Grph: list of algorithms

Luc Hogie (CNRS), Aurélien Lancin (INRIA), Issam Tahiri (INRIA),
Grgory Morel (INRIA), Nathan Cohen (INRIA), David Coudert (INRIA)

September 16, 2013

Contents

1 Traversal **4**
- 1.1 Breadth First Search 4
- 1.2 Depth First Search 4
- 1.3 Random Search 4
  - 1.3.1 Single visit 4
- 1.4 Parallelization 4

2 Shortest paths **4**
- 2.1 K-Shortest paths 4
- 2.2 BFS-based 5
- 2.3 Dijkstra 5
- 2.4 Bellman-Ford 5
  - 2.4.1 David mod 5
- 2.5 Floyd Warshall 5

3 Clustering **5**
- 3.1 Kernighan-Lin 5
- 3.2 Number of triangles 5
  - 3.2.1 Native 5
  - 3.2.2 Matrix-based 5
- 3.3 List all triangles 6
- 3.4 List all paths 6
- 3.5 List all cycles 6
- 3.6 Finding cliques 6

4 Distances **6**
- 4.1 Distance matrix 6
- 4.2 Eccentricity 6
- 4.3 Radius 6
- 4.4 Diameter 6
- 4.5 Center vertex 6
  - 4.5.1 Minimum eccentricity 6
  - 4.5.2 Two sweeps 6
  - 4.5.3 Distance matrix-based diameter 7
  - 4.5.4 Two sweeps 7
  - 4.5.5 Four sweeps 7
- 4.6 Shortest cycle and girth 7

5 Connected components **7**
5.1 Set of connected components . . . 7
5.2 Connected component containing a given vertex . . . 7

6 Cuts **7**
6.1 Test if two vertex sets form a cut in the graph . . . 7
6.2 Test if a set of vertex sets form a cut in the graph . . . 7
6.3 Computes the set of edges of a given cut . . . 7
6.4 Computes the size of given cut . . . 7
6.5 Sparse cut . . . 7

7 Navigation **8**
7.1 Adjacency matrix . . . 8
7.2 Incidence matrix . . . 8
7.3 Degrees . . . 8
7.4 Clustering coefficient . . . 8
7.5 Set of edges connecting two given vertices . . . 8
7.6 Set of edges incident to two given vertices . . . 8
7.7 Headless/tailless edges . . . 8
7.8 In/out neighbors . . . 8
7.9 Isolated vertices . . . 8
7.10 Loops . . . 8
7.11 Set of vertices at a given distance . . . 8
7.12 Set of vertices incident to a given edge . . . 9
7.13 Set of vertices with a given in/out degree . . . 9
7.14 Inclusion . . . 9
7.15 Equality . . . 9
7.16 Difference detection . . . 9
7.17 Cloning . . . 9

8 Structural tests **9**
8.1 Connected . . . 9
8.2 Complete . . . 9
8.3 Null . . . 9
8.4 Reflexive . . . 9
8.5 Regular . . . 10
8.6 Trivial . . . 10
8.7 Simple . . . 10
8.8 Tree . . . 10
8.9 Chordal . . . 10
8.10 Hypergraph . . . 10
8.11 Multigraph . . . 10
8.12 Irreflexive . . . 10
8.13 Acyclic . . . 10
8.14 Hamiltonian . . . 10
8.14.1 Ore's theorem . . . 10
8.14.2 Dirac's theorem . . . 10
8.15 Anti-graph . . . 10
8.16 Bipartiteness . . . 11

9 Topology generators **11**
9.1 GNM . . . 11
9.2 GNP . . . 11
9.3 Clique . . . 11
9.4 Bus . . . 11
9.5 Chain . . . 11
9.6 Star . . . 11
9.7 Grid . . . 11

9.8 Inet 11
9.9 Wireless backhaul 12
9.10 Ad hoc network 12
9.11 Random Newman Watts Strogatz 12
9.12 Random tree 12
9.13 Ring 12
9.14 Chordal 12
9.15 Brite 12
9.16 GLP 12
9.17 R-mat 12

10 Input/Output **13**
10.1 DOT/GraphViz 13
10.2 Dimacs 13
10.3 LAD 13
10.4 GraphML 13
10.5 GML 13
10.6 DGS (read-only) 13
10.7 Grph binary 13
10.8 Grph text 13

11 Unclassified **14**
11.1 Subgraph isomorphism 14
11.2 Graph label-based pattern matching 14
11.3 Transitive closure 14
11.4 Line graph 14
11.5 Pruning 14
11.6 Evolutionary creation of graph instances 14
11.7 Random-walk based centrality 14
11.8 Graph isomorphism 14
11.9 Subgraph isomorphism 15
11.10Full-featured subgraph isomorphism 15
11.11Maximum clique 15
11.12Topological sort 15
11.13Complement 15

12 Set operations **15**

13 Value distribution **15**

14 Covering and packing **15**
14.1 Minimum Vertex Cover 15
14.1.1 BruteForceMinimumVertexCoverAlgorithm 16
14.1.2 BranchingMinimumVertexCoverAlgorithm 16
14.1.3 NiedermeierMinimumVertexCoverAlgorithm 16
14.2 Maximum Matching 16
14.2.1 BipartiteMaximumMatchingAlgorithm 16
14.2.2 LPBasedMaximumMatchingAlgorithm 16
14.3 Maximum Independent Set 16
14.3.1 LPBasedMaximumIndependentSetAlgorithm 16
14.3.2 FominGrandoniKratschMaximumindependentSetAlgorithm 17

15 Coloring **17**
15.1 Chromatic number 17
15.2 BipartitenessAlgorithm 17

16 Coloring **17**
16.1 OGDF 17

1 Traversal

Graph traversal algorithms are designed in the form of “abstract traversers”, which means that at each step of the traversal, they invoke dedicated abstract method that the user need to implement in order to meet its needs.

1.1 Breadth First Search

Breadth-first search (BFS) is a graph search algorithm that begins at the root node and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal, if any.

1.2 Depth First Search

Depth-first search (DFS) starts at the root and explores as far as possible along each branch before backtracking.

1.3 Random Search

The random search starts from a given root node and select a random neighbor as the next step. It does not guarantee that the entire graph will be visited, even though it is connected. Nodes may be visited several times.

1.3.1 Single visit

An adaptation of the previously described scheme, but ensures that any single node will be visited only once.

1.4 Parallelization

When searches are requested for multiple source, they are computed in parallel. To do this *Grph* instantiates 4 times more threads than cores on the computer.

2 Shortest paths

The shortest path problem is the problem of finding a path between two vertices (or nodes) such that the sum of the weights of its constituent edges is minimized. An example is finding the quickest way to get from one location to another on a road map; in this case, the vertices represent locations and the edges represent segments of road and are weighted by the time needed to travel that segment.

2.1 K-Shortest paths

Grph provides a bridge to an implementation of Yen’s algorithms, hosted at Google code `http://code.google.com/p/k-shortest-paths/`.

2.2 BFS-based

2.3 Dijkstra

Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1956 and published in 1959,[1][2] is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree.

2.4 Bellman-Ford

The Bellman-Ford algorithm computes single-source shortest paths in a weighted digraph. For graphs with only non-negative edge weights, the faster Dijkstra's algorithm also solves the problem. Thus, Bellