Computing 67, 209–224 (2001)

Computing

Weighted Mean of a Pair of Graphs

H. Bunke and **S. Günter,** Bern

Received April 9, 2001

Abstract

Graph matching and graph edit distance are fundamental concepts in structural pattern recognition. In this paper, the weighted mean of a pair of graphs is introduced. Given two graphs, G and G', with $d(G, G')$ being the edit distance of G and G', the weighted mean of G and G' is a graph G'' that has edit distances $d(G, G'')$ and $d(G'', G')$ to G and G', respectively, such that $d(G, G'') + d(G'', G') = d(G, G')$. We'll show formal properties of the weighted mean, describe a procedure for its computation, and give examples.

AMS Subject Classifications: 05c99, 68R10, 68T10.

Key Words: Graph matching, graph edit distance, weighted mean, generalized median, self organizing map.

1. Introduction

Graph matching is a fundamental technique in structural pattern recognition. Given a pair of graphs representing an object each, the task considered in graph matching is to find an optimal assignment of the nodes and edges of one graph to the nodes and edges of the other graph, where the term *optimal* means that errors and distortions in the underlying graph representations are to be corrected. One approach to error correction is based on graph edit operations. These edit operations typically include the insertion, deletion, and substitution of nodes and edges. In order to make the approach more general, a cost is usually assigned to each edit operation. The costs of edit operations are application dependent and must be chosen based on knowledge of the considered domain. Typically, the more likely a particular type of error is to occur, the lower is the cost of the corresponding edit operation. Given a pair of graphs and a set of edit operations together with their costs, the edit distance of the two graphs is defined as the minimum cost taken over all sequences of edit operations that transform one of the given graphs into the other [1–6]. It is a generalization of the well-known concept of string edit distance [7] from the domain of strings of symbols to the domain of graphs. Graph edit distance provides both a similarity measure for two given graphs and an error correcting correspondence between their nodes and edges. Applications of graph matching include recognition of graphical symbols [8], character recognition [9, 10], shape analysis [11, 12], three-dimensional object recognition [13], video and image database indexing [14, 15] and protein determination [16].

Recently, the generalized median of a set of graphs was introduced [17]. It is useful for applications where a set of (similar) graphs is to be represented by a single prototype graph. The generalized median of a set of graphs is defined as a graph that has the minimum average edit distance to the members of a given set. In [17] a genetic algorithm for generalized median computation was introduced. An application of the generalized median to the learning of prototypical symbols in graphics recognition was described in [18]. The generalized median of a set of graphs is an extension of a similar concept from the domain of strings, which was addressed in a series of recent papers [19–24].

In the present paper we propose a modification of the generalized median of a set of graphs for the case of just a pair of graphs, which will be called the *weighted mean* of a pair of graphs. Consider two points in the n-dimensional real space, $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{R}^n$, where $n \geq 1$. Their weighted mean can be defined as a point $\boldsymbol{z}$ such that

$$\boldsymbol{z} = \gamma \boldsymbol{x} + (1 - \gamma)\boldsymbol{y}, \quad 0 \leq \gamma \leq 1 \tag{1}$$

Clearly, if $\gamma = \frac{1}{2}$ then $\boldsymbol{z}$ is the (normal) mean of $\boldsymbol{x}$ and $\boldsymbol{y}$. If $\boldsymbol{z}$ is defined according to Eq. (1) then $\boldsymbol{z} - \boldsymbol{x} = (1 - \gamma)(\boldsymbol{y} - \boldsymbol{x})$ and $\boldsymbol{y} - \boldsymbol{z} = \gamma(\boldsymbol{y} - \boldsymbol{x})$. In other words, $\boldsymbol{z}$ is a point on the line segment in n dimensions that connects $\boldsymbol{x}$ and $\boldsymbol{y}$, and the distance between $\boldsymbol{z}$ and both $\boldsymbol{x}$ and $\boldsymbol{y}$ is controlled via the parameter γ.

In this paper we describe a concept for the domain of graphs, which resembles the weighted mean as described by Eq. (1). Given two graphs G and G' and a number γ, the weighted mean of G and G' is defined as a graph G'', for which $d(G, G'') = (1 - \gamma)d(G, G')$ and $d(G'', G') = \gamma d(G, G')$ holds, where $0 \leq \gamma \leq 1$. We also introduce a procedure for computing weighted means, describe some properties of weighted mean, and discuss potential applications.

In the next section, we will introduce our basic notation. Then in Section 3 it will be shown how the weighted mean of two graphs can be computed based on any of the known algorithms for graph edit distance computation. Application examples of the weighted mean will be given in Section 4, and conclusions drawn in Section 5.

2. Definitions and Notations

Some of the following definitions are taken from [5]. The algorithms considered in this paper work on labeled graphs. Let L_V and L_E denote the set of node and edge labels, respectively.

Definition 1. A *graph* G is a 4-tuple $G = (V, E, \mu, \nu)$, where

- V is the set of nodes
- $E \subseteq V \times V$ is the set of edges
- $\mu : V \rightarrow L_V$ is a function assigning labels to the nodes
- $\nu : E \rightarrow L_E$ is a function assigning labels to the edges $\square$

In this definition, the edges are directed, i.e. there is an edge form v_1 to v_2 if $(v_1, v_2) \in E$. For graphs with undirected edges, we require $(v_2, v_1) \in E$ for any edge $(v_1, v_2) \in E$. A node $v \in V$ is called *isolated* if there exists no node $v' \in V, v' \neq v$, such that $(v, v') \in E$ or $(v', v) \in E$.

Definition 2. A bijective function $f : V \to V'$ is a *graph isomorphism* from a graph $G = (V, E, \mu, \nu)$ to a graph $G' = (V', E', \mu', \nu')$ if:

- $\mu(v) = \mu'(f(v))$ for all $v \in V$
- for any edge $e = (v_1, v_2) \in E$ there exists an edge $e' = (f(v_1), f(v_2)) \in E'$ such that $\nu(e) = \nu'(e')$, and for any $e' = (v_1', v_2') \in E'$ there exists an edge $e = (f^{-1}(v_1'), f^{-1}(v_2')) \in E$ such that $\nu'(e') = \nu(e)$ □

Graph isomorphism is a useful concept to find out if two objects are the same, up to invariance properties inherent to the underlying graph representation. However, real world objects are usually affected by noise such that the graph representations of identical objects may not be isomorphic any longer. In order to model graph distortions, a set of edit operations are introduced.

