A New Algorithm for Graph Matching with Application to Content-Based Image Retrieval

Adel Hlaoui and Shengrui Wang*

DMI, University de Sherbrooke
Sherbrooke (Quebec), J1K 2R1, Canada
{Hlaoui,Wang}@dmi.usherb.ca

Abstract. In this paper, we propose a new efficient algorithm for the inexact matching problem. The algorithm decomposes the matching process into *K* phases, each exploiting a different part of solution space. With most plausible parts being searched first, only a small number of phases is required in order to produce very good matching (most of them optimal). A Content-based image retrieval application using the new matching algorithm is described in the second part of this paper.

1 Introduction

With advances in the computer technologies and the advent of the Internet domain, the task of finding visual information is increasingly important and complex. Many attempts have been reported in the literature using low-level features such as colour, texture, shape and size. We are interested in the use of graph representation and graph matching [1] [2] for content-based image retrieval. The graph allows representation of image content by taking advantage of object/region features and their interrelationships. Graph matching [3] makes it possible to compute similarity between images. Given a database of images, retrieving images similar to a query image amounts to determining the similarity between graphs.

Many algorithms have been proposed for computing similarity between graphs by finding graph isomorphism or sub-graph isomorphism [4]. However, the algorithms for optimal matching are combinatorial in nature and difficult to use when the size of the graphs is large. The goal of this work is to develop a general and efficient algorithm that can be used easily to solve practical graph matching problems. The proposed algorithm is based on an application independent search strategy and can be run in a time-efficient way and, under some very general conditions, provides even optimal matching between graphs. We will show that the new algorithm can be effectively applied to content-based image retrieving. More importantly, this algorithm could help in alleviating the complexity problem in graph clustering, which is a very important step towards bridging the cap between structural pattern recognition and statistical pattern recognition [11].

---

* Dr. S. Wang is currently with School of Computer Science, University of Windsor, Windsor, Ontario, N9B 3P4, Canada

T. Caelli et al. (Eds.): SSPR&SPR 2002, LNCS 2396, pp. 291-300, 2002.

2 The New Graph Matching Algorithm

In this section, we present a new algorithm for the graph-matching problem. Given two graphs, the goal is to find the best mapping between their nodes that leads to the smallest matching error. The matching error between the two graphs is a function of the dissimilarity between each pair of matched nodes and the dissimilarity between the corresponding edges. It can be viewed as the distance between the two graphs [5]. The basic idea of the new algorithm is iterative exploration of the best possible node mappings and selection of the best mapping at each iteration phase by considering both the error caused by node matching as well as that caused by corresponding edge mapping. The underlying hypothesis of this algorithm is that a good mapping between two graphs likely match similar nodes. The advantage of this algorithm is that this iterative process often allows finding the optimal mapping within a few iterations by searching only the most plausible regions of solution space. In the first phase, the algorithm selects the best possible mapping(s) that minimize the error induced by node matching only. Of these mappings, those that also give the smallest error in terms of edge matching are retained. In the second phase, the algorithm examines the mappings that contain at least one second-best mapping between nodes and then again computes those mappings that give rise to the smallest error in terms of edge matching. This process continues through a predefined number of phases.

2.1 Algorithm Description

We suppose that distance measures associated with the basic graph edit operations have been defined; i.e. costs have already been associated with substitution of nodes and edges, deletion of nodes and edges, etc. The technique proposed here is inspired by both Ullman's [1] algorithm and the error-correcting sub-graph isomorphism procedure [4],[6],[9],[10]. The new algorithm is designed for substitution operations only. It can easily be extended to deal with deletion and insertion operations by considering some special cases. For example, matching a node to a special (non-) node can perform deletion of the node. The algorithm is designed to find a graph isomorphism when both graphs have the same number of nodes and a sub-graph isomorphism when one has fewer nodes than the other.

Given two graphs $G_1 = (V_1, E_1, \mu_1, \nu_1)$ and $G_2 = (V_2, E_2, \mu_2, \nu_2)$, a $n \times m$ matrix $P = (p_{ij})$ is introduced, where n and m are the numbers of nodes in the first and the second graph, respectively. Each element p_{ij} in P denotes the dissimilarity between node i in G_1 and node j in G_2. We also use a second $n \times m$ matrix $B = (b_{ij})$.

