Structural Graph Matching With Polynomial Bounds On Memory and on Worst-Case Effort

Fred W. DePiero
CalPoly State University, San Luis Obispo, CA, USA, fdepiero@calpoly.edu

Abstract

A new method of structural graph matching is introduced and compared against an existing method and against the maximum common subgraph. The method is approximate with polynomial bounds on both memory and on the worst-case compute effort. Methods work on arbitrary types of graphs and tests with strongly regular graphs are included. No node or edge colors are needed in the methods; the common subgraph is extracted based in structural comparisons only. Monte Carlo trials are benchmarked with 100% additional (clutter) nodes. Results are shown to be typically within 1-2 nodes of the maximum common subgraph. Over 7500 test trials are reported with graphs up to 100 nodes.

1. Introduction

In this paper we address the problem of finding the maximum common subgraph via methods suited for practical, real-time measurement systems. Our approach has polynomial bounds on memory and on worst-case compute effort. Graph matching is accomplished solely via comparisons of structure. No assumptions on graph structure (planar, for example) are made herein. Our methods do ensure a one-to-one mapping between nodes in the two input graphs, and ensure the resultant common subgraph is a proper subgraph. However the method is approximate, so no guarantee of a maximum number of common nodes is possible.

The reason for setting these goals is to develop a method with broad applicability. Of particular interest are real-time applications where an approximation to the maximal common subgraph is acceptable, provided it can be found deterministically. For example with real-time range image registration, having fewer nodes than the maximum common subgraph is tolerable, but lengthy computations are not [4]. Use of graph matching in this application permits the steps of determining correspondence and pose to be separated and accomplished in a non-iterative fashion.

Established methods for graph matching may be categorized as either exact or approximate. As the problem of finding a maximum common subgraph is know to be NP-complete, exact methods inevitably have an exponential worst-case compute effort. Recently published approximate methods include [10] [12] [15] [9] [7]. The technique in [10] is optimized for large databases of objects that may contain similar subgraph structures. The method is efficient during recognition, but does require preprocessing time to construct a recognition library. It also uses attributed graphs. Most reported methods not only rely on graph attributes but are also iterative, making them less desirable for real-time systems. For example in methods based on relaxation labeling comparisons of node and edge colors are needed to establish an initial guess for the node mapping, before iterations begin [8]. More recent work in this area uses the color comparisons initially and during iterations [2]. Expectation-maximization is another method that has been used recently to iteratively adjust mapping probabilities [9]. In these iterative methods no guarantee of a globally optimum solution is possible. Hence the methods are both approximate and non-deterministic. Some methods also have exponential memory requirements [15], which may be problematic in applications.

Earlier work in graph matching included methods that provided exact results, but that required exponential worst-case execution times [14]. Other methods matched whole graphs, but not subgraphs, such as [11].

2. Comparing Graph Structure Dynamically

Two approximate methods are compared in this paper, one using 'Basis Graphs' ('BG', a new approach) and one using the 'LeRP' algorithm, which is based on length-r paths [5].

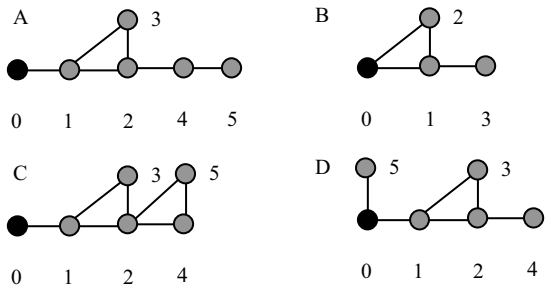

Figure 1. Basis graphs A-D. Root nodes are darker. The order of nodes used during placement is indicated. Basis E is a series of end-to-end links, 4 nodes total.

A feature that distinguishes the BG and LeRP methods from other techniques has to do with the size of the neighborhood used to compare local graph structure. In our techniques the size of the neighborhood varies dynamically – the more similar the structure, the larger

Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)
1051-4651/04 $ 20.00 IEEE

the neighborhood. We refer to the size of the neighborhood as the 'horizon'. Hence our techniques have a dynamic horizon.

Wilson and Hancock describe using a 'superclique' neighborhood in [15]. This is a good counter example of a method that uses a static horizon. The local neighborhood always consists of a central node and its adjacent nodes. Methods that employ a limited horizon for an initial comparison of structure must somehow expand or combine the local measures in order to then approximate the maximum common subgraph. This is accomplished in various ways, for example by making soft assignments and then iterating [7], via MAP probabilities and hill climbing [3], or via MAP & EM [9].

Using a dynamic horizon that can extend to potentially include all nodes in the graph is advantageous compared to a static horizon. As is benchmarked herein, the use of a dynamic horizon enables matching techniques which are non-iterative and that do not require any graph coloring or other attributes.