Definition 3. Given a graph $G = (V, E, \nu, \mu)$, a *graph edit operation* δ on G is any of the following:

1. $\mu(v) \to l$, $v \in V$, $l \in L_V$: substituting the label $\mu(v)$ of node v by l (for the correction of node label distortions)
2. $\nu(e) \to l'$, $e \in E$, $l' \in L_E$: substituting the label $\nu(e)$ of edge e by l' (for the correction of edge label distortions)
3. $v \to \$$, $v \in V$: deleting the node v from G (for the correction of missing nodes); here it is required that $v \in V$ is an isolated node
4. $\$ \to v, v \notin V$: inserting a (new) node in G with label $l \in L_V$ (for the correction of extraneous nodes)
5. $e \to \$$, $e \in E$ deleting the edge e from G (for the correction of missing edges)
6. $\$ \to e = (v_1, v_2), e \notin E, v_1, v_2 \in V$: inserting a (new) edge in G with label $l \in L_E$ between two existing nodes v_1, v_2 of G (for the correction of extraneous edges) □

In order to completely specify the insertion of a node v (an edge e), not only the node (edge) to be inserted, but also the corresponding label l must be given. Hence, the notation $\$ \to (v, l)$ and $\$ \to (e, l)$ may be preferable over $\$ \to v$ and $\$ \to e$, respectively, for certain applications. In the context of the present paper, however, the labels of inserted nodes and edges are not important and will not be explicitly included in the notation.

The six edit operations in Definition 3 are powerful enough to transform any graph G into any other graph G'. In order to model the fact that certain distortions are more likely than others, each graph edit operation δ is assigned a certain

cost $C(\delta)$ which is a non-negative real number. The costs of the graph edit operations are strongly application dependent and must be defined on the basis of heuristic knowledge.

Definition 4. Given a graph $G = (V, E, \mu, \nu)$ and an edit operation δ, the edited graph, $\delta(G)$, is a graph $\delta(G) = (V_\delta, E_\delta, \mu_\delta, \nu_\delta)$ with

1. $V_\delta = \begin{cases} V \cup \{v\} & \text{if } \delta = (\$ \to v) \\ V - \{v\} & \text{if } \delta = (v \to \$) \\ V & \text{otherwise} \end{cases}$

2. $E_\delta = \begin{cases} E \cup \{e\} & \text{if } \delta = (\$ \to e) \\ E - \{e\} & \text{if } \delta = (e \to \$) \\ E \cap (V_\delta \times V_\delta) & \text{otherwise} \end{cases}$

3. $\mu_\delta(v) = \begin{cases} l & \text{if } \delta = (\mu(v) \to l) \text{ or } \delta = (\$ \to v) \\ \mu(v) & \text{otherwise} \end{cases}$

4. $\nu_\delta(e) = \begin{cases} l & \text{if } \delta = (\nu(e) \to l) \text{ or } \delta = (\$ \to e) \\ \nu(e) & \text{otherwise} \end{cases}$ □

Definition 5. Given a graph $G = (V, E, \mu, \nu)$ and a sequence of edit operations $\Delta = (\delta_1, \delta_2, \ldots, \delta_k)$, the edited graph, $\Delta(G)$, is a graph

$$\Delta(G) = \delta_k(\ldots \delta_2(\delta_1(G))\ldots)$$

The total cost of the transformation of G into $\Delta(G)$ is given by $C(\Delta) = \sum_{i=1}^{k} C(\delta_i)$. □

Edit operations are used to transform a given graph G into another graph G'. According to Definition 3, the substitution of a node label $\mu(v)$ (or an edge label $\nu(e)$) requires that v (or e) exists in G. Conversely, the insertion of a node v (or an edge e) requires that v (or e) doesn't yet exist in G. More severe constraints are imposed on edge insertions and node deletions. The insertion of an edge (u, v) is possible only if both incident nodes, u and v, exist in G. Node deletions are defined for isolated nodes only. If a non-isolated node u is to be deleted from a graph, then all its incident edges (u, v) and (v, u) need to be deleted prior to the deletion of u. Similarly, if an edge (u, v) is to be inserted in a graph, but u (or v) doesn't exist, then u (or v) must be inserted prior to the insertion of (u, v). As a consequence of these constraints, $\Delta(G)$ may not exist for certain graphs G and certain sequences Δ. In the remainder of this paper we'll consider, for any graph G, only such sequences of edit operations, Δ, for which $\Delta(G)$ exist.

A subsequence Δ' of a sequence of edit operations, $\Delta = \delta_1, \ldots, \delta_k$, is obtained from Δ by deleting i edit operations where $0 \leq i \leq k$. Only such subsequences Δ' will be considered, for which $\Delta'(G)$ is defined. For example, if δ_i and δ_j denote the insertion of node u and v, respectively, and δ_k the insertion of edge (u, v), where $i < j < k$, then dropping δ_i or δ_j from Δ without dropping δ_k will result in a sequence Δ' for which $\Delta'(G)$ is undefined. Similarly, if $\delta_{i_1}, \ldots, \delta_{i_n}$ denote the deletion of all edges

incident to node u, and δ_j denotes the deletion of u, where $i_1, \ldots, i_n < j$, then dropping any of the d_{i_l} without dropping d_j will result in a sequence Δ' such that $\Delta'(G)$ is undefined. Next we combine the concepts of edited graph and graph isomorphism into the concept of error correcting graph isomorphism.

Definition 6. Given two graphs G and G', an *error-correcting graph isomorphism* (*ecgi*) φ from G to G' is a 2-tuple $\varphi = (\Delta, f_\Delta)$ where Δ is a sequence of edit operations and f_Δ is a graph isomorphism from $\Delta(G)$ to G'.

The cost of an *ecgi* φ is the cost of the edit operations in Δ, i.e. $C(\varphi) = C(\Delta)$. It is easy to see that there is usually more than one sequence of edit operations Δ such that there exists an isomorphism f_Δ from $\Delta(G)$ to G' and, consequently, there is usually more than one *ecgi* from G to G'. For the distance measure defined in the following, we are particularly interested in the *ecgi* with minimum cost.

Definition 7. Let G and G' be two graphs. The *graph edit distance* from G to G', $d(G, G')$, is given by the minimum cost taken over all *ecgi*'s φ from G to G':

$$d(G, G') = \min_{\Delta}\{C(\Delta) | \varphi = (\Delta, f_\Delta) \text{ is an } ecgi \text{ from } G \text{ to } G'\} \quad \square$$

An *ecgi* $\varphi = (\Delta, f_\Delta)$ with $d(G, G') = C(\Delta)$ is also called an *optimal ecgi* from G to G'.

Lemma 1. Let G, G' and G'' be graphs. Then

$$d(G, G'') \leq d(G, G') + d(G', G'') \tag{2}$$

Proof: Let $\varphi = (\Delta, f_\Delta), \varphi' = (\Delta', f'_\Delta)$ and $\varphi'' = (\Delta'', f''_\Delta)$ be optimal *ecgi*'s from G to G'', G to G', and G' to G'', respectively. Consider the sequence $\bar{\Delta} = (\Delta', \Delta'')$, i.e. the concatenation of Δ' and Δ''. Clearly $\bar{\Delta}$ is a sequence of edit operations that first edit G into a graph that is isomorphic to G', and then edit this graph into a graph that is isomorphic to G''. Hence $\bar{\Delta}$ is a sequence of edit operations that transform G into a graph that is isomorphic to G''. Let $f_{\bar{\Delta}}$ denote the corresponding graph isomorphism from $\bar{\Delta}(G)$ to G''. Clearly, $\bar{\varphi} = (\bar{\Delta}, f_{\bar{\Delta}})$ is an *ecgi* from G to G'', which has cost $C(\bar{\Delta}) = C(\Delta') + C(\Delta'')$. Because of the optimality of φ, we conclude $C(\Delta) \leq C(\Delta') + C(\Delta'')$, which implies the assertion. $\square$

In the remainder of this paper we'll assume that

$$C(\delta) \leq C(\delta') + C(\delta'') \tag{3}$$

for any three edit operations, where δ' followed by δ'' has the same effect as δ, i.e. $\delta'(\delta''(G)) = \delta(G)$. This is not a real restriction because in the computation of $d(G, G')$ we are searching for a sequence of edit operations with minimum cost.

Hence if Eq. (3) is not satisfied by any special triple δ, δ' and δ'', we can always change the cost such that $C(\delta) = C(\delta') + C(\delta'')$ without affecting $d(G, G')$.

As already mentioned in the introduction, many algorithms are known from the literature for the computation of $d(G, G')$. Any of those can be used in conjunction with the methods discussed in the remainder of this paper.

3. Weighted Mean Graph Computation

In this section the weighted mean of a pair of graphs will be formally introduced, and a procedure for its computation developed.

Definition 8. Let G and G' be graphs

a) The *mean* of G and G' is a graph G'' such that

$$d(G, G'') = d(G'', G') \quad \text{and} \tag{4}$$

$$d(G, G') = d(G, G'') + d(G'', G') \tag{5}$$

b) The *weighted mean* of G and G' is a graph G'' such that

$$d(G, G'') = \alpha \quad \text{and} \tag{6}$$

$$d(G, G') = \alpha + d(G'', G') \tag{7}$$

where $0 \leq \alpha \leq d(G, G')$ is a constant. □

According to this definition, a mean G'' has the same edit distance to G and G' and is – intuitively speaking – a graph 'centered between' G and G'. Eqs. (4) and (5) are special cases of Eqs. (6) and (7) if $\alpha = d(G'', G')$ or, equivalently, $\alpha = \frac{1}{2} \cdot d(G, G')$. In the introduction it was stated that for the weighted mean G'' of G and G' the equations $d(G, G'') = (1 - \gamma) d(G, G')$ and $d(G'', G') = \gamma d(G, G')$ hold, where $0 \leq \gamma \leq 1$. Obviously, this is equivalent to Eqs. (6) and (7) if we let $\alpha = (1 - \gamma) d(G, G')$. In the rest of the paper we will always use Eqs. (6) and (7).

It is to be noted that the weighted mean of two graphs is usually not unique. Consider, for example, the graphs G and G' shown in Fig. 1. If we assign a cost equal to one to each of the edit operations of Def. 3, then $d(G, G') = 4$ (resulting from two node and two edge insertions). Let $\alpha = 2$. Then both G_1 and G_2 in Fig. 1 are a weighted mean, as $d(G, G_1) = d(G, G_2) = d(G_1, G') = d(G_2, G') = 2$.

Next we turn to the computation of weighted mean. Consider graphs G and G' and let δ be an edit operation of an optimal *ecgi* $\varphi = (\Delta, f_\Delta)$ for computing $d(G, G')$. If we apply δ to G then we obtain a new graph G''. Obviously, $d(G, G'') = C(\delta)$ as δ transforms G into G'', and any other combination of edit operations that also transform G into G'' must have a cost $c' \geq C(\delta)$ because of Eq. (3).

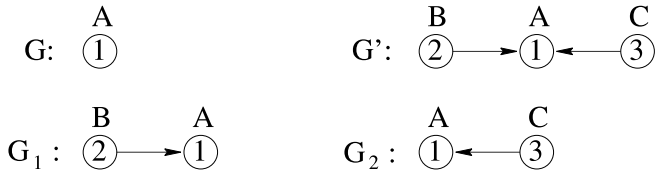

Figure 1. Four graphs, G, G', G_1 and G_2. The numbers inside the circles are the node identifiers, i.e., they denote elements of V, while the letters next to the circles are node labels. There are no edge labels in this example

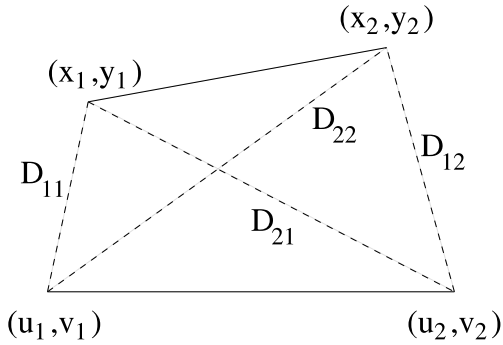

Figure 2. Graphical illustration of substitution cost $D_1 = D_{11} + D_{12}$, $D_2 = D_{21} + D_{22}$ (see text)

Now consider the edit distance $d(G'', G')$ between G'' and G'. Clearly, $d(G'', G')$ can't be larger than $d(G, G') - C(\delta)$ because if we take the sequence Δ and drop edit operation δ we get a sequence of edit operations that transforms G'' into G', and this sequence has cost $d(G, G') - C(\delta)$. But there could exist another sequence of edit operations that also transforms G'' into G', and has a cost smaller than $d(G, G') - C(\delta)$. Actually, it is easy to show that such a sequence does not exist. Assume that $d(G'', G') < d(G, G') - C(\delta)$. Because $d(G, G'') = C(\delta)$ it follows that $d(G'', G') < d(G, G') - d(G, G'')$. But this is a contradiction to Eq. (2). Hence we have proven the following lemma.

Lemma 2. Let G and G' be graphs, $\varphi = (\Delta, f_\Delta)$ an optimal *ecgi* from G to G', δ an edit operation of Δ and $\alpha = C(\delta)$. Then we can construct a graph G'' for which Eqs. (6) and (7) hold. □

Next we show an extension of this lemma to the case where we apply not a single, but a whole sequence of edit operations to derive graph G''.

Lemma 3. Let G, G' and $\varphi = (\Delta, f_\Delta)$ be the same as in Lemma 2 and Δ' a subsequence of Δ. Furthermore let $\alpha = \sum_{\delta \in \Delta'} C(\delta)$. Then we can construct a graph G'' for which Eqs. (6) and (7) hold.

Proof: If we apply all edit operations of Δ' to G, we obtain a graph G'' such that $d(G, G'') \leq \alpha$, because Δ' has cost α and transforms G into G''. Furthermore,

$d(G'', G') \leq d(G, G') - \alpha$ because if we drop all edit operations belonging to Δ' from sequence Δ, we get a sequence of edit operations with cost $d(G, G') - \alpha$, and this sequence transforms G'' into G'. There cannot be any cheaper sequence for the transformation of G'' into G', because from $d(G'', G') \leq d(G, G') - \alpha$ and $d(G, G'') \leq \alpha$ it follows that $d(G'', G') \leq d(G, G') - d(G, G'')$. But because of Eq. (2) it must be $d(G'', G') = d(G, G') - d(G, G'')$. Hence $d(G, G'') = \alpha$ and $d(G'', G') = d(G, G') - \alpha$. □

Lemma 3 suggests the following computational procedure. To compute a weighted mean of two graphs G and G', we first compute their edit distance $d(G, G')$. From the optimal edit sequence Δ, we may select any subsequence Δ' of edit operations and apply them to G. This results in a graph G'', for which Eqs. (6) and (7) hold with α being equal to the cost of the edit operations of subsequence Δ'.

Now we turn to an extension of Lemma 3 to the case of decomposable edit operations. We call an edit operation δ *decomposable* into δ_1 and δ_2 if there exist edit operations δ_1 and δ_2 such that the application of δ_1 followed by δ_2 always results in the same graph as the application of δ, i.e. $\delta_2(\delta_1(G)) = \delta(G)$ for any graph G, and $C(\delta) = C(\delta_1) + C(\delta_2)$. We write $\delta = \delta_1 \circ \delta_2$ if δ is decomposable into δ_1 and δ_2. If $\delta = \delta_1 \circ \delta_2$ then δ_1 is called a *partial realization* of δ. If we allow identical node substitutions $\mu(v) \rightarrow l$ with $\mu(v) = l$ and identical edge substitutions $\nu(e) \rightarrow l'$ with $\nu(e) = l'$ and set the cost of identical substitutions equal to zero, then any edit operation is in fact decomposable. A decomposition $\delta = \delta_1 \circ \delta_2$ where either δ_1 or δ_2 is an identical substitution is called a *trivial decomposition*. Decomposable edit operations including non-trivial decompositions are useful, for example, in domains where node and edge labels represent numerical quantities, such as lengths of line segments or distances of points in the image plane.

Let G and G' be graphs and δ be an edit operation of an optimal sequence Δ for computing $d(G, G')$. Let δ be decomposable into δ_1 and δ_2, i.e. $\delta = \delta_1 \circ \delta_2$ and $C(\delta) = C(\delta_1) + C(\delta_2)$. If we apply edit operation δ_1 to G, we obtain a graph G''. Clearly, $d(G, G'') = C(\delta_1)$. To edit G'' into G', we apply all edit operations of Δ to G'', except for δ, which is replaced by δ_2. Hence $d(G'', G') \leq d(G, G') - C(\delta) + C(\delta_2)$. Because $C(\delta) = C(\delta_1) + C(\delta_2)$, we conclude $d(G'', G') \leq d(G, G') - C(\delta_1) = d(G, G') - d(G, G'')$. But because of Eq. (2) it must be actually $d(G'', G') = d(G, G') - d(G, G'')$. This result can be summarized as follows.

Lemma 4. Let G, G' and $\varphi = (\Delta, f_\Delta)$ be the same as in Lemma 2. Furthermore let $\delta = \delta_1 \circ \delta_2$ be a decomposable edit operation of Δ and $\alpha = C(\delta_1)$. Then we can construct a graph G'' for which Eqs. (6) and (7) hold. □

Moreover, we can extend this result in the following way.

