|\mu(\alpha_j))$$

which is the cost for α_j to be deleted, and define

$$c'''_{212}(\alpha_j) = \min_{h \notin M_1 \cup M_{31}} \left[-\ln r_D(\lambda|\epsilon(\gamma_n), \mu'(\alpha_h), \mu'(\alpha'_k)) - \ln r_S(\mu'(\alpha'_h)|\mu(\alpha_j)) \right]$$

which is the minimum cost for the relation $\gamma_n = (\alpha_j, \alpha_l)$ to be deleted while α_j is matched with an unmatched node α'_h not related to α'_k in ω'. Note that all unmatched nodes related to α'_k have been mapped in ⟨1⟩ and ⟨2⟩ above. If no h exists, set $c'''_{212}(\alpha_j) = \infty$. Also define $c'''_{21}(\alpha_j) = \min(c'''_{211}, c'''_{212})$ which is the minimum cost for α_j to be removed from α_l. Then for the added null set M'_{31}, try to find a subset $M'_{51} \subset M_{51}$ with n'_{31} indices such that the total cost $c'_{21}(M'_{31}, M'_{51}) = \sum_{j \in M'_{51}} c'''_{21}(\alpha_j)$ is minimized among all possible M'_{51}. That is $\min_{M'_{51} \subset M_{51}} c'_{21}(M'_{31}, M'_{51})$ is the minimum cost for $n'_{31} = n_{41} - n_{31}$ nodes related to α_l to be removed from α_l.

⟨4⟩ For each $j \in M_{52}$ define

$$c'''_{221}(\alpha_j) = -\ln r_D(\lambda|\mu(\alpha_j))$$

and

$$c'''_{222}(\alpha_j) = \min_{h \notin M_1 \cup M_{32}} \left[-\ln r_D(\lambda|\epsilon(\gamma_n), \mu'(\alpha'_k), \mu'(\alpha'_h)) - \ln r_S(\mu'(\alpha'_h)|\mu(\alpha_j)) \right]$$

where $\gamma_n = (\alpha_l, \alpha_j)$. Define $c''_{22}(\alpha_j) = \min(c'''_{221}, c'''_{222})$. $c'''_{221}, c'''_{222}, c''_{22}$ are all interpreted similarly to $c'''_{211}, c'''_{212}, c''_{21}$. Then for the null set M'_{32}, try to find a subset $M'_{52} \subset M_{52}$ with n'_{32} indices such that the total cost $c'_{22}(M'_{32}, M'_{52}) = \sum_{j \in M'_{52}} c''_{22}(\alpha_j)$ is minimized among all possible M'_{52}. That is $\min_{M'_{52} \subset M_{52}} c'_{22}(M'_{32}, M'_{52})$ is the minimum cost for $n'_{32} = n_{42} - n_{32}$ nodes related to α_l to be removed from α_l.

Note that the mappings from M'_{31} to M'_{51} and from M'_{32} to M'_{52} defined in ⟨3⟩ and ⟨4⟩ are one-to-one, but while defining the costs c'''_{212} and c'''_{222}, the chosen α'_h (an unmatched nodes not related to α'_k in ω') for all $j \in M_{51}$ or M_{52} may not be all different (i.e., the mapping from j to h may be many-to-one).

The total cost for the mapping from M''_3 to M_4 now can be summed up as

$$c'(k, l) = \sum_{i \in M_{31}} \min_{j \in M_{41}} c'_{11}(\alpha'_i, \alpha_j) + \sum_{i \in M_{32}} \min_{j \in M_{42}} c'_{12}(\alpha'_i, \alpha_j) + \min_{M'_{51} \subset M_{51}} c'_{21}(M'_{31}, M'_{51}) + \min_{M'_{52} \subset M_{52}} c'_{22}(M'_{32}, M'_{52}),$$

which is a lower-bounded cost for all the nodes related to α'_k in ω' to be matched with all the nodes related to α_l in ω. Finally set $\hat{g}_2(N_M) = c'(k, l)$ as an estimate of $g_2(N_M)$.

Rule ⟨4⟩ provides a consistent lower bound of $g_2(N_M)$. Other lower bounds, either looser or tighter, can also be derived. A looser lower bound has been derived in [17]. A tighter bound for matching structure-preserved graphs is derived in [1].

E. An Ordered-Search Algorithm for Finding MLSGECI's

We are now ready to propose an ordered-search algorithm for determining a MLSGECI from an ARG ω' to another ARG ω. The algorithm essentially is similar to the one proposed in [1] for finding maximum-likelihood structure-preserved ECI's. Both follow that given in [3].

MLSGECI Algorithm:

Input: An observed ARG $\omega' = (N', B', \mu', \epsilon')$ over $V_{N'} \cup V_{B'}$ and a pure ARG $\omega = (N, B, \mu, \epsilon)$ over $V_N \cup V_B$ with $n_{N'} \leqslant n_N$ and $n_{B'} \leqslant n_B$, where n_A means the number of the elements in set A.

Output: A MLSGECI h: $\omega' \rightarrow \omega$ with likelihood $P_h(\omega'|\omega)$ if h exists; otherwise the output $P(\omega'|\omega) = 0$.

Steps: In the following steps, notations M, N_M, M_1, N'', computations of $\hat{g}_1(N_M)$, $\hat{g}_2(N_M)$, and $\hat{f}(N_M)$, and operations for node expansion are all as defined in the previous sections.

(1) Put the start node N_{M_0} with $M_0 = \varnothing$ on a list called OPEN, and set $\hat{f}(N_{M_0}) = 0$.

(2) If OPEN is empty, then no MLSGECI exists; set $P(\omega'|\omega) = 0$ and exist.

(3) Remove from OPEN the node N_M with a smallest $\hat{f}$ value and put it on a list called CLOSED.

(4) If the M_1 of this node N_M is equal to the set of the indices of all the nodes in N'', then an MLSGECI h represented by M is found whose likelihood is given by $P_h(\omega'|\omega) = \text{EXP}[-\hat{g}_1(N_M)]$. Otherwise continue.

(5) Expand node N_M using all operators applicable to M. Compute the value $\hat{f}(N_{M'}) = \hat{g}_1(N_{M'}) + \hat{g}_2(N_{M'})$ for each successor $N_{M'}$ of N_M. Put these successors on OPEN.