3. Approach Using Basis Graphs

Local structural comparisons are computed using basis graphs. Specifically, the basis graphs are employed to form an invariant ordering of nodes within a local neighborhood.

The basis graphs used herein were relatively small (4 to 6 nodes) compared to the graphs being matched that had up to 100 nodes. Basis graphs have a designated root node and do not contain any structural symmetry (automorphism). The root node has a special designation, making it non-symmetric to any other node. See Figure 1.

To compare the structural similarity of a pair of nodes (n1, n2) in graphs G1 and G2, first local neighborhoods L1 and L2 are established. L1 and L2 contain the nodes n1 and n2, respectively. The nodes within each L-neighborhood are ordered. Comparisons of L1 and L2 are made by counting the number of identical entries in the adjacency matrices (A1 and A2) of L1 and L2. This is similar to the complement of the edit distance. Because the nodes are ordered within L1 and L2, cyclic representations of the L-neighborhood are not necessary, as with [12]. When two neighborhoods contain a different number of nodes, the adjacency matrix for the smaller one is padded with zeros.

The invariant ordering of nodes within an L-neighborhood is accomplished using basis graphs. In this process a basis graph, B, is rooted at node n1 and all possible placements within G1 are enumerated from this root position. A histogram H1[n1][nx][i] is incremented if node i of B coincides with node nx in G1 during the placement operation. After histogramming, non-overlapping instances, b_k, of the basis graph are laid on top of G1, rooted at n1, by selecting nodes with the largest corresponding H1 value. The local ordering for L1 is then given by the order of nodes encountered during the b_k placement operation. See Figure 2.

Using the above histogramming method, basis graphs b_k are located in the 'most common' location within G1. The local ordering for L1 is then given by the order of nodes encountered during the b_k placement operation. The L2 neighborhoods in G2 are setup in a similar fashion. (See Figure 2). Instances of b_k in G1 may be partial versions. This can occur due to constraints of the G1 graph structure.

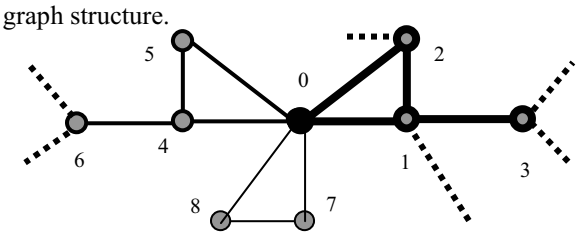

Figure 2. Three basis graphs are located relative to a common root node (in black). The order of placement of the basis graphs is indicated by bolder and lighter edges. Resulting node order for the neighborhood is indicated. Note the last basis graph placed was only partially complete. Additional edges present in the graph, not coincident with any instance of the basis, are dashed.

The ordered L-neighborhoods are formed and the edit distance (complement) is then computed for each pair of nodes in G1-G2. The degree of structural similarity for n1-n2 is given by the edit distance complement C(n1,n2). At this point a candidate mapping between the nodes in G1 and G2 may be identified. This is done in a greedy fashion, by selecting the nodes n1 and n2 with the largest C(n1,n2). The next largest C() value is then chosen and so on. The process continues provided all adjacencies are preserved for mapped nodes between G1 and G2. This greedy selection process yields a candidate mapping, M1. The greedy selection process is repeated, P times, using each of the P-highest $C(n_x,n_y)$ values to start the greedy process. The final mapping is based on the node-to-node correspondences that appear most often across all the candidate pairs in M1-MK and that yield the largest mapping.

This later step of finding the node-to-node mapping enforces global constraints associated with the overall graph structure. In a final step of the algorithm, nodes with zero degree were dropped from the final mapping that was computed.

In the test trials reported, multiple basis graphs were used. Each basis graph is placed in turn and the C(n1,n2) values summed. Various basis graphs and combinations of bases have been used in our experiments. Two quantitative measures to rank the basis graphs are presented below, involving 1) size of the matched graph and 2) the uniformity of inclusion of nodes of varying degree in the common subgraph.

The matching algorithm may readily be expanded to include comparisons of graph color or other attributes. These restrict potential matches, improving performance in terms of both speed and the size of the common subgraph, however benchmarks were not included herein.

4. Compute Effort & Memory Requirements

The compute effort and memory requirements for each stage of the algorithm are given in Table 1. This assumes an N-node input, and a V-node basis. Basis graphs used in this study were limited to 4 to 6 nodes (4<=V<=6).

	Processing Step	Effort	Memory
1a	Histogramming	$O(N^V)$	VN^2
1b	Placement	$O(N^2)$	VN^2
2	Neighborhood Comparisons	$O(N^2)$	N^2
3	Mapping	$O(PN^2)$	N^2

Table 1. Order of computational effort and memory.