Lemma 5. Let G, G' and $\varphi = (\Delta, f_\Delta)$ be the same as in Lemma 2. Furthermore, let $\Delta' = (\delta_1, \ldots, \delta_l)$ be a subsequence of Δ, where each δ_i is decomposable, i.e. $\delta_i = \delta_i' \circ \delta_i''$, $1 \leq i \leq l$, and $\alpha = \sum_{i=1}^{l} C(\delta_i')$. Then we can construct a graph G'' for which Eqs. (6) and (7) hold.

Proof: The proof is similar to that of Lemma 3 and 4. If we apply all edit operations δ_i' to G, we get a graph G'' such that $d(G, G'') \leq \alpha$. Furthermore, $d(G'', G') \leq d(G, G') - \sum_{i=1}^{l} C(\delta_i) + \sum_{i=1}^{l} C(\delta_i'') = d(G, G') - \sum_{i=1}^{l} C(\delta_i') = d(G, G') - \alpha \leq d(G, G') - d(G, G'')$. Because of Eq. (2) we actually conclude $d(G, G'') = \alpha$ and $d(G'', G') = d(G, G') - \alpha$. □

Lemma 5 is an extension of Lemma 3 in the following sense. From an optimal sequence of edit operations for the computation of $d(G, G')$, we may select any subsequence Δ'. Next each edit operation is replaced by one of its partial realizations (notice that such a partial realization may be identical to the considered edit operation or it may be an identical substitution). The resulting sequence of edit operations is applied to G, yielding a graph G'' for which Eqs. (6) and (7) hold, with α being equal to the cost of the applied edit operations. Because the sequence of edit operations applied to G may include identical substitutions, Lemma 3 is a special case of Lemma 5. Also Lemmas 2 and 4 are special cases, where sequence Δ' is of length one. In Section 2 it was pointed out that for any given graph G only sequences of edit operations, Δ, will be considered for which $\Delta(G)$ exists. This condition applies in particular to any sequence Δ' that is derived from another sequence Δ by replacing edit operations by a partial realization. As an example, assume that the cost of deleting an edge $e = (u, v)$ with label l is equal to the cost of substituting l by l' and subsequently deleting e with label l'. Then in a sequence Δ that contains the deletion of e with label l, followed by the deletion of u or v, we must not replace the deletion of l by the substitution of l by l', because this would violate the condition that only isolated nodes can be deleted.

Lemma 5 implies a procedure for constructing, for given G and G', graphs G'' for which Eqs. (6) and (7) hold true. We can ask, conversely, if there is any graph G'' fulfilling Eqs. (6) and (7) that cannot be constructed according to Lemma 5. The answer to that question is negative, as we will show in the following.

Consider graphs G, G' and G'' such that $d(G, G') = d(G, G'') + d(G'', G')$. Let Δ_1 be an optimal sequence of edit operations for computing $d(G, G'')$. Similarly, let Δ_2 be an optimal sequence of edit operations for computing $d(G'', G')$. Then $C(\Delta_1) = d(G, G'')$ and $C(\Delta_2) = d(G'', G')$. If we concatenate sequences Δ_1 and Δ_2 we get $\Delta = (\Delta_1, \Delta_2)$. Obviously, Δ is a sequence of edit operations that transform G into G' with cost $C(\Delta) = C(\Delta_1) + C(\Delta_2) = d(G, G'') + d(G'', G')$. Because $d(G, G') = d(G, G'') + d(G'', G')$ we conclude $C(\Delta) = d(G, G')$. Hence Δ is an optimal edit sequence for computing $d(G, G')$. We notice that Δ_1 is a subsequence of Δ in the sense of Lemma 5. Hence any graph G'' that fulfills $d(G, G') = d(G, G'') + d(G'', G')$ can be constructed according to the procedure implied by Lemma 5. This result, together with Lemmas 2 to 5 can be summarized as follows:

Theorem 1. Let G, G' and G'' be graphs and Δ an optimal sequence of edit operations for computing $d(G, G')$. Then $d(G, G') = d(G, G'') + d(G'', G')$ if and only if G'' is obtained from G by the following procedure:

1. Select a subsequence Δ' of Δ.
2. Replace each edit operation of Δ' by a partial realization (some or all of these partial realizations may be identical to the original edit operations or may be identical substitutions), resulting in sequence Δ''.
3. Apply the sequence of edit operations Δ'' obtained in step 2 to G, resulting in graph G''.

For any graph G'' obtained under this procedure, $\alpha = C(\Delta'') = d(G, G'')$, where α corresponds to Eqs. (6) and (7). □

As a final remark in this section we want to point out that any weighted mean G'' of graphs G and G' is a generalized median in the sense of [19]. The generalized median of a set of graphs, S, is a graph that has the minimal average edit distance to all members of S. Consider the case $S = \{G, G'\}$. Clearly, any weighted mean G'' minimizes the average edit distance to G and G', because $d(G, G'') + d(G'', G') = d(G, G')$ and there cannot be any other graph $\bar{G}$ with $d(G, \bar{G}) + d(\bar{G}, G') < d(G, G')$ because of Eq. (2). Notice in particular that for $\alpha = 0$ and $\alpha = d(G, G')$ (see Eqs. (6) and (7)) we obtain $G'' = G$ and $G'' = G'$, respectively, which means that both G and G' are a generalized median of S. Hence, we can conclude that for any pair of graphs, G and G', there exists always a generalized median, and any generalized median G'' is a weighted mean with $\alpha = d(G, G'')$. Notice that these properties hold true for both finite and infinite sets of node and edge labels.

4. Application Examples

In this section we will show a few examples of weighted mean graph computation. In the examples, graph representations of drawings consisting of straight line segments will be considered. There is no standard way of transforming a line drawing into a graph representation. As a matter of fact, many different representations have been described in the literature [2, 5, 26–28]. The suitability of a particular graph representation depends on a number of factors, for example, on desired invariance properties of the pattern representation, or the type of expected errors. In the examples shown in this section two different graph representation, called R1 and R2, will be used.