(6) Go to Step (2).

In the above algorithm if $\hat{g}_2(N_{M'})$ is always set zero, the algorithm reduces to a uniform-cost search algorithm which, though guaranteeing a MLSGECI, expands more nodes in general in the search than an ordered-search algorithm like the one just proposed.

F. An Illustrative Example

We continue the example given in Section III-B. After performing the proposed MLSGECI algorithm to find a MLSGECI from ω' to ω, where ω' and ω are as shown in Figs. 1 and 2, respectively, the resulting state-space search tree is shown in Fig. 3. Due to the small-sized ARG's ω' and ω we have, and the speedup the algorithm offers, the resulting search tree as shown is quite small in size. The circled numbers besides the nodes specify the expansion order of the nodes. The values besides the arcs are the $\hat{f}(N_M)$ values. The solution path is indicated by the dark arcs, which corresponds, as specified by the goal node $\{(1,4),(2,3),(3,2),(4,1)\}$, to the following mapping:

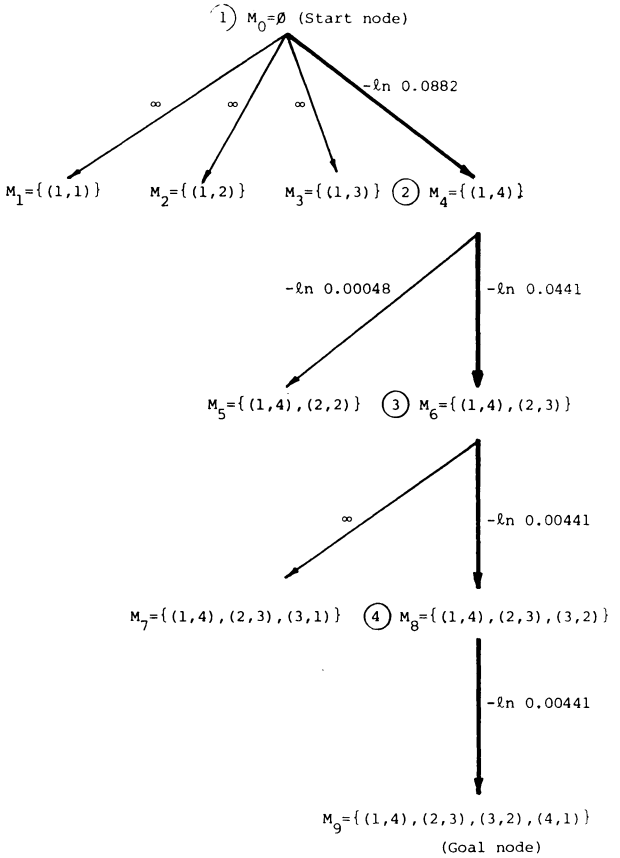

Fig. 3. State-space search tree.

$$h: \begin{cases} \alpha'_1 \rightarrow \alpha_4 \\ \alpha'_2 \rightarrow \alpha_3 \\ \alpha'_3 \rightarrow \alpha_2 \\ \lambda \rightarrow \alpha_1 \end{cases}$$

where $\lambda \rightarrow \alpha_1$ means that α_1 is deleted. In the following we compute selectively the $\hat{f}$ values for nodes N_{M_1}, N_{M_4}, and N_{M_8} for illustrative purpose.

(A) At N_{M_1} where $(k, l) = (1, 1)$:

(a) $R_1 = R_2 = R_3 = R_4 = \varnothing$, $\hat{g}_1 = c(1,1) = -ln\, r_S(a|c_1) = -ln 0.2$.

(b) $M_{31} = \varnothing$, $M_{32} = \{2,3\}$, $n_{31} = 0$, $n_{32} = 2$, $M_{41} = \{2,4\}$, $M_{42} = \varnothing$, $n_{41} = 2$, $n_{42} = 0$.
In Rule (1) set $g_2 = \infty$ since $n_{32} > n_{42}$.

(c) $\hat{f} = \hat{g}_1 + \hat{g}_2 = \infty$.

(B) At N_{M_4} where $(k, l) = (1, 4)$

(a) $R_1 = R_2 = R_3 = R_4 = \varnothing, \hat{g}_1 = c(1,4) = -\ln r_S(a'_1|a_4) = -\ln r_S(a|a) = -\ln 1.0$.

(b) (1) $M_{31} = \varnothing$, $M_{32} = \{2, 3\}$, $n_{31} = 0$, $n_{32} = 2$,
$M_{41} = \varnothing$, $M_{42} = \{1, 2, 3\}$, $n_{41} = 0$, $n_{42} = 3$.
Since $n_{41} = n_{31}$, and $n_{42} > n_{32}$, we check Rule (2).

(2) $M_{51} = \varnothing$, $M_{53} = \{1, 3\}$, $n_{51} = 0$, $n_{52} = 2$.
Since $n_{41} - n_{51} = n_{31}$, $n_{42} - n_{52} < n_{32}$, we check Rule (3).

(3) For $i = 2 \in M_{32}$, choose $j = 3 \in M_{42}$ such that
$a_2' = b \in D(a_j) = \{b, a, \lambda\}$ and
$e_1' = y \in D(e_5|a_1', a_2') = D(e_5|a, b) = \{y, x\}$, where
$e_1' = \epsilon'(\gamma_m')$ for $\gamma_m' = (\alpha_1', \alpha_2') = \gamma_1'$ and
$e_5 = \epsilon(\gamma_n)$ for $\gamma_n = (\alpha_4, \alpha_2) = \gamma_2$. For $i = 3 \in M_{32}$, choose $j = 2 \in M_{42}$ such that
$a_3' = c \in D(a_2) = \{c, a, b\}$ and
$e_2' = z \in D(e_2|a_1', a_3') = D(e_2|a, c) = \{z\}$, where
$e_2' = \epsilon'(\gamma_m')$ for $\gamma_m' = (\alpha_1', \alpha_3') = \gamma_2'$ and
$e_2 = \epsilon(\gamma_n)$ for $\gamma_n = (\alpha_4, \alpha_2) = \gamma_2$.
Therefore, we can check Rule (4). Note that here we can also choose $j = 2$ for $i = 2$ and $j = 1$ for $i = 3$. This fact is used in Rule (4) below.

(4) $n_{13}' = n_{41} - n_{31} = 0$, $n_{32}' = n_{32} - n_4 = 1$
$M_{32}'' = M_{32} \cup \{\lambda\} = \{2, 3, \lambda\}$
⟨2⟩ For $i = 2 \in M_{32}$, if $j = 3 \in M_{42}$ is chosen, then

$$\begin{aligned} c_{12}'(\alpha_2', \alpha_3) &= -\ln r_S(a_2'|a_3) \\ &\quad -\ln r_S(e_1'|e_5, a_1', a_2') \\ &= -\ln r_S(b|a_3) \\ &\quad -\ln r_S(y|y, a, b) \\ &= -\ln 0.7 - \ln 0.9. \end{aligned}$$

If $j = 2 \in M_{42}$ is chosen, then

$$\begin{aligned} c_{12}'(\alpha_2', \alpha_2) &= -\ln r_S(a_2'|a_2) \\ &\quad -\ln r_S(e_1'|e_2, a_1', a_2') \\ &= -\ln r_S(b|a_2) \\ &\quad -\ln r_S(y|z, a, b) \\ &= -\ln 0.1 - \ln 0.2. \end{aligned}$$

Since $-\ln 0.7 - \ln 0.9 < -\ln 0.1 - \ln 0.2$, we choose $j = 3$, and $\min_{j \in M_{41}} c_{12}'(\alpha_i', \alpha_j) = -\ln(0.7 \times 0.9)$ for $i = 2$.
For $i = 3 \in M_{32}$, $c_{12}'(\alpha_i', \alpha_j)$ are computed, for $j = 2$ and 1, to be $c_{12}'(\alpha_3', \alpha_2) = -\ln 0.7 - \ln 1.0$, and $c_{12}'(\alpha_3', \alpha_1) = -\ln 0.6 - \ln 1.0$. Therefore we choose $j = 2$ and $\min_{j \in M_{41}} c_{12}'(\alpha_i', \alpha_j) = -\ln(0.7 \times 1.0)$ for $i = 3$.

⟨4⟩ For $j = 1 \in M_{52}$,
$c_{221}''(\alpha_1) = -\ln r_D(\lambda|\mu(\alpha_1)) = -\ln r_D(\lambda|a_1) = -\ln 0.1$.
To compute $c_{222}''(\alpha_1)$, since $M_1 \cup M_{32} = \{1\} \cup \{2, 3\}$, no h exists and $c_{222}''(\alpha_1)$ is set ∞. So $c_{22}''(\alpha_1) = -\ln 0.1$.
For $j = 3 \in M_{52}$, similarly,
$c_{22}''(\alpha_3) = c_{221}''(\alpha_3) = -\ln r_D(\lambda|\mu(\alpha_3)) = -\ln r_D(\lambda|a_3) = -\ln 0.2$.
Since $c_{22}''(\alpha_3) = -\ln 0.2 < -\ln 0.1 = c_{22}''(\alpha_1)$, we choose $M_{52}' = \{3\} \subset M_{52}$ such that

$$\min_{M_{52}' \subset M_{52}} c_{22}'(M_{32}', M_{52}') = c_{22}''(\alpha_3) = -\ln 0.2.$$

Then $\hat{g}_2$ is set to be

$$\begin{aligned} c'(k, l) &= \sum_{i \in M_{32}} \min_{j \in M_{42}} c_{12}'(\alpha_i', \alpha_j) \\ &\quad + \min_{M_{52}' \subset M_{52}} c_{22}'(M_{32}', M_{52}') \\ &= -\ln(0.7 \times 0.9) \\ &\quad -\ln(0.7 \times 1.0) - \ln 0.2 \\ &= -\ln(0.7 \times 0.9 \times 0.7 \times 0.2) \\ &= -\ln 0.0882. \end{aligned}$$

And so $\hat{f} = \hat{g}_1 + \hat{g}_2 = -\ln 1.0 - \ln 0.0882 = -\ln 0.0882$.

(C) At N_{M_8} where $(k, l) = (3, 2)$
(a) $R_1 = \varnothing$, $R_2 = \{\gamma_2'\}$, $R_3 = \varnothing$, $R_4 = \{\gamma_4\}$ where $\gamma_2' = (\alpha_1', \alpha_3')$, $\gamma_4 = (\alpha_4, \alpha_2)$.

$$\begin{aligned} c(3, 2) &= -\ln r_S(\mu'(\alpha_k')|\mu(\alpha_l)) \\ &\quad + \sum_{\gamma_m' \in R_2} [-\ln r_S(\epsilon'(\gamma_m')|\epsilon'(\gamma_n), \\ &\quad \cdot \mu'(\alpha_i'), \mu'(\alpha_k'))] \\ &\quad + \sum_{\gamma_n \in R_4} [-\ln r_D(\lambda|\epsilon(\gamma_n), \\ &\quad \cdot \mu'(\alpha_i'), \mu'(\alpha_k'))] \\ &= -\ln r_S(a_3'|a_2) - \ln r_S(e_2'|e_2, a_1', a_3') \\ &\quad \cdot - \ln r_D(\lambda|e_4, a_2', a_3') \\ &= -\ln 0.7 - \ln 1.0 - \ln 0.1 \\ &= -\ln 0.07. \end{aligned}$$

$$\begin{aligned} \hat{g}_1 &= c(1, 4) + c(2, 3) + c(3, 2) \\ &= -\ln 1.0 - \ln 0.63 - \ln 0.07 \\ &= -\ln 0.0441 \end{aligned}$$

where $c(2, 3) = -\ln 0.63$ is computed at H_{M_6} (not shown here).

(b) $M_{31} = M_{32} = \varnothing$
$M_{41} = \varnothing$, $M_{42} = \{1\}$
$M_{51} = \varnothing$, $M_{52} = \{1\}$
Since (i) $n_{41} = n_{31} = 0$, $n_{41} = 1 > 0 = n_{32}$,
(ii) $n_{41} - n_{51} = n_{31} = 0$, $n_{42} - n_{52} = 1 - 1 = 0 = n_{32}$, and
(iii) $M_{31} = M_{32} = \varnothing$,
we can check Rule (4) below.

(4) ⟨4⟩ For $j = 1 \in M_{52}$,

$$c_{221}''(\alpha_1) = -\ln r_D(\lambda|\mu(\alpha_1)) = -\ln 0.1$$

and since $M_1 \cup M_{32} = \{1, 2, 3\} \cup \varnothing = \{1, 2, 3\}$, h does not exist. So $c_{222}''(\alpha_1) = \infty$ and $c_{22}''(\alpha_1) =$

$-\ln 0.1$. Therefore

$$\hat{g}_2 = c'(k,l) = \min_{M'_{52} \subset M_{52}} c'_{22}(M'_{32}, M'_{52}) = c'_{22}(\alpha_1) = -\ln 0.1.$$

And so $\hat{f} = \hat{g}_1 + \hat{g}_2 = -\ln 0.0441 - \ln 0.1 = -\ln 0.00441$.

G. Complexity Analysis of MLSGECI Problem

Unfortunately the MLSGECI problem, like the subgraph isomorphism problem is an NP-complete problem [18], and the search of a MLSGECI needs exponential time in the worst case. But in practice most subgraph isomorphism algorithms behave quite well for graphs encountered in real applications, scarcely showing the exponential time requirement [19]. In particular we make the same assumption as that made in [20] for graph homomorphism complexity analysis that the resulting state-space search tree usually will, with the help of heuristic information, reduce to just a single line (i.e., the solution path) or at most a few branches at tree level. The resulting search tree shown in Fig. 3 is a justification of this assumption. A typical search tree with several branches at each tree level contains approximately αn^2 generated nodes, where α is a constant taking care of the branches [20] and n is the number of nodes in the pure graph. What remains is to analyze the complexity of the work required to generate each node in the search tree. Such work is specified in Step (5) of the proposed MLSGECI algorithm. The total complexity of the MLSGECI algorithm will then be αn^2 times the complexity for such work.

Generation of a node $N_{M'}$ while expanding its predecessor N_M in Step (5) of the proposed MLSGECI algorithm, can be further divided into two major parts—validity checking of (k, l) and computation of $f(N_{M'})$. In the following we assume $M = \{(i_1, j_1), (i_2, j_2), \cdots, (i_{L-1}, j_{L-1})\}$ and $M' = M \cup \{(i_L, j_L) = (k, l)\}$.

Validity checking of (k, l): This is Step (2) in the state-space formulation of an SGECI problem. When $1 \leqslant k \leqslant n_{N'}$, an appropriate search technique like hashing is used to check if $\mu'(\alpha'_k) \in D(\mu(\alpha_l))$ requires approximately a constant time c_1. Next for each $(i, j) \in M$, we first have to check if $\gamma'_m = (\alpha'_k, \alpha'_i)$ or (α'_i, α'_k) is in B'. If so then we have to check further if $\gamma_n = (\alpha_l, \alpha_j)$ or (α_j, α_l) is in B and if $\epsilon'(\gamma'_m)$ is in $D(\epsilon(\gamma_n)|\mu'(\alpha'_k), \mu'(\alpha'_i))$ or in $D(\epsilon(\gamma_n)|\mu'(\alpha'_i), \mu'(\alpha'_k))$. If γ'_m is not in B' then we check further if $\gamma_n = (\alpha_l, \alpha_j)$ or (α_j, α_l) is in B or not. If so we have to check if λ is in $D(\epsilon(\gamma_n)|\mu'(\alpha'_k), \mu'(\alpha'_i))$ or in $D(\epsilon(\gamma_n)|\mu'(\alpha'_i), \mu'(\alpha'_k))$. Let m, b', b be the number of elements in M (or M_1 or M_2), B', B, respectively. Then for either case above (γ'_m in B' or not in B'), we need time $m(bc_2 + b'c_2 + c_3)$ where c_2 is a constant time for comparing the identicalness of two branches in B' or in B, and c_3 is a constant time for checking if an edge label $\epsilon'(\gamma'_m)$ or λ is in $D(\epsilon(\gamma_n)|\mu'(\alpha'_k), \mu'(\alpha'_i))$ or in $D(\epsilon(\gamma_n)|\mu'(\alpha'_i), \mu'(\alpha'_k))$. When $n_{N'} + 1 \leqslant k \leqslant n_N$, to check if λ is in $D(\mu(\alpha_l))$ also requires time c_1. In summary we need time no more than $t_1 = c_1 + m(bc_2 + b'c_2 + c_3)$ for validity checking of (k, l).

Computation of $\hat{f}(N_{M'})$: Recall that

$$\hat{f}(N_{M'}) = \hat{g}_1(N_{M'}) + \hat{g}_2(N_{M'})$$

where

$$\hat{g}_1(N_{M'}) = \sum_{k=1}^{L} c(i_k, j_k) = \hat{g}_1(N_M) + c(k, l),$$

and $\hat{g}_2(N_{M'})$ is computed according to the four estimation rules. $\hat{g}_1(N_M)$ is already computed at node N_M. To compute $c(k, l)$ we first have to find out the four sets R_1, R_2, R_3, and R_4. This can be accomplished by checking for all i in M_1 with $1 \leqslant i \leqslant n_N$, if $\gamma'_m = (\alpha'_k, \alpha'_i)$ or (α'_i, α'_k) is in B', and if not, checking further if $\gamma_n = (\alpha_l, \alpha_i)$ or (α_i, α_l) is in B. This requires time no more than $m(c_4 + b'c_2 + bc_2)$ where c_4 is a constant time for number comparison (to check if i is no greater than $n_{N'}$). Next according to the definition of $c(k, l)$, and noting that the total number of the elements in R_1, R_2, R_3, and R_4 is less than m (the number of elements in M_1), we see that the computation of $c(k, l)$ requires time no more than $m(c_5 + c_6)$. c_5 is the time to take the logarithm of each γ_s or γ_D value and c_6 is the time for subtraction (or addition). Therefore to compute

$$\hat{g}_1(N_{M'}) = \hat{g}_1(N_M) + c(k, l)$$

requires time no more than

$$t_2 = m(c_4 + b'c_2 + bc_2) + m(c_5 + c_6) + c_5 = m(c_4 + b'c_2 + bc_2 + c_5 + c_6) + c_5.$$

