This is the published version:


Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30044870

©1998 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Copyright: 1998, IEEE
An Efficient Least Common Subgraph Algorithm for Video Indexing

Kim Shearer, Svetha Venkatesh
Department of Computer Science, Curtin University of Technology
GPO Box U1987, Perth, 6001, Western Australia, kims@cs.curtin.edu.au

Horst Bunke
Institut für Informatik und Angewandte Mathematik, Universität Bern, Switzerland

Abstract

Many tasks in computer vision can be expressed as graph problems. This allows the task to be solved using a well studied algorithm, however, many of these algorithms are of exponential complexity. This is a disadvantage when considered in the context of searching a database of images or videos for similarity.

Recent work by Messmer and Bunke has suggested a new class of graph matching algorithms which uses a priori knowledge about a database of models to reduce the time taken during on-line classification. This paper presents a new algorithm which extends the earlier work to detection of the largest common subgraph.

1. Introduction

The subgraph isomorphism problem has applications in many areas of computer vision. Graphs provide a rich relational representation, with isomorphisms representing structural similarities. Furthermore inexact isomorphism detection provides an elegant and intuitive way of defining similarity measures. The graph isomorphism problem has been used in numerous vision applications, such as [7, 1].

One area where inexact subgraph isomorphism is used as a similarity measure is in image and video database indexing and retrieval[6, 2]. Although this provides an elegant and intuitive similarity measure, the exponential time complexity of graph isomorphism means that such systems are slow for large databases. In recent work Messmer and Bunke[4] have proposed two new algorithms for the subgraph isomorphism problem. These algorithms preprocess a database of models to reduce the complexity of on-line query processing, offering clear advantages in performance for image and video database applications.

One limitation of the Messmer and Bunke algorithms is that inexact isomorphism detection uses an edit distance measure, which is inappropriate for applications such as image and video database. In such situations a preferred measure is the largest common subgraph. In this paper a new algorithm is presented which develops the work of Messmer and Bunke to allow the detection of largest common subgraphs. The results section compares performance of these recent algorithms and traditional algorithms for typical queries over a video database.

2. Matching algorithms

The current algorithm for similarity retrieval by largest common subgraph is the maximal clique algorithm [3]. For a database of \(L\) models of size \(m\), and an input of size \(n\), table 1 gives the best and worst case complexity for the algorithms discussed in this paper. The performance of the clique matching algorithm makes it unsuitable for pictorial database work as the response time is unacceptable.

The work presented by Messmer and Bunke[4] provides complexity results, and empirical analysis on random graphs for their algorithms. In order to better assess the performance for video database applications we have applied these algorithms to the retrieval of video clips used in a video resequencing application[5]. The two algorithms used to represent the best traditional algorithms are Ullman’s algorithm for exact isomorphism detection and the A* algorithm for inexact isomorphism detection.

2.1. Decomposition Network algorithm

The decomposition network algorithm detects graph and subgraph isomorphisms from a set of model graphs to an
input graph. In order to reduce the matching time the network algorithm decomposes the model graphs into a network of subgraphs. \( G \) may be expressed as the four–tuple \((G, G', G'', E)\), where the two graphs \( G' \) and \( G'' \) are subgraphs of \( G \), which form \( G \) when joined by the edges in the edge set \( E \). The decomposition process can be continued recursively until \( G' \) and \( G'' \) are individual nodes. This decomposition forms a network, as in figure 1. Here two graphs are compiled into one decomposition network, sharing any common subgraphs.

The power of this algorithm comes from the sharing of structure, as given two similar graphs, there will be common subgraphs. Merging the two decompositions but including each common subgraph only once, gives a network which represents two model graphs and is more compact than the two decompositions in isolation. In addition to providing a compact representation, isomorphism search is also facilitated by this network, since each subgraph is matched to the input only once, rather than once per model. Matching is performed by comparing each vertex of the input against each initial one vertex node of the network. Each input that is correctly matched by an intitial node is passed down an edge to all descendents. For each internal \( n_i \) node of the network, the matching algorithm examines the subgraphs \( G' \) and \( G'' \) passed from its ancestors, and decides whether the input graph contains the edges in \( E \). If a correct mapping is found, the graph \( G \) is passed on to the descendents of \( n_i \). This leads to improvements in speed proportional to the similarity found between graphs.

### 2.2. Decomposition based LCSG algorithm

The decomposition based algorithm may be separated into two parts: the representation of multiple model graphs in a decomposition network, and the algorithm used to find isomorphisms. Messmer and Bunke provide two such algorithms, one to find exact isomorphisms from the models to the input, and one to provide inexact isomorphism detection using an edit distance measure. This section presents an additional algorithm to detect largest common subgraphs, using a decomposition network.

The algorithm to find the largest common subgraph introduces a wild card vertex label. This label, indicated by the symbol \( ? \), is used to map vertices from the input for which there is no correct mapping in the decomposition network. The initial step of the LCSG algorithm is to detect any exact isomorphisms between the input and the models. If there are no exact isomorphisms, a wild card label is introduced at every initial node of the network. These wild card nodes are then combined with other nodes to complete mappings for which there is no exact isomorphism. Figure 1 shows a decomposition network, annotated with the mappings for an example input. The absence of a vertex labelled \( b \) in the input is compensated for by mapping it to a wild card. This allows detection of a subgraph of order three in the left hand model. It remains to provide a control algorithm which uses the wild card mappings to ensure all best solutions are detected, as efficiently as possible.