Under representation R1, each straight line segment in a drawing is represented by a node in the corresponding graph. Each node has the pair of coordinates of the two endpoints of the corresponding line segment, $((x_1, y_1), (x_2, y_2))$, as label. The order of the endpoints is arbitrary. As all information about a line drawing is captured in the nodes and their labels, no edges are needed in representation R1. Consequently, only edit operations on nodes have to be considered. The costs of these operations are defined as follows:

- node insertion and deletion cost

$$C(((x_1, y_1), (x_2, y_2)) \to \varepsilon) = C(\varepsilon \to ((x_1, y_1), (x_2, y_2))) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

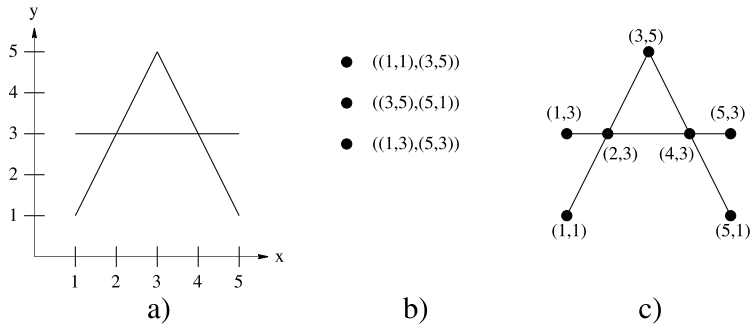

Figure 3. An example of graph representations R1 and R2; **a** original figure; **b** representation R1; **c** representation R2

- node substitution cost

$$C(((x_1, y_1), (x_2, y_2)) \rightarrow ((u_1, v_1), (u_2, v_2))) = \beta \cdot \min(D_1, D_2) \text{ where}$$

$$D_1 = D_{11} + D_{12} = \sqrt{(x_1 - u_1)^2 + (y_1 - v_1)^2} + \sqrt{(x_2 - u_2)^2 + (y_2 - u_2)^2}$$

$$D_2 = D_{21} + D_{22} = \sqrt{(x_1 - u_2)^2 + (y_1 - v_2)^2} + \sqrt{(x_2 - u_1)^2 + (y_2 - u_1)^2}$$

Thus the cost of a node insertion and deletion is equal to the length of the affected line segment. For a graphical illustration of the substitution cost see Fig. 2. The cost of a node substitution is proportional to the sums of the Euclidean distances of the two pairs of endpoints of the affected line segments. As the order of the two endpoints of a line segment is arbitrary, the minimum of D_1 and D_2 is taken. The constant β is a weighting factor that controls the node substitution cost relative to the cost of a node insertion or deletion.

An example of graph representation R1 is shown in Fig. 3[1]. Under graph representation R1 and the cost of the edit operations given above, an optimal *ecgi* is an assignment of the line segments represented by one graph to the line segments represented by the other graph in such a way that the length, the rotation angle, and the geometric adjacency of the end points is maintained as closely as possible. However, it will not be attempted to maintain topological relationships between corresponding pairs of line segments, for example, the intersection of a pair of line segments, or the coincidence of endpoints.

[1] The fact that graphs under representation R1 do not have edges does not result in a significant simplification of the task of finding an optimal *ecgi*. A minor simplification comes from the fact that no edit operations on edges need to be considered. But for finding the mapping f_Δ in the pair $\varphi = (\Delta, f_\Delta)$ still all possible assignments from any subset of nodes of the one graph to all subsets of equal size of the other graph need to be considered.

Under representation R2, a node in the graph corresponds to an endpoint of a line segment or a point of intersection between two line segments, and edges represent lines that connect two adjacent intersections or end points which each other. The label of a node represents the location of that node in the image. There are no edge labels. The cost of the edit operations under R2 are defined as follows:

- node insertion and deletion cost

$$C((x,y) \to \varepsilon) = C(\varepsilon \to (x,y)) = \beta_1$$

- node substitution cost

$$C((x_1,y_1) \to (x_2,y_2)) = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2}$$

- cost of deleting or inserting an edge e

$$C(e \to \varepsilon) = C(\varepsilon \to e) = \beta_2$$

The cost of a node substitution is equal to the Euclidean distance of the locations of the two considered nodes in the image, while the cost of node deletions and insertions, as well as edge deletions and insertions, are user defined constants, β_1 and β_2. As there are no edge labels, no edge substitutions need to be considered.

An example of the representation R2 is shown in Fig. 3c. Using graph representation R2 and the cost function introduced above, an optimal *ecgi* will be an assignment of line segments in such a way that the Euclidean distance between corresponding end points and points of intersection, as well as the topology of the two line drawings, is preserved to the maximum possible degree.

Given two line drawings and their graph representations, G and G', the algorithm described in [17] is applied to compute the edit distance $d(G,G')$. It is an optimal algorithm based on a best-first search procedure. If there are several optimal edit sequences Δ one is randomly chosen. Then for a given value of α (see Eqs. (6) and (7)), a subsequence of edit operations or their partial realizations are selected from Δ, the cost of which approximate α as closely as possible. These edit operations are then applied to G, resulting in a weighted mean according to Theorem 1.

Figure 4 shows five instances of character F. In this example, graph representation R1 has been used. The first and last character correspond to G and G' in Eqs. (6) and (7), while all other characters G_i'' correspond to G'' for increasing values of $\alpha = \frac{i}{4} \cdot d(G,G'), i = 1,2,3$. It can be clearly observed that with an increasing value of α the characters represented by the weighted mean graph, G_i'', become more similar to G'. Figure 5 shows another example using graph representation R1.