5. Testing Method

A Monte Carlo-style analysis was performed to benchmark the BG and LeRP methods [5]. Benchmarks of processing time and of the final size of the common subgraph are reported for both BG and LeRP methods. Statisics on the better of the two methods is also reported. The better result was selected on a trial-by-trial fashion depending on the technique yielding the larger common subgraph.

Comparisons of BG and LeRP versus the maximum common subgraph are also reported. (Here the maximality refers to the number of nodes). These tests were more limited as the maximum common subgraph was found via exhaustive means. For these tests, the absolute difference in the number of nodes and the edit distance are both reported. The edit distance is given by the absolute sum of differences in the adjacency matrices of the maximum common subgraph and the approximate common subgraph. All permutations were enumerated to find the proper (lowest) edit distance.

Two different types of random graphs were used for inputs: Model A and strongly regular. Using Model A [13] is analogous to flipping a weighted coin to determine the existence of an edge. The strongly regular graphs were generated iteratively by randomly choosing pairs of nodes that each had a degree below a given target value. The strongly regular graphs were used because these are notoriously difficult [16] particularly for techniques that partition nodes by degree [11]. A test trial began by generating graphs G1 and G2 identically, randomizing node order, and then randomly adding nodes (100% increase in number).

6. Testing Results

Table 2 gives the size of the common subgraph computed, for 5000 total trials. Tests included Model A graphs (A ranging 0.15 to 0.3) and strongly regular graphs (degree ranging 3-7). The number of nodes in the initial graph varied. In each case 100% additional clutter nodes were added to each graph. Sizes of the common subgraph appear as a percentage of the number of nominal nodes (mean +/- one standard deviation).

Nominal	Better	Basis-G	LeRP
10	103 ± 8 %	101 ± 10 %	100 ± 12 %
50	105 ± 4 %	105 ± 6 %	105 ± 4 %
75	106 ± 3 %	99 ± 8 %	106 ± 3 %
100	106 ± 3 %	91 ± 10 %	106 ± 3 %

Table 2. Benchmarks of the number of nodes in the common subgraph. Data is given for 5000 trials, total. 100% additional clutter nodes, for each graph. Selecting the 'Better' result – based on size of match – yields good results over a wide range of tests. Note results from BG taper off for larger graphs – larger bases are needed in these cases. Common subgraphs over 100% of the initial number of nodes are possible due to the additive noise.

In the above tests, the mean compute time for basis graphs with inputs having 50, 75 and 100 nodes was 1.0, 4.2, and 14 seconds, respectively. When using LeRP, these times were 0.8, 4.6 and 18.4 seconds, respectively. Standard deviations were typically under 10%. Timing was benchmarked on a 1.6GHz PC.

	Better	Basis-G	LeRP
Edit Distance	0.5 ± 0.7	0.8 ± 1.4	1.4 ± 3.0
|Node Difference|	0.2 ± 0.2	0.3 ± 0.5	0.4 ± 0.8

Table 3. Edit distance between computed subgraphs versus the maximum common subgraph. The mean absolute difference in the number of nodes of the maximum common subgraph is also shown. Results are given for 125 trials, input graphs had 10 nodes, with 100% clutter added to one graph only. Table values are the means of tests that included Model A (0.15-0.25) and strongly regular graphs (degree 5-7).

These tests demonstrate the ability to find a common subgraph within 1-2 nodes of the maximum, rapidly, and while not requiring no node or edge colors. This is advantageous compared to methods such as [7] and [10] which have performance that decrease with reduced dynamic range of coloring. Results also appear to have higher accuracy while in the presence of greater noise (100% rather than 50%) than in [18].

Note the BG algorithm failed to report a result once out of 7625 trials. This case is under study…

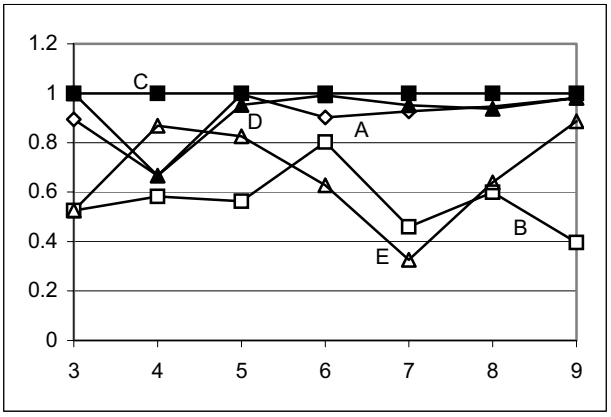

Table 4. Probability estimate of a node being included in a common subgraph, as a function of degree. Pairs of input graphs had 50 nodes, each with 100% clutter. Each basis graph (A-E) was used individually for Table 4. Tests with basis C in combination with others had nearly identical performance to C alone, 2500 tests total.