On the other hand to compute $\hat{g}_2(N_{M'})$, we first have to find the sets M_{31}, M_{32}, M_{41}, M_{42}, M_{51}, and M_{52}. One way to find M_{31} and M_{32} is to check, for all those $i \notin M_1$, if $\gamma'_m = (\alpha'_i, \alpha'_k)$ or (α'_k, α'_i) is in B' or not. Since there are $n' - m$ indices not included in M_1, this requires time equal to $(n' - m)b'c_2$. Similarly to compute M_{41} and M_{42} requires time equal to $(n - m)bc_2$. Here we use n', n, b', and b to denote the numbers of the elements in N', N, B', and B, respectively. c_2 is, as defined previously, the time for comparing two branches in B' or B. To compute M_{51} and M_{52} we have to check, for all j in M_{41} or in M_{42}, if the node α_j, which is connected to α_l by $\gamma_n = (\alpha_j, \alpha_l)$ or (α_l, α_j), is removable from α_l. That is, if λ is in $D(\mu(\alpha_j))$ and/or in $D(\epsilon(\gamma_n)|\cdot, \mu'(\alpha'_k))$ or in $D(\epsilon(\gamma_n)|\mu'(\alpha'_k), \cdot)$, where the dot "$\cdot$" means $\mu'(\alpha'_i)$ with any i not in $M_1 \cup M_{31}$ or not in $M_1 \cup M_{32}$. To check if λ is in $D(\mu(\alpha_j))$ requires time c_1. One way to check if λ is in $D(\epsilon(\gamma_n)|\cdot, \mu'(\alpha'_k))$ or in $D(\epsilon(\gamma_n)|\mu'(\alpha'_k), \cdot)$ is to check, for all i not in $M_1 \cup M_{31}$ or not in $M_1 \cup M_{32}$, if λ is in $D(\epsilon(\gamma_n)|\mu'(\alpha'_i), \mu'(\alpha'_k))$ or in $D(\epsilon(\gamma_n)|\mu'(\alpha'_k), \mu'(\alpha'_i))$. This requires time $(n' - m - n_{31})c_1$ or time $(n' - m - n_{32})c_1$, where n_{31} and n_{32} are the numbers of the elements in M_{31} and M_{32}. So to compute

M_{51} and M_{52} requires time no more than

$$\begin{aligned} & n_{41}[c_1 + (n' - m - n_{31})c_1] \\ & \quad + n_{42}[c_1 + (n' - m - n_{32})c_1] \\ & \leqslant (n - m)[c_1 + (n' - m)c_1] \\ & \quad + (n - m)[c_1 + (n' - m)c_1] \\ & = 2(n - m)(n' - m)c_1 + 2(n - m)c_1, \end{aligned}$$

where n_{41} and n_{42}, both no more than $n - m$, are the numbers of the elements in M_{41} and M_{42}, respectively. In summary to compute all the sets M_{31}, M_{32}, M_{41}, M_{42}, M_{51}, and M_{52} requires time no more than $t_3 = (n' - m)b'c_2 + (n - m)bc_2 + 2(n - m)(n' - m)c_1 + 2(n - m)c_1$.

To compute $\hat{g}_2(N_{M'})$ we next have to analyze the time needed in using the four estimation rules (1)–(4):

(1) We have to count n_{31}, n_{32}, n_{41}, and n_{42}, followed by comparisons of n_{31} with n_{41} and of n_{32} with n_{42}. This requires time

$$\begin{aligned} & (n_{31} + n_{32} + n_{41} + n_{42})c_7 + 2c_4 \\ & \qquad \leqslant [(n' - m) + (n - m)]c_7 + 2c_4 = t_4, \end{aligned}$$

where c_7 is the constant time for a number increment operation and c_4 is the constant time for number comparison.

(2) This steps require time

$$\begin{aligned} (n_{51} + n_{52})c_7 + 2c_6 + 2c_4 & \leqslant (n_{41} + n_{42})c_7 + 2c_6 + 2c_4 \\ & \leqslant (n - m)c_7 + 2c_6 + 2c_4 \\ & = t_5, \end{aligned}$$

where c_6 is the constant time for addition.

(3) It is not difficult to see that this step requires time

$$\begin{aligned} & n_{31}n_{41}(c_1 + c_2) + n_{32}n_{42}(c_1 + c_2) \leqslant \\ & \quad \cdot (n' - m)(n - m)(c_1 + c_2) = t_6. \end{aligned}$$

(4)⟨1⟩ In this step, for each i in M_{31}, we have to compute the cost $c'_{11}(\alpha'_i, \alpha_j)$ for all j in M_{41} and then choose one j with the minimum c'_{11} value. This requires time

$$\begin{aligned} & n_{31}[n_{41}(c_5 + c_6 + c_4)] \\ & \qquad \leqslant (n' - m)(n - m)(c_5 + c_6 + c_4) = t_7. \end{aligned}$$

(4)⟨2⟩ Similar to ⟨4⟩⟨1⟩, this step requires no more time than t_7.

(4)⟨3⟩ For each j in M_{51} we first have to compute c'''_{211} and c'''_{212} and then set $c''_{21}(\alpha_j)$ as the smaller of c'''_{211} and c'''_{212}. To compute c'''_{211} we need time c_5. To compute c'''_{212} we need time $(n' - m - n_{31}) \cdot (2c_5 + c_6 + c_4)$. To compute $c'_{21}(\alpha_j)$ we need time c_4. Therefore we need time

$$\begin{aligned} & n_{51}[c_5 + (n' - m - n_{31})(2c_5 + c_6 + c_4) + c_4] \\ & \quad \leqslant n_{41}[(n' - m - n_{31})(2c_5 + c_6 + c_4) + c_5 + c_4] \\ & \quad \leqslant (n - m)[(n' - m)(2c_5 + c_6 + c_4) + c_5 + c_4]. \end{aligned}$$

Next, we have to find a subset $M'_{51} \subset M_{51}$ with n'_{31} indices such that the cost

$$c'_{21}(M'_{31}, M'_{51}) = \sum_{j \in M'_{51}} c''_{21}(\alpha_j)$$

is minimized. This can be done by first sorting the costs $c''_{21}(\alpha_j)$ for all j in M_{51} and then choosing those n'_{31} indices whose corresponding $c''_{21}(\alpha_j)$ values are the smallest in the n_{51} ones. This requires time Kn_{51}^2 at most for sorting and time $n'_{31}(c_4 + c_6)$ for summing the n'_{31} smallest $c''_{21}(\alpha_j)$ values, or together

$$\begin{aligned} Kn_{51}^2 + n'_{31}(c_4 + c_6) & \leqslant Kn_{41}^2 + (n_{41} - n_{31})(c_4 + c_6) \\ & \leqslant K(n - m)^2 + (n - m)(c_4 + c_6), \end{aligned}$$

where K is a constant and $n'_{31} = n_{41} - n_{31} \leqslant n_{41} \leqslant n - m$. Totally, in this step we need time no more than

$$\begin{aligned} t_8 = (n - m)[(n' - m)(2c_5 + c_6 + c_2) + c_5 + c_2] \\ + K(n - m)^2 + (n - m)(c_4 + c_6). \end{aligned}$$

(4)⟨4⟩ Similar to (4)⟨3⟩ the time needed for this step is also no more than t_8.