Under R1 no explicit information about the topology of a line drawing is maintained. In the computation of $d(G,G')$ the graph matching algorithm assigns a line segment l in G to a line segment l' in G' taking only spatial proximity of their end points into regard. Hence it is not attempted to preserve the connectivity of the line segments. By contrast, under R2 both topological relations and spatial

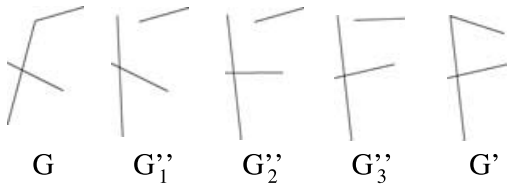

Figure 4. Sequence of weighted means for varying values of α using graph representation R1

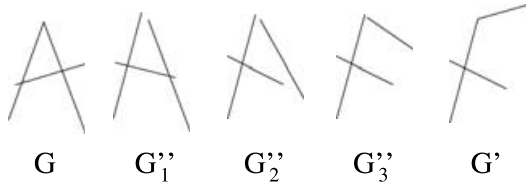

Figure 5. Sequence of weighted means for varying values of α using graph representation R1

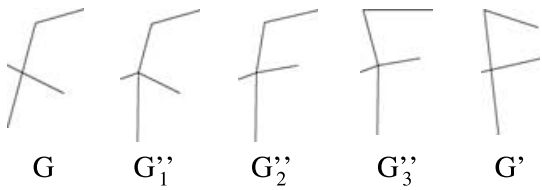

Figure 6. Sequence of weighted means for varying values of α using graph representation R2

proximity are maintained. The example of Fig. 4 using graph representation R2 is shown in Fig. 6. Intuitively speaking, in the transition from G to G' the system moves the nodes of G towards their corresponding location in G' with the edges behaving like rubber bands that adjust their length according to the location of the node.

In the examples considered until now, the line drawings corresponding to G and G' consist of the same number of strokes, which results in the number of nodes and edges in G and G' being identical. Consequently, no deletions and insertions are needed to derive any of the weighted mean graphs shown. By contrast, G and G' in Fig. 7 consist of 3 and 4 nodes, respectively. Figure 7 is based on graph representation R1. Hence one node insertion is needed for the transformation of G into G'. This insertion occurs in the step from G to G''_1. Additionally, the lower part of the vertical stroke in character F has been slightly tilted so as to make it more similar to its corresponding stroke in character E. In G''_2 the lower part of the vertical stroke in F has been completely aligned with the corresponding line segment in E. Additionally, the horizontal stroke in the middle has been aligned. In G''_3 edit operations involving 75% of the total cost of transforming G into G' have been applied to G, resulting in a shape that is identical to E up to the horizontal stroke, on top. Finally, applying all edit operations results in G'.

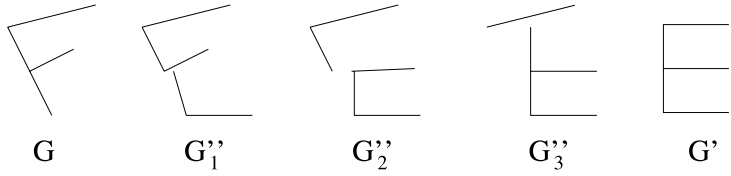

Figure 7. Sequence of weighted means for varying values of α using graph representation R1; The two graphs G and G' represent line drawings of different letters

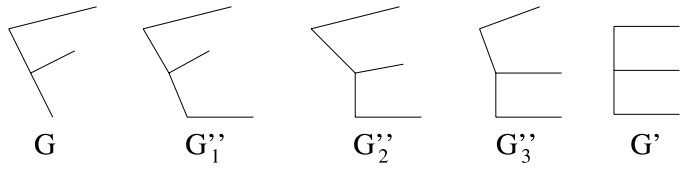

Figure 8. Sequence of weighted means for varying values of α using graph representation R2; The two graphs G and G' represent line drawings of different letters

Figure 8 shows the same example under graph representation R2. Now G consists of 5 nodes and 4 edges, while in G' we have 6 and 5 nodes and edges, respectively. Hence one node and one edge must be inserted when transforming G into G'. The node and edge insertions occur in the step that leads from G to G''_1. The changes occurring in the following sequence, G''_2, G''_3 and G', are similar to Fig. 7.

5. Conclusions

In this paper the weighted mean of a pair of graphs was introduced. The weighted mean of G and G' is a graph, G'', that has given edit distances α and $d(G, G') - \alpha$ to G and G', respectively, where $0 \leq \alpha \leq d(G, G')$. Hence a weighted mean of a pair of graphs resembles the weighted mean of a pair of numbers or vectors in the n-dimensional real space, with α controlling the degree of similarity of G to G'. A number of theoretical properties of weighted mean were studied. Also a procedure for weighted mean graph computation was given. It was shown that this procedure is correct and complete. That is, any graph G'' that is generated for a given pair of input graphs, G and G', is in fact a weighted mean of G and G' and, conversely, there does not exist any weighted mean of G and G' that cannot be generated by the given procedure.

The generalized median of a set of symbolic structures has received some attention recently. The weighted mean of a pair of graphs is a concept that is closely related to the generalized median. In fact, the set of weighted means is identical to the set of generalized medians for any two given graphs, G and G'.

From the domain of line drawing analysis, a number of practical examples of weighted mean graph were given. These examples demonstrate that weighted mean is a useful concept to synthesize patterns G'' that have certain degrees of similarity to given patterns G and G'. In the examples the intuitive notion of shape

similarity occurs to correspond well with the formal concept of graph edit distance.

As a further remark we want to point out that there are a number of well established procedures in statistical pattern recognition where a given pattern $\boldsymbol{x} \in \mathrm{R}^n$ is to be changed so as to make it more similar to another pattern $\boldsymbol{y} \in \mathrm{R}^n$. An example is self-organizing map [25], where the following formula is applied: $\boldsymbol{x} := \boldsymbol{x} + \alpha(\boldsymbol{y} - \boldsymbol{x})$. Obviously, this is very similar to weighted mean graph computation as considered in this paper, because it is an operation that changes $\mathbf{x}$ so as to make it more similar to $\mathbf{y}$. Consequently, the procedure introduced in this paper makes it possible to transfer the concept of self-organizing map from the n-dimensional real space to the domain of graphs. Further examples of procedures from statistical pattern recognition that become applicable in the domain of graphs through weighted mean graph computation include vector quantization and learning vector quantization [25].

The actual application of self-organizing feature map, vector quantization, or learning vector quantization in the domain of graphs will be subject to future research. Additional questions for future research include the generalization of weighted mean computation from two given graphs to sets consisting of $n > 2$ graphs.