The first step is to initialize matrix P by setting $p_{ij} = d(\mu_1(v_i), \mu_2(v_j))$. The second step consists of initializing B by setting $b_{ij} = 0$. The third (main) step contains K phases. In the first phase $(Current\_Phase = 1)$, the elements of B corresponding to the minimum elements in each row of matrix P are set to 1, $(b_{ij} = 1)$. Then, for each possible mapping extracted from B, the algorithm computes the error induced by nodes and the error induced by edges. The mapping that gives the smallest matching

error will be recorded. In the second phase ($Current\_Phase = 2$), the algorithm will set the value to 1 those elements of B corresponding to the second-smallest elements in each row of matrix P. The algorithm will extract the mappings from matrix B that contain at least one node-to-node mapping added to matrix B at this phase. Of these mappings and the mappings obtained in the first phase, those with the smallest cost are retained. The algorithm then proceeds to the next phase, and so on.

A direct implementation of the above ideas would result in redundant extraction and testing of mappings, since any mapping extracted from matrix B at a given time will also be extracted from any subsequent matrix B. To solve this problem, a smart procedure has been designed. First, a matrix B' is introduced to contain all the possible node-to-node mappings considered by the algorithm so far. B is used as a 'temporary' matrix. At each phase (except the first), each of the n rows of B is examined successively. For each row i of B, all of the previous rows of B will contain all of the possible node-to-node mappings examined so far. The row i contains only the possible node-to-node mapping in the present phase. Finally, all of the following rows of B will contain only the possible node-to-node mappings examined in the previous phases. Such a matrix B guarantees that the mappings extracted as the algorithm progresses will never be the same and that all of the mappings that need to be extracted at each phase will indeed be extracted.

To illustrate the algorithm, we present a detailed example. Fig. 1 shows the weights attributed to nodes and edges in the input and the model graphs respectively. The first step in the proposed algorithm computes a P matrix. Each row in P represents a node in the model graph and the columns represent nodes in the input graph. The P matrix is given in Table 1. The second step of the algorithm computes the B matrix. Each element b_{ij} in this matrix is set to 1 if the corresponding p_{ij} has the smallest value in the i^{th} row of P, to 0 otherwise. At this stage, there is no possible matching. This step can be interpreted as level one or $Current\_Phase = 1$. Next the algorithm enters its second phase, exploring mappings containing at least one node-to-node matching which corresponds to the second-smallest value in a row of the matrix P. Table 4 illustrates the possible mappings extracted from the current B.

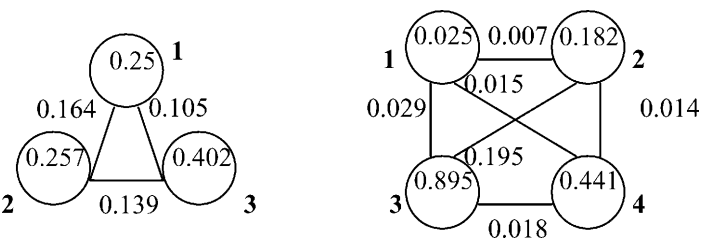


Fig. 1. Input graph and model graph

Table 1. Matrix P

0.225	0.068	0.645	0.19
0.232	0.075	0.638	0.183
0.377	0.22	0.493	0.038

Table 2. Matrix *B* (first phase)

0	1	0	0
0	1	0	0
0	0	0	1

Table 3. Matrix *B* (second phase)

1	1	0	0
0	1	0	0
0	0	0	1

Table 4. Best matching with $Current\_Phase = 2$

Mappings	Matching error
(1,1) (2,2) (3,4)	0.711

2.2 Algorithm and Complexity

Input: two attributed graphs G_1 and G_2 .
Output: matching between nodes in G_1 and G_2, from the smaller graph (e.g., G_1) to the larger (e.g., G_2)

1. **I**nitialize *P* as follows: For each p_{ij} , set $p_{ij} = d(\mu_1(v_i), \mu_2(v_j))$.
2. **I**nitialize *B* as follows: For each b_{ij} , $i = 1,...,n$ and $j = 1,...,m$, set $b_{ij} = 0$.
3. **W**hile $Current\_Phase < K$

   If $Current\_Phase = 1$, **T**hen

   For $i = 1,...,n$

   Set the value 1 to elements of *B* corresponding to the smallest value in i^{th} row of *P*;

   Call Matching_Nodes(*B*).

   Else **F**or all $i = 1,...,n$

   Set $B' = B$

   For all $j = 1,...,m$ set $b_{ij} = 0$

   Select the element with the smallest value in *P* that is not marked 1 in *B'* and set it to 1 in *B* and *B'*;

   Call Matching_Nodes(*B*);

   Set $B = B'$.