In summary the total time needed for estimation rules (1)–(4) above is no more than $t_4 + t_5 + t_6 + t_7 + t_7 + t_8 + t_9$ which, after simplification, is equal to

$$\begin{aligned} t_9 = {} & 2K(n - m)^2 + (c_1 + 3c_2 + 2c_4 + 6c_5 + 4c_6) \\ & \cdot (n' - m)(n - m) \\ & + (2c_2 + 2c_4 + 2c_5 + 2c_6 + 2c_7) \\ & \cdot (n - m) + c_7(n' - m) + 4c_4 + 2c_6. \end{aligned}$$

Now to compute $c'(k, l)$ we need the following time for additions:

$$\begin{aligned} (n_{31} + n_{32} + n'_{31} + n'_{32})c_6 & \leqslant (n_{31} + n_{32} + n_{41} + n_{42})c_6 \\ & \leqslant [(n' - m) + (n - m)]c_6 \\ & = t_{10}. \end{aligned}$$

Therefore the total time to generate a node N_M in the state-space is

$$t = t_1 + t_2 + (t_3 + t_9 + t_{10})$$

which, after simplification, is

$$\begin{aligned} t = {} & m(bc_2 + b'c_2 + c_3 + c_4 + c_5 + c_6) + n'b'c_2 + nbc_2 \\ & + 2K(n - m)^2 + (n' - m)(n - m) \\ & \cdot (3c_1 + 3c_2 + 2c_4 + 6c_5 + 4c_6) + (n - m) \\ & \cdot (2c_1 + 2c_2 + 2c_4 + 2c_5 + 3c_6 + 2c_7) \\ & + (n' - m)(c_6 + c_7) + (c_1 + 4c_4 + c_5 + 2c_6). \end{aligned}$$

Since m is the number of the elements in M, it is not greater than n' which in turn is not greater than n because $N' \subset N$. Also, $b' \leqslant b$ because $B' \subset B$. Therefore the time complexity for t above can be reduced to

$$t \approx K_1 nb + K_2 n^2$$

for large n, where K_1 and K_2 are two constants. When the connectivity of nodes in the pure graph ω is low such that $b < n$ the term $K_2 n^2$ dominates in t. Otherwise $K_1 nb$ dominates. Actually b can be as large as to the order of n^2 when almost every two nodes in N are connected to each other with a branch. Since approximately αn^2 nodes will normally be generated in the state space (see previous discussions), we conclude that the time complexity of the

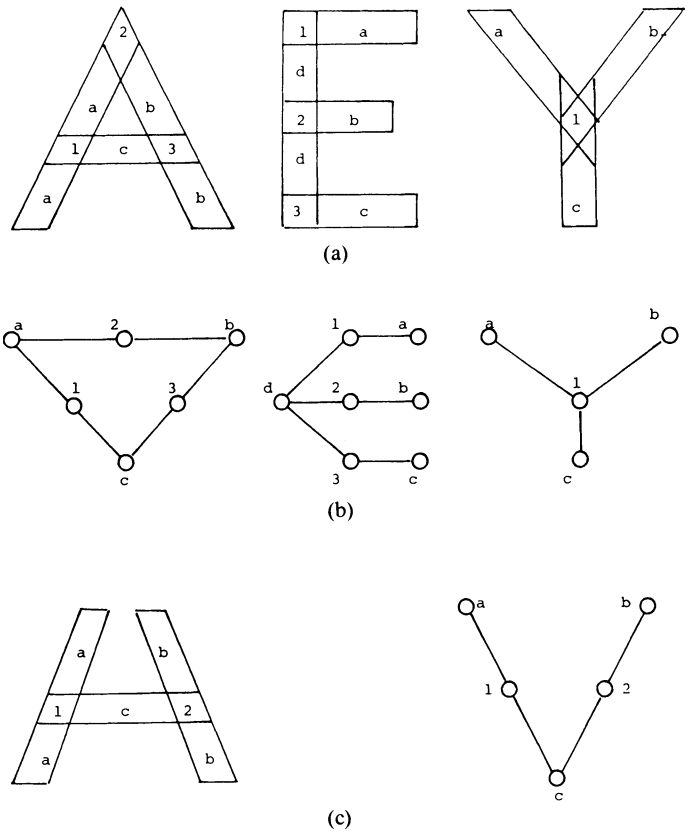

Fig. 4. Shape analysis by primary graphs constructed from decomposed polygonal approximations of given shapes. (a) Decomposed polygonal approximation of three English letters A, E, and Y. (b) Primary graphs for (a). Smaller letters and numbers specify node indices; they are not node labels. (c) Distorted A and its corresponding primary graph.

proposed MLSGECI algorithm is to the order of n^3b when node connectivity is high ($b > n$) in the pure graph ω, or to the order of n^4 otherwise, where n and b are the numbers of nodes and branches in ω, respectively.

H. An Application Example

Pavlidis [21] proposed an approach to shape analysis which consists of the approximation of given shapes by polygons, the decomposition of polygons into smaller convex sets, and the description of the convex sets in terms of a special kind of labeled graphs, called *primary graphs*. Each node of a primary graph represents a simple shape primitive (labeled by P) or the intersection of two primitives (labeled by N). Nodes of different labels corresponding to intersecting sets are connected by branches. Fig. 4(a) shows the decomposed polygonal approximations of three English letters A, E, and Y. The corresponding primary graphs are shown in Fig. 4(b). To describe further the nodes in a primary graph, Pavlidis [21] proposed, in addition to the labels P and N mentioned above, the following set of node attributes:

(1) number of vertices,
(2) area,
(3) direction of major axis of inertia, and
(4) coordinates of center of gravity.