The results of this paper make heavily use of the triangular in Eq. (2), which follows from the fact that edit costs are *additive* and edit distance is defined as the *minimum* cost taken over all possible transformations from G to G'. In particular, the results do not make any assumptions on the type of the underlying graphs. An interesting topic for future research may be the problem of weighted mean computation of objects in metric spaces more general than graphs.

Acknowledgement

The authors want of thank Dr. X. Jiang for continuous collaboration, stimulating discussion and useful hints on various topics related to the present paper.

References

[1] Tsai, W. H., Fu, K. S.: Error-correcting isomorphisms of attributed relational graphs for pattern recognition. IEEE Trans. Syst. Man Cybernetics *9*, 757–768 (1979).
[2] Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Rec. Lett. *1*, 245–253 (1983).
[3] Sanfeliu, A., Fu, K. S.: A distance measure between attributed relational graphs for pattern recognition. IEEE Trans. Syst. Man Cybernetics. *13*, 353–362 (1983).
[4] Wang, J. T., Zhang, K., Chirn, G.-W.: Algorithms for approximate graph matching. Information Sci. *82*, 45–74 (1995).
[5] Messmer, B., Bunke, H.: A new algorithm for error-tolerant subgraph isomorphism detection. IEEE Trans. Pattern Anal. Mach. Intell. *20*, 493–504 (1998).
[6] Myers, R., Wilson, R., Hancock, E.: Bayesian graph edit distance. IEEE Trans. Pattern Anal. Mach. Intell. *22*, 628–635 (2000).
[7] Wagner, R. A., Fischer, M. J.: The string-to-string correction problem. J. Assoc. Comput. Mach. *21*, 168–173 (1974).

[8] Lee, S. W., Kim, J. H., Groen, F. C. A.: Translation- rotation- and scale invariant recognition of hand-drawn symbols in schematic diagrams. Int. J. Pattern Rec. Art. Intell. *4*, 1–15 (1990).
[9] Lu, S. W., Ren, Y., Suen, C. Y.: Hierarchical attributed graph representation and recognition of handwritten Chinese characters. Pattern Rec. *24*, 617–632 (1991).
[10] Rocha, J., Pavlidis, T.: A shape analysis model with applications to a character recognition system. IEEE Trans. Pattern Anal. Mach. Intell. *17*, 393–404 (1994).
[11] Cantoni, V., Cinque, L., Guerra, C., Levialdi, S., Lombardi, L: 2-D object recognition by multiscale tree matching. Pattern Rec. *31*, 1443–1455 (1994).
[12] Pelillo, M., Siddiqi, K., Zucker, S.: Matching hierarchical structures using associated graphs. IEEE Trans. Pattern Anal. Mach. Intell. *21*, 1105–1120 (1999).
[13] Wong, E. K.: Model matching in robot vision by subgraph isomorphism. Pattern Rec. *25*, 287–304 (1994).
[14] Folkers, A., Samet, H., Soffer, A.: Processing pictorial queries with multiple instances using isomorphic subgraphes. Proc. 15th Int. Conf. on Pattern Recognition, pp. 51–54 (2000).
[15] Shearer, K., Bunke, H., Venkatesh, S.: Video indexing and similarity retrieval by largest common subgraph detection using decision trees. Pattern Rec. *34*, 1075–1091 (2001).
[16] Baxter, K., Glasgow, J.: Protein structure determination: combining inexact graph matching and deformable templates. Proc. Vision Interface 6, 179–186 (2000).
[17] Bunke, H., Münger, A., Jiang, X.: Combinatorial search versus genetic algorithms: a case study based on the generalized mean graph problem. Pattern Rec. Lett. *20*, 1271–1277 (1999).
[18] Jiang, X., Münger, A., Bunke, H.: Synthesis of representative graphical symbols by computing generalized median graphs. In: Graphics recognition (Chhabra, A., Dori, D., eds.), pp. 183–192. Springer LNCS 1941. Berlin Heidelberg New York Tokyo: Springer, 2000.
[19] Kohonen, T.: Median strings. Pattern Rec. Lett. *3*, 309–313 (1985).
[20] Casacuberta, F., de Antonio, M. D.: A greedy algorithm for computing approximate median strings. Proc. of National Symposium on Pattern Recognition and Image Analysis, Barcelona Spain, 1997, pp 193–198.
[21] Kruszlicz, F.: Improved greedy algorithm for computing approximate median strings. Acta Cybern. *14*, 331–339 (1999).
[22] Jiang, X., Schiffmann, L., Bunke, H.: Computation of median shapes. Proc. 4th Asian Conference on Computer Vision, Taipei, Taiwan, pp. 300–305 (2000).
[23] Martinez-Hinarejos, C., Juan, A., Casacuberta, F.: Use of median string for classification. Proc. 15th Int. Conf. on Pattern Recognition, Barcelona, Vol. 2, pp. 907–910 (2000).
[24] Bunke, H., Jiang, X., Abegglen, K., Kandel, A.: On the weighted mean of a pair of strings, submitted.
[25] Kohonen, T.: Self-organizing map. Berlin Heidelberg New York Tokyo: Springer, 1995.
[26] Suganthan, P. N., Tesh, E. K., Mital, D. P.: Pattern recognition by graph matching using Potts MFT neural networks. Pattern Rec. *28*, 997–1009 (1995).
[27] Kittler, J., Petrou, M., Christmas, W. J.: A non-iterative probabilistic method for contextual correspondence matching. Pattern Rec. *31*, 1455–1468 (1998).
[28] Huet, B., Cross, A., Hancock, E. R.: Sensitivity analysis for graph matching from large structural libraries. In: Proc 2nd IAPR-TC-15 Workshop on Graph-based Representations. Österreichische Computer Gesellschaft, pp. 89–98 (1999).

Horst Bunke
Simon Günter
Department of Computer Science
University of Bern
Neubrückstr. 10
CH-3012 Bern
Switzerland
e-mails: bunke@iam.unibe.ch
sguenter@iam.unibe.ch