   If all the elements in *B* are marked 1, **T**hen

   Set $Current\_Phase = K$

   Else **a**dd 1 to *Current_Phase*.

Matching_Nodes(B)

For each valid mapping in *B*

1. **C**ompute the matching error induced by nodes.
2. **A**dd the error induced by the corresponding edges to the matching error.
3. **S**ave the actual matching if the matching error is minimal.

The major parameter *K* defines the number of phases to be performed in order to find the best matching. Suppose, without loss of generality, that the size of the two graphs satisfies the following condition $n = |V_1| \leq |V_2| = m$, then the worst case complexity of the new algorithm is $O(n^2 K^n)$. This is to compare with $O(n^2 m^n)$, the complexity for Ullman's algorithm [1] and the A*-based error-correcting sub-graph isomorphism algorithm [4],[6]. In general, the new algorithm reduces the number of steps in the error-correcting algorithm by the factor of about $(m/K)^n$. This can be very significant when matching large graphs. Table 5 shows a comparison with the A*-based error-correcting algorithm over 1000 pairs of graphs generated randomly. The size of each graph is between 2 and 10 nodes. The experiment was run on a Sun Ultra 60 workstation (450 MHz CPUs). From the table, one can notice that the new algorithm performs extremely well in computing the optimal matching while maintaining very low average CPU times. For instance, when using $K = 4$, the algorithm finds the optimal matching in 971 cases while using only 11 seconds in average. The A*-based algorithm needs 186 seconds in average although it guarantees to find the optimal matching. It is to be remarked that due to its complexity, the A*-based algorithm is generally not usable when the graphs to be matched have more than 10 nodes. The new algorithm does not suffer this limit. For example, matching two graphs of 11 and 30 nodes with $K = 5$ takes about 100 seconds. Details about the deduction of the complexity and about the performance of the algorithm can be found in our technical report [8]. The new algorithm does not require the use of heuristics. It can be used to find good matchings (usually optimal) in a short time. In this sense, it can be categorised in the class of approximate algorithms.

Table 5. Comparison with the error-correcting sub-graph isomorphism algorithm

Number of phases *K*	1	2	3	4	5	Error-Correcting (A*)
Optimal matchings reached by the proposed algorithm	609	827	940	971	1000	1000
Average time in seconds	2.14	3.69	6.14	11.04	16.28	186.57

3 Image Retrieval Based on the New Graph Matching Algorithm

The aim of this section is to show how graph matching contributes to image retrieval. In particular, we would like to show how the new matching algorithm could be used. For this purpose, we have generated an artificial image database so that extraction of

objects and representation of the content by a graph are simplified. Our work is divided into two parts. First, we build an image database and define a graph model to represent images. Second, we make use of the new matching algorithm to derive a retrieval algorithm for retrieving similar images.

The advantage of using a generated database is that it allows us to evaluate a retrieval algorithm in a more systematic way. We suppose that each image in the database contains regular shapes such as rectangles, squares, triangles, etc. An algorithm has been developed to build such a database. Only the number of images needs to be given by the user. The algorithm randomly generates all the other parameters. These random parameters define the number of object, the shape, color, size and position of each object in the image.

For easy manipulation of the database, only the description of the image is stored in a text file and a subroutine is created to save an image from and restore it to this text file. The description includes following variables: the numerical index of each image, the number of objects in the image, the shape of an object represented by a value between 1 and 5 (a square is represented by 1, a rectangle by 2, etc.), the size of the object; its color; its position; and its dimension.

The second step in the process is to use graphs to represent the contents of images. Each node represents an object in an image and an edge represents the relation between two objects. In our work, three features describe a node: the shape, size and color of the object. Two features describe an edge: the distance between two objects and their relative position. These features are represented, respectively, using S, Z, C, D, and RP. The values of the first three features figure in the database. The Hausdorff distance [7] is computed for D. The relative position RP is a discrete value describing the location of objects with respect to each other [8].