The measure used by the LCSG algorithm to determine fitness of a node is the number of vertices mapped to wild cards, subtracted from the order of the largest model to which the node contributes. The order of the largest model is recorded for each node during network compilation. In figure 2 the decomposition network has been annotated with the order of largest models, and fitness measure for the example input at each node. This measure gives an upper limit to the possible size of the LCSG to which each network node can contribute, that is monotonically decreasing over the network descent. Unfortunately this measure provides only an estimate of the best order, which may in fact be a large over estimate in early parts of the network. This estimate may be refined by introducing a vertex frequency table at each node of the decomposition network.

The problem when calculating the fitness measure at a node is that we do not know whether a vertex that is mapped to a wild card may be correctly mapped in a separate branch of the network. It is possible to improve on this in some cases by recording, at each node, the maximum number of instances of each vertex label that exist in any models to which the node contributes. If a vertex which is contained in the input graph is mapped to \( ? \), then the frequency table can be examined to see whether there is another instance of the vertex, which may be correctly mapped. As an example,
A* with lookahead | Inexact network

<table>
<thead>
<tr>
<th>Query</th>
<th>Error</th>
<th>Time</th>
<th>Query</th>
<th>Error</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>liblr.10</td>
<td>0</td>
<td>10737 ms</td>
<td>liblr.10</td>
<td>0</td>
<td>172 ms</td>
</tr>
<tr>
<td>liblr.0</td>
<td>0</td>
<td>9073 ms</td>
<td>liblr.0</td>
<td>0</td>
<td>164 ms</td>
</tr>
<tr>
<td>wayq</td>
<td>6</td>
<td>9122 ms</td>
<td>wayq</td>
<td>6</td>
<td>223 ms</td>
</tr>
<tr>
<td>libq</td>
<td>12</td>
<td>35674 ms</td>
<td>libq</td>
<td>12</td>
<td>2851 ms</td>
</tr>
</tbody>
</table>

LCSG algorithm

<table>
<thead>
<tr>
<th>Query</th>
<th>Size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>liblr.10</td>
<td>9</td>
<td>35 ms</td>
</tr>
<tr>
<td>liblr.0</td>
<td>9</td>
<td>30 ms</td>
</tr>
<tr>
<td>wayq</td>
<td>4</td>
<td>144 ms</td>
</tr>
<tr>
<td>libq</td>
<td>6</td>
<td>234 ms</td>
</tr>
</tbody>
</table>

Table 2. Inexact isomorphism detection

if there is no more than one vertex labeled $a$ in any of the models to which a node $N$ contributes, then if the current node has mapped a vertex labelled $a$ to $?$, the best possible order for the LCSG is the order of the input minus one.

This gives an algorithm for detection of the LCSG between an input graph and a database of models that is efficient enough in space requirement to be used for general applications. Video and image databases generally provide a high level of similar structure making the decomposition network representation an attractive option.

3. Results

The video database used in the experiments was drawn mainly from our campus guide database [5]. In addition to the clips from the campus guide, there are a number of other clips of park and city scenes, and a small number of disparate clips of completely different types of scene. The total example database contains approximately 10 minutes of video. All times given in tables are in milliseconds, and are the average of a number of executions.

Table 2 outlines performance for inexact queries over the database. Each inexact algorithm was used to solve four queries, two of which had exact solutions, and two inexact. As can be seen performance of the A* algorithm rapidly becomes too slow for practical purposes, taking 35 seconds for the query libq. The inexact network algorithm returns much more useful performance, taking slightly less than three seconds for the query. As would be expected the LCSG algorithm returns the best performance. Using LCSG as the measure rather than edit distance means that less edges are examined, leading to slightly faster execution. The reduction from 35 seconds to less than two seconds indicates an important decrease in classification time.

More specific tests were run on the three inexact algorithms, using queries designed to produce the worst performance from the LCSG algorithm. The results of each of these queries, run over a collection of eleven specifically constructed graphs, are presented in table 3. Here we see that even with queries designed to extract the worst performance from the LCSG algorithm, and only a small number of graphs, the new algorithm performs admirably.

<table>
<thead>
<tr>
<th>Query</th>
<th>A*</th>
<th>Network</th>
<th>LCSG</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.6</td>
<td>282</td>
<td>30</td>
<td>17</td>
</tr>
<tr>
<td>12.2</td>
<td>53</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>14.1</td>
<td>84</td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>15.2</td>
<td>642</td>
<td>34</td>
<td>20</td>
</tr>
<tr>
<td>15.3</td>
<td>692</td>
<td>36</td>
<td>21</td>
</tr>
<tr>
<td>16.1</td>
<td>136</td>
<td>20</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 3. Approximate match against 11 graphs (in milliseconds)

3.1. Discussion

The recent developments in graph matching have clear implications for the digital library community. The ability to liberate the retrieval time from the number of elements of the database is a clear advantage. While the efficiency of the decomposition algorithm depends on the level of common structure in the models, image and video databases are but one example of an application where such structure occurs.

Further developments of these algorithms, or preprocessing versions of other graph matching algorithms, may revolutionise the area of image and video database indexing and retrieval. Current work is on an application of the decomposition network algorithm to efficient matching of dynamically changing graphs.

References