Based on the above probability estimates and on testing of the size of common subgraphs, bases A-D were selected to form a set for the BG algorithm. The A-D set was used in the tests reported in Tables 2 and 3.

7. Dissemination of Software

See the author's web page [6] to download. The software is free, for non-commercial, non-profit purposes.

8. Conclusion & On-Going Studies

The basis graph technique incorporates a dynamic comparison horizon, as does LeRP. This mitigates some of the problems associated with localized structural comparisons in approaches with a limited horizon. As benchmarked here, BG and LeRP yield results near the maximum common subgraph.

We characterize a good basis graph (or set) as one that yields matched graphs near the maximum common subgraph and one with uniform probabilities of appearance for nodes of varying degrees.

Comparisons against the maximum common subgraph indicate the BG method may be somewhat better than LeRP. If time permits in an application, then running each and selecting the larger result would be preferred.

In the BG approach, larger graphs will require larger bases. While larger bases may certainly be provided, LeRP may be preferable in applications where the input graph size varies widely or cannot be predicted.

We have interest in pattern matching with graphs that include a probabilistic description of structure. These probabilities describe how likely a given node and edge is present. Matching the probabilistic graphs could be quite helpful in a clustering analysis used with graph-valued measurements. This is currently under investigation for speech recognition.

As suggested by a conference reviewer, it may be possible to improve efficiency by using a method similar to Tarjan [17] to reduce effort while histogramming.

The author would like to acknowledge reviews and consultations with John K. Carlin and Leonard D. Myers.

9. References

[1] M. Carcassoni and E.R. Hancock, Correspondence Matching with Modal Clusters, *IEEE Trans. PAMI*, 25 (12) (2003) 1609-1614.

[2] W J Christmas, J Kittler and M Petrou, Probabilistic feature-labelling schemes: modelling compatibility coefficient distributions. *Image and Vision Comp*, 14 (1996) 617-625.

[3] A.D.J. Cross, E.R. Hancock, Graph matching with a dual-step EM algorithm, *IEEE Trans. PAMI*, 20 (11) (1998) 1236.

[4] F. W. DePiero, "Deterministic Surface Registration at 10Hz Based on Landmark Graphs With Prediction," 14th British Machine Vision Conf. (*BMVC2003*), Norwich, UK, Sept, 2003.

[5] F. W. DePiero and D.W. Krout, LeRP: An algorithm using length-r paths to determine subgraph isomorphism, *Pattern Rec Journal,* 24 (1) (2003) 33-46.

[6] F. W. DePiero., "Home Page", Software for Graph Matching, www.ee.calpoly.edu/~fdepiero/ (June, 2004).

[7] S. Gold, A Rangarajan, A graduated assignment algorithm for graph matching, *IEEE Trans. PAMI*, 18 (4) (1996) 377-388.

[8] J. Kittler, E. R. Hancock, Combining Evidence in Probabilistic Relaxation, *Intl. Journal of Pattern Recognition and Artificial Intelligence*, 3 (1989) 29-51.

[9] B. Luo and E.R. Hancock, Structural graph matching using the EM algorithm and singular value decomposition, *IEEE Trans. PAMI*, 23 (10) (2001) 1106-1119.

[10] B.T.Messmer, H. Bunke, A new algorithm for error-tolerant subgraph isomorphism detection, *IEEE Trans. PAMI*, 20 (5) (1998) 493-504.

[11] B. McKay. Practical Graph Isomorphism, *Congressus Numerantium*, 30 (1981) 45-87.

[12] R. Myers, R.C. Wilson, E.R. Hancock, Bayesian graph edit distance, *IEEE Trans. PAMI*, 22 (6) (1997) 628-635.

[13] E. M. Palmer, Graphical Evolution – An Introduction to the Theory of Random Graphs, Wiley-Interscience, 1985.

[14] A. Sanfeliu, K.S. Fu, A distance measure between attributed relational graphs for pattern recognition, *IEEE Trans. Systems, Man and Cybernetics*, 13 (1983) 353-363.

[15] R.C. Wilson, E.R. Hancock, Structural matching by discrete relaxation, *IEEE Trans. PAMI*, 19 (6) (1997) 634-648.

[16] R. C. Read and D. G. Corneil, The graph isomorphism disease, Journal of Graph Theory, 1 (1) 339-363 (1977).

[17] R. Tarjan, et.al., Time Bounds for Selection, CS Dept., Stanford, Tech. Report STAN-CS-73-349 (1973).

[18] T. Caelli and S. Kosinov, An eigenspace projection clustering method for inexact graph matching, *IEEE Trans. PAMI*, 26 (4) (2004) 515-519.