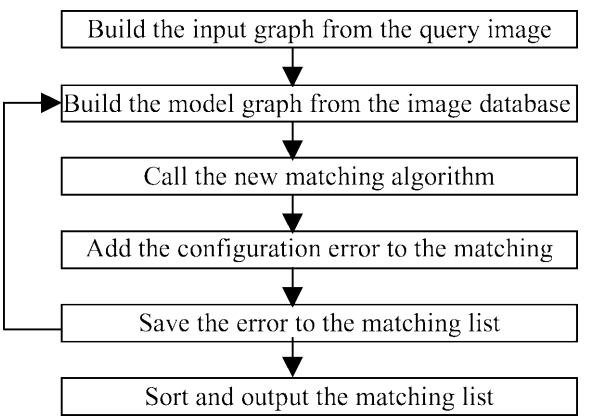


Fig. 2. The flow diagram of the retrieval algorithm

3.1 The Retrieval Algorithm

In this section, we adapt the matching algorithm described in Section 2 for retrieving images by content using graphs. Given a query image, the algorithm computes a matching error for each image in the data base, finds the best matching between the query image and any of the images in the database and extracts the similar images

from the database. Fig.2 gives the schema of the retrieval algorithm. Obviously, if the database is very large, such a retrieval algorithm may not be appropriate. Organization of the database indices would be required so that the matching process will be done only on those images that are most likely similar to the query image. Graph clustering is one of the issues that we plan to investigate in the near future.

The retrieval algorithm has six steps. The construction of the input and model graphs from the query and database images is done in the first and the second steps respectively. The new matching algorithm is then called in the third step to compute the matching error. To perform this task, the algorithm should compute f_n, the error induced by the node-to-node matching, and f_e, the error induced by the edge-to-edge matching. Since a node includes multiple features, f_n must combine them using a weighting scheme. It is formulated as follows:

$$f_n = \alpha\ e_s(S_I, S_B) + \beta\ e_z(Z_I, Z_B) + \gamma\ e_c(C_I, C_B) \tag{1}$$

Where I and B represent the input and the database graph respectively, and α, β, γ are the weighting coefficients for the shape, color and size. Similarly, f_e is defined as:

$$f_e = \delta\ \ e_p(PR_I, PR_B) + \varepsilon\ \ e_d(D_I, D_B) \tag{2}$$

The error related to the shape e_s is set to zero if the two objects have the same shape; otherwise it is set to 1. Similarly, the error related to the relative position e_p is set to zero if the pair of objects have the same value according to this feature; otherwise the error is set to 1. The respective errors related to the size, the color and the distance between two objects, e_z, e_c and e_d, are defined by the following formulas:

$$e_z(Z_I, Z_B) = \frac{|Z_I - Z_B|}{(Z_I + Z_B)} \tag{3}$$

$$e_c(C_I, C_B) = \sqrt{(CL_I - CL_B)^2 + (CU_I - CU_B)^2 + (CV_I - CV_B)^2} \tag{4}$$

$$e_d(D_I, D_B) = \frac{|D_I - D_B|}{(D_I + D_B)} \tag{5}$$

In the fourth step, the retrieval algorithm computes a configuration error f_c associated to the image that does not have the same number of objects or of edges as the query image. This error is effectively added to the matching error if the coefficient c is greater than zero.

$$f_c = c\ \ (\|V_I| - |V_B\| + \|E_I| - |E_B\|) \tag{6}$$

$$matching\_error = f_n + f_e + f_c \tag{7}$$

Here $|V_I|, |E_I|, |V_B|$ and $|E_B|$ are the number of objects and edges in the query and the database images respectively.

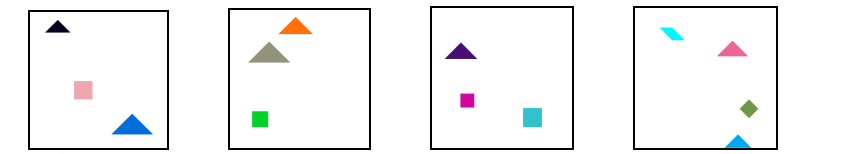
In the next step, the algorithm saves the matching error and the corresponding mappings into a matching list. This process will be repeated for each image in the database. Finally, the algorithm sorts the matching list and outputs the most similar images. The different parameters α, β, γ, δ, ε, and *c* provide a variety of possibilities for the users to control the query.

3.2 The Experimental Results

In this section, we present some image retrieval experiments performed using the new retrieval algorithm. The aim of these experiments is to show that the algorithm can indeed retrieve expected images similar to the query image and that such retrieval can be performed according to various needs of the user. We have conducted the retrieval with the generated database containing 1000 images. The number of objects in each image varies between 2 and 9. For each experiment, the specification of the query will be detailed and the first three similar images will be showed. For these experiments, the query image itself is not a member of the database.

3.2.1 Image Retrieval by Shape

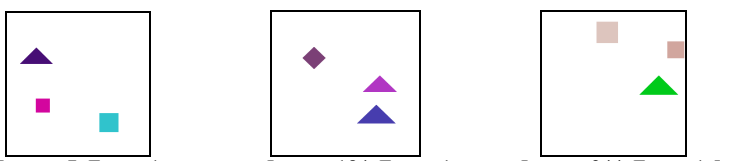
In this experiment, the user is searching for images that contain three objects. Only the shape (two triangles and a square) is important to the user. For this purpose, the parameters in the two dissimilarity functions should be set as follows: $\alpha = 1$, $c = 1$ and all other parameters are set to zero.



The image 528 has exactly the same objects as the query image according to the shape. In the second image only two objects can be matched and thus the error is not null. The third image has four objects and only two objects can be matched.

3.2.2 Image Retrieval by Shape and Relative Position

In this experiment, the same query image is used. The user is searching for images that contain objects having the same shape and relative position as in the query image. For this purpose, the parameters in the two dissimilarity functions should be set as follows: $\alpha = 0.5$, $\delta = 0.5$, $c = 1$ and all other parameters are set to zero.



The algorithm is able to find the similar images considering both criteria. The image 7 is one of the two closest ones to the query image. The result is appealing visually. The

(minimum) error of 1 is caused by two factors. One is the presence of a square object in the image 7 instead a triangle in the query image. The other one is the difference between the relative position square-triangle(big) in the query image and relative position Square-Square in the image 7.

4 Conclusion and Perspectives

The new graph-matching algorithm presented in this paper performs the search process in K phases. The promising mappings are examined in early phases. This allows computation of good matching with a small number of phases and increased computational efficiency. The new algorithm compares extremely well to the A*-based error correcting algorithm on randomly generated graphs. The new matching algorithm will be part of our content-based image retrieval system. A preliminary retrieval algorithm based on the new graph-matching algorithm has been reported here. Investigation is underway to discover cluster structures in the graphs so that the retrieval process can be focused on a reduced set of model graphs.

Acknowledgement

This work has been supported by a Strategic Research Grant from Natural Sciences and Engineering Research Council of Canada (NSERC) to the team composed of Dr. F. Dubeau, Dr. J. Vaillancourt, Dr. S. Wang and Dr. D. Ziou. Dr. S. Wang is also supported by NSERC via an individual research grant.

References

1. J. R. Ulmann. An algorithm for subgraph isomorphism, Journal of the association for Computing Machinery, vol. 23, no 1, January 1976, pp. 31-42.
2. D. G. Corneil and C. G. Gotlieb. An Efficient Algorithm for Graph Isomorphism, Journal of the Association for Computing Machinery, vol. 17, no. 1, January 1970, pp. 51-64.
3. J. Lladós. Combining Graph Matching and Hough Transform for Hand-Drawn Graphical Document Analysis. http://www.cvc.uab.es/~josep/articles/tesi.html.
4. B. T. Messmer and H. Bunke. A New Algorithm for error-Tolerant Subgraph Isomorphism Detection, IEEE Trans on PAMI, vol. 20, no. 5, May 1998.
5. Sanfeliu and K.S. Fu, A Distance Measure Between Attributed Relational Graphs for Pattern Recognition. IEEE Trans. on SMC, vol. 13, no. 3. May/June 1983.
6. W.H. Tsai and K.S. Fu. Error-Correcting Isomorphisms of Attributed Relational Graphs for Pattern Analysis. IEEE Trans. on SMC, vol. 9, no. 12. December 1979.
7. Hausdorff. Hausdorff distance http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html

8. Hlaoui and S. Wang. Graph Matching for Content-based Image Retrieval Systems. Rapport de Recherche, No. 275, Département de mathématiques et d'informatique, Université de Sherbrooke, 2001.
9. Y. Wang, K. Fan and J. Horng. Genetic-Based Search for Error-Correcting Graph Isomorphism. IEEE Trans. on SMC, Part B, vol. 27, no. 4. August 1997.
10. Huet, A. D. J. Cross And E. R. Hancock. Shape Retrieval by Inexact Graph Matching. ICMCS, vol. 1, 1999, pp. 772-776. http://citeseer.nj.nec.com/325326.html
11. X. Jiang, A. Munger, and H. Bunke. On Median Graphs: Properties, Algorithms, and Applications. IEEE Trans on PAMI, vol. 23, no. 10, October 2001.