Quantities specifying different types of primitive intersections are also defined to describe the branches in a primary graph. Such quantities can be regarded as branch attributes. Obviously primary graphs so defined are just a special type of attributed relational graphs. Consequently the proposed subgraph error-correcting isomorphism is applicable for the recognition of shapes represented by such graphs. For example Fig. 4(c) shows a distorted A and its corresponding primary graph, which is a subgraph of the reference primary graphs of both A and E shown in Fig. 4(b). With the help of node and branch attributes and the use of the proposed MLSGECI algorithm, the graph of the distorted A, when matched with that of the noise-free A, will yield the maximum likelihood value and so can be classified as a letter A. More detailed development of matching primary graphs for shape analysis using the proposed SGECI is left for further investigation.

V. Concluding Remarks

In this paper the structure-preserved graph deformation model, the structure-preserved error-correcting graph isomorphism, and the ordered-search algorithm for finding such isomorphisms [1], [2] are successfully and nontrivially extended to subgraphs. These subgraph ECI's can be used to match attributed relational graphs which not only cover the conventional symbolic labeled graphs but also can include numerical attributes. As a result, the proposed MLSGECI algorithm is very useful for matching patterns

represented by attributed relational graphs, which usually include structures and attributes. The error-correcting and subgraph-handling capability of the isomorphism also makes inexact matchings of occluded, partially viewed, or structurally distorted patterns possible. Further researches may be directed to applying the proposed approach to real world problems.

References

[1] W. H. Tsai and K. S. Fu, "Error-correcting isomorphisms of attributed relational graphs for pattern analysis," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 12, Dec. 1979.

[2] ____, "Error-correcting isomorphisms of attributed relational graphs for pattern classification," Tech. Rep. TR-EE 79-3, Purdue Univ., Lafayette, IN, Jan. 1979.

[3] N. J. Nilsson, *Problem Solving Methods in Artificial Intelligence.* New York: McGraw-Hill, 1971.

[4] J. R. Ullman, "An algorithm for subgraph isomorphism," *J. Ass. Comput. Mach.*, vol. 23, no. 1, Jan. 1976.

[5] S. H. Unger, "GIT—A heuristic program for testing pairs of directed line graphs for isomorphism," *Commun. Ass. Comput. Mach.*, vol. 7, no. 1, Jan. 1964.

[6] D. G. Corneil and C. C. Gotlieb, "An efficient algorithm for graph isomorphism," *J. Ass. Comput. Mach.*, vol. 17, no. 1, Jan. 1970.

[7] A. T. Berztiss, "A backtrack procedure for isomorphism of directed graphs," *J. Ass. Comput. Mach.*, vol. 20, no. 3, July 1973.

[8] E. H. Sussenguth, Jr., "A graph-theoretic algorithm for matching chemical structures," *J. Chemical Documentation*, vol. 5, pp. 36–44, 1965.

[9] O. Firschein and M. A. Fischler, "A study in descriptive representation of pictorial data," in *Proc. Int. Joint Conf. on AI*, Imperial College, London, pp. 258–269, 1971.

[10] C. Nugent *et al.*, "An experimental comparison of techniques for the assignment of facilities to locations," *Opl. Res. Q.*, vol. 16, pp. 150–173, 1968.

[11] A. Ginzberg, *Algebraic Theory of Automata.* New York: Academic, 1968.

[12] L. G. Shapiro and R. M. Haralick, "Structural descriptions and inexact matching,Tech. Rep. CS-79011-R, Virginia Polytech. Inst. and State Univ., Blacksburg VA., Nov. 1979.

[13] L. Kitchen, "Relaxation applied to matching quantitative relational structures," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-10, no. 2, Feb. 1980.

[14] D. E. Ghahraman, A. K. C. Wong, and T. Au, "Graph Optimal Monomorphism Algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-10, no. 4, Apr. 1980.

[15] S. Y. Lu and K. S. Fu, "Stochastic error-correcting syntax analysis," *IEEE Trans. Comput.*, vol. C-26, no. 12, Dec. 1977.

[16] W. H. Tsai and K. S. Fu, "A pattern deformational model and Bayes error-correcting recognition system," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 12, 1979.

[17] W. H. Tsai, "Optimal subgraph error-correcting isomorphisms for matching attributed relational graphs," in *Proc. International Computer Symposium*, Taipei, Taiwan, Republic of China, Dec. 1980.

[18] R. C. Read and D. G. Corneil, "The graph isomorphism disease," *J. Graph Theory*, vol. 1, 1977.

[19] L. Babai, P. Erdos, and S. M. Selkow, "Random graph isomorphism," *SIAM J. Comput.*, vol. 9, no. 3, Aug. 1980.

[20] R. M. Haralick and J. S. Kartus, "Arrangements, homomorphisms, and discrete relaxations," in *Proc. IEEE Conf. on Pattern Recog. and Image Processing*, Chicago, IL, May 1978.

[21] T. Pavlidis, *Structural Pattern Recognition.* New York: Springer, 1977.

Generation of a Pseudothesaurus for Information Retrieval Based on Cooccurrences and Fuzzy Set Operations

SADAAKI MIYAMOTO, TERUHISA MIYAKE, AND KAZUHIKO NAKAYAMA

Abstract—**A thesaurus in bibliographic information retrieval is a list of technical terms with relations among them, enabling generic retrieval of documents having different but related keywords. Since the construction of a thesaurus is resource consuming an automatic generation method of a thesauruslike structure is needed. A set-theoretical model of an abstract thesaurus is developed which is related to an automatic generation method based on cooccurrences of terms in the set of texts. Replacement of a basis set in the model and transformation of cooccurrence frequencies into fuzzy sets enables the transition from the abstract mathematical model to an actual procedure of automatic generation. The generated structure is called a pseudothesaurus. An algorithm to generate the pseudothesaurus from a large amount of data is developed. Moreover, two examples based on a dictionary of scientific usage and on an actual bibliographic database are given.**

Manuscript received June 1, 1982; revised August 30, 1982.

