Graph Similarity for Data Mining
Lot 6 of Data Analytics FY17/18

Meeting Presentation

Peter Rodgers

Algorithms for the Comparison of Graphs

- Graph Isomorphism
- Graph Edit Distance
- Various Others

- Used in Data Mining
  - Motif finding
  - Pattern matching by finding similar subgraphs

- This project implemented and profiled a number of algorithms

Dover

- The Dover software system is designed for implementing performance graph algorithms
- Analysis of the order of 1M nodes and 10M edges and beyond
    - Changes in this project to deal with directed, labelled graphs.
- Open Source (was GPL, now Apache)
- Pure Java for portability

https://www.cs.kent.ac.uk/projects/dover/

https://github.com/peterrodgers/dover

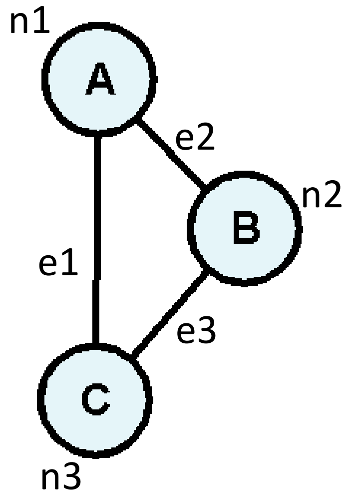
Algorithm A: Exact Graph Isomorphism

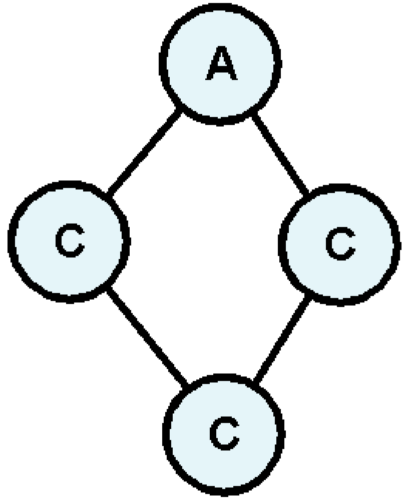
- Graph isomorphism describes the equality of graphs.
- Algorithms for exact isomorphism are exponential in the worst case
  - but a number of optimizations means that many non-isomorphic graphs can be found in polynomial time
  - And there is much pruning that can be done in the exponential case
- In terms of similarity, isomorphism gives a binary similarity result in that graphs are either the same or not
- Exact Isomorphism on directed, node labelled graphs added to Dover

Graph Edit Distance

- Graph Edit Distance (GED) is by far the most common graph similarity measure.
- Edit operations are each given a cost
- The GED of two graphs is the sum of the cost of the edits required in one graph to turn it into an isomorphism of the other
  - We would like the minimum cost

GED Example





Relabel n2 to “C” – cost 4

Remove edge e1 – cost 3

Add node with label “C” – cost 5

Add edge from n3 to new node – cost 3

Add edge from n1 to new node – cost 3

Total GED cost between graphs = 16

Algorithm B: Node degree profile

- This highly simplistic, but scalable, graph difference algorithm has been modified to deal with directed graphs.
- As previously, it takes the degree profiles of the two graphs and sums the difference in node count of each degree
- New, the directed graphs case where it considers the in degree and out degree separately

Algorithm C: Exact Graph Edit Distance

- A brute force test that applies each graph edit operation in turn
- A* algorithm adding edits to the cheapest edit list first
  - Pruning known non-minimal branches (e.g. double node relabels)
- Highly exponential
  - Around 7 edit operation limit

Algorithm D: Simple Approximate GED

- Applies a mapping between nodes based on node degree
  - Takes account of the directed case
  - Does not consider labels in initial mapping
- Random swaps between mapped nodes followed by a test of edit cost
  - Simulated annealing, takes some bad swaps early on
  - Edit cost includes node labels
- Good at finding exact edit distance in small cases
- Will scale, but accuracy drops off rapidly
- Developed for the project, no prior work

Algorithm E: Bipartite Approximate GED

- Forms an estimated cost matrix
  - g1 nodes on rows and g2 nodes on columns
  - Local node costs found between nodes in g1 and g2 based on comparing node labels and connecting edges
- Then runs an assignment algorithm to map rows to columns
  - Various assignment algorithms tested
  - Volgenant-Jonker proved to be the fastest

Fankhauser S, Riesen K, Bunke H. Speeding up graph edit distance computation through fast bipartite matching. Graph-based representations in pattern recognition. 2011:102-11

Algorithm F: Hausdorff Distance GED

- Lower bound method
- Takes a local approach to discovering the smallest possible edits for each node and edge
- Issues
  - Poor results and unimpressive performance
  - Perhaps because of lack of a reference implementation

Fischer, A., Suen, C. Y., Frinken, V., Riesen, K., and Bunke, H. (2015). Approximation of graph edit distance based on Hausdorff matching. Pattern Recognition, 48(2), 331-343.

- A simple lower bounds method was also implemented, counting edges, and number of node labels that need to change

Algorithm G: Iterative Neighbourhood Graph Similarity

- Uses the structural similarity of local neighbourhoods
  - Results in values between 0 and 1, with 0 similar, 1 dissimilar
- Issues
  - Lack of symmetry
    - Similarity(g1,g2) != Similarity(g2,g1)
  - Tends to 1 for most random graphs in the directed version
  - Poor performance, unfeasible beyond around 1000 nodes

L. Zager, G. Verghese, Graph similarity scoring and matching, Applied Mathematics Letters 21 (2008) 86-94

Algorithm H: Belief Propagation Graph Similarity

- This needs a node mapping between the two graphs to work
  - So seems to be less useful than other methods
  - But may have use in testing differences between changes in large graphs
- We use a node mapping method similar to that used for the Simple GED code
- Issues
  - It works only on simple, not self-sourcing graphs
  - It does not consider node labels
  - Calculates difference with adjacency matrices so a directed version is not feasible
  - As the graph size increases, it tends to 1

Danai Koutra, Joshua T. Vogelstein, and Christos Faloutsos. Deltacon: A principled massive-graph similarity function. SIAM International Conference on Data Mining. 2013.

Algorithm I: Random Trail Graph Similarity

- From each node in g1, take **t** random trails of length **k** nodes
  - A trail is a route through a graph using edges only once, but potentially visiting a node multiple times
- For each node in g2, find the longest matching trail using breadth first search.
  - Exact matches score 0, the shorter the trails, the closer the score is to 1
- This produces an affinity score between each node
  - Pick the node mapping that minimizes overall affinity
- Issues
  - Highly dependent on choices for t and k
  - Lacks symmetry
  - $O(n^2)$ time complexity, but with a big constant
- However it is a good approximation to graph isomorphism
- Developed for the project, no prior work

Scaling Data Summary

Randomly Generated, unlabelled, undirected, simple graphs. Timing in seconds. GED simple t=1, k=3

900 Nodes - 9,000 Edges

GED simple cost:	GED bipartite cost:	GED hausdorff cost:	GED lower cost:	belief simple cost:	neighbour hood cost:	degree difference cost:	random trail cost:
102813.4	103395.6	125.125	1402.3	0.998031	0.999593	185.1	0.112399

GED simple time:	GED bipartite time:	GED hausdorff time:	GED lower time:	belief simple time:	neighbour hood time:	degree difference time:	random trail time:
1.0158	0.142	0.1764	0	1.6222	155.612	0	1.5771

4,000 Nodes – 40,000 Edges

GED simple cost:	GED bipartite cost:	GED hausdorff cost:	GED lower cost:	belief simple cost:	degree difference cost:	random trail cost:
498942.3	498934.5	763.3	7833.5	0.999193	574.3	0.07166

GED simple time:	GED bipartite time:	GED hausdorff time:	GED lower time:	belief simple time:	degree difference time:	random trail time:
1.0376	5.0748	3.4314	0	188.0526	0	33.6019

9,000 Nodes - 90,000 Edges

GED simple cost:	GED bipartite cost:	GED hausdorff cost:	GED lower cost:	degree difference cost:	random trail cost:
960064.1	960058.4	1259	13507.8	1459	0.078996

GED simple time:	GED bipartite time:	GED hausdorff time:	GED lower time:	degree difference time:	random trail time:
1.1545	43.7068	17.7142	0	0	219.5

10,000 Nodes – 100,000 Edges (memory error for bipartite)

GED simple cost:	GED bipartite cost:	GED hausdorff cost:	GED lower cost:	degree difference cost:
1113962	1114088	1506.9	15722.1	1489.9

GED simple time:	GED bipartite time:	GED hausdorff time:	GED lower time:	degree difference time:
1.1797	55.6051	21.5357	0	0.0016

40,000 Nodes – 400,000 Edges

GED simple cost:	GED hausdorff cost:	GED lower cost:	degree difference cost:
3954985	6156.6	35352	3327.4

GED simple time:	GED hausdorff time:	GED lower time:	degree difference time:
2.9656	330.1511	0	0.0032

1,200,000 Nodes – 12,000,000 Edges

GED simple cost:	GED lower cost:	degree difference cost:
1.17E+08	1349439	157636.9

GED simple time:	GED lower time:	degree difference time:
592.8334	0.0001	0.0277

Isomorphism Data

20 to 1000 nodes, 5x edges, variety of labelled/unlabelled, directed/undirected, simple/nonsimple

Largest graphs approx 2 mins for the exact isomorphism, 10 seconds for random trail, t=3, k=4

minor rewiring in g2 compared to g1

	GED simple	GED bipartite	random trail	exact isomorphism
isomorphic	0	0	106	0
non-isomorphic	4000	4000	3894	4000

g1 and g2 isomorphic

	GED simple	GED bipartite	random trail	exact isomorphism
isomorphic	51	22	4000	4000
non-isomorphic	3949	3978	0	0

Varying GED between g1 and g2

Directed Labelled Non-simple, 100 runs each, all edit costs = 1. GED simple t=3, k=4

20 Nodes 30 Edges

edits	GED simple cost:	GED bipartite cost:	random trail cost:
1	4.72	25.52	0.144417
2	4.67	27.22	0.249031
3	11.99	29.49	0.336084
4	10.33	31.61	0.410321
5	10.5	33.59	0.472403
6	15.95	35.77	0.529533
7	19.42	38.15	0.575857
8	15.42	40.48	0.614709
9	17.46	42.32	0.634673
10	21.3	43.45	0.657355

100 Nodes 1000 Edges

edits	GED simple cost:	GED bipartite cost:	random trail cost:
1	1621.99	699.78	0.03445
2	1624.48	706.36	0.067385
3	1622.85	710.24	0.096878
4	1622.69	717.4	0.127452
5	1625.8	723.58	0.156464
6	1623.88	729.52	0.184635
7	1622.72	733.49	0.210832
8	1626.87	742.65	0.23697
9	1623.42	748.37	0.260726
10	1624.93	750.49	0.282923

Conclusions

- We have developed a number of graph similarity measures.
- The immediate conclusions of this work are that, unless there are very specific needs, graph edit distance remains the most effective non-binary similarity measure.
- Exact GED is not feasible, except in very small cases, so approximate methods, as implemented in this project must be used.
- Graph isomorphism can be effectively approximated up to graphs of several thousand
- More details on the project and code can be found at: https://www.cs.kent.ac.uk/projects/dover/