S. Miyamoto and K. Nakayama are with the Institute of Information Sciences and Electronics, University of Tsukuba, Sakura-mura, Ibaraki 305, Japan.

T. Miyake is with the Science Information Processing Center, University of Tsukuba, Sakura-mura, Ibaraki 305, Japan.

I. Introduction

RECENT DEVELOPMENT of on-line information retrieval systems has stimulated various types of data organization methods such as automatic indexing [1] and document clustering [2]. Association retrieval using a thesaurus [3] or citations [2] has been also studied. A thesaurus is a standard tool of bibliographic information retrieval, and some databases, e.g., Educational Resources Information Center (ERIC) [4], are accompanied by

0018-9472/83/0100-0062$01.00

represented by attributed relational graphs, which usually include structures and attributes. The error-correcting and subgraph-handling capability of the isomorphism also makes inexact matchings of occluded, partially viewed, or structurally distorted patterns possible. Further researches may be directed to applying the proposed approach to real world problems.

References

[1] W. H. Tsai and K. S. Fu, "Error-correcting isomorphisms of attributed relational graphs for pattern analysis," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 12, Dec. 1979.

[2] ____, "Error-correcting isomorphisms of attributed relational graphs for pattern classification," Tech. Rep. TR-EE 79-3, Purdue Univ., Lafayette, IN, Jan. 1979.

[3] N. J. Nilsson, *Problem Solving Methods in Artificial Intelligence.* New York: McGraw-Hill, 1971.

[4] J. R. Ullman, "An algorithm for subgraph isomorphism," *J. Ass. Comput. Mach.*, vol. 23, no. 1, Jan. 1976.

[5] S. H. Unger, "GIT—A heuristic program for testing pairs of directed line graphs for isomorphism," *Commun. Ass. Comput. Mach.*, vol. 7, no. 1, Jan. 1964.

[6] D. G. Corneil and C. C. Gotlieb, "An efficient algorithm for graph isomorphism," *J. Ass. Comput. Mach.*, vol. 17, no. 1, Jan. 1970.

[7] A. T. Berztiss, "A backtrack procedure for isomorphism of directed graphs," *J. Ass. Comput. Mach.*, vol. 20, no. 3, July 1973.

[8] E. H. Sussenguth, Jr., "A graph-theoretic algorithm for matching chemical structures," *J. Chemical Documentation*, vol. 5, pp. 36–44, 1965.

[9] O. Firschein and M. A. Fischler, "A study in descriptive representation of pictorial data," in *Proc. Int. Joint Conf. on AI*, Imperial College, London, pp. 258–269, 1971.

[10] C. Nugent *et al.*, "An experimental comparison of techniques for the assignment of facilities to locations," *Opl. Res. Q.*, vol. 16, pp. 150–173, 1968.

[11] A. Ginzberg, *Algebraic Theory of Automata.* New York: Academic, 1968.

[12] L. G. Shapiro and R. M. Haralick, "Structural descriptions and inexact matching,Tech. Rep. CS-79011-R, Virginia Polytech. Inst. and State Univ., Blacksburg VA., Nov. 1979.

[13] L. Kitchen, "Relaxation applied to matching quantitative relational structures," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-10, no. 2, Feb. 1980.

[14] D. E. Ghahraman, A. K. C. Wong, and T. Au, "Graph Optimal Monomorphism Algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-10, no. 4, Apr. 1980.

[15] S. Y. Lu and K. S. Fu, "Stochastic error-correcting syntax analysis," *IEEE Trans. Comput.*, vol. C-26, no. 12, Dec. 1977.

[16] W. H. Tsai and K. S. Fu, "A pattern deformational model and Bayes error-correcting recognition system," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 12, 1979.

[17] W. H. Tsai, "Optimal subgraph error-correcting isomorphisms for matching attributed relational graphs," in *Proc. International Computer Symposium*, Taipei, Taiwan, Republic of China, Dec. 1980.

[18] R. C. Read and D. G. Corneil, "The graph isomorphism disease," *J. Graph Theory*, vol. 1, 1977.

[19] L. Babai, P. Erdos, and S. M. Selkow, "Random graph isomorphism," *SIAM J. Comput.*, vol. 9, no. 3, Aug. 1980.

[20] R. M. Haralick and J. S. Kartus, "Arrangements, homomorphisms, and discrete relaxations," in *Proc. IEEE Conf. on Pattern Recog. and Image Processing*, Chicago, IL, May 1978.

[21] T. Pavlidis, *Structural Pattern Recognition.* New York: Springer, 1977.

Generation of a Pseudothesaurus for Information Retrieval Based on Cooccurrences and Fuzzy Set Operations

SADAAKI MIYAMOTO, TERUHISA MIYAKE, AND KAZUHIKO NAKAYAMA

Abstract—**A thesaurus in bibliographic information retrieval is a list of technical terms with relations among them, enabling generic retrieval of documents having different but related keywords. Since the construction of a thesaurus is resource consuming an automatic generation method of a thesauruslike structure is needed. A set-theoretical model of an abstract thesaurus is developed which is related to an automatic generation method based on cooccurrences of terms in the set of texts. Replacement of a basis set in the model and transformation of cooccurrence frequencies into fuzzy sets enables the transition from the abstract mathematical model to an actual procedure of automatic generation. The generated structure is called a pseudothesaurus. An algorithm to generate the pseudothesaurus from a large amount of data is developed. Moreover, two examples based on a dictionary of scientific usage and on an actual bibliographic database are given.**

Manuscript received June 1, 1982; revised August 30, 1982.

S. Miyamoto and K. Nakayama are with the Institute of Information Sciences and Electronics, University of Tsukuba, Sakura-mura, Ibaraki 305, Japan.

T. Miyake is with the Science Information Processing Center, University of Tsukuba, Sakura-mura, Ibaraki 305, Japan.

I. Introduction

RECENT DEVELOPMENT of on-line information retrieval systems has stimulated various types of data organization methods such as automatic indexing [1] and document clustering [2]. Association retrieval using a thesaurus [3] or citations [2] has been also studied. A thesaurus is a standard tool of bibliographic information retrieval, and some databases, e.g., Educational Resources Information Center (ERIC) [4], are accompanied by

0018-9472/83/0100-0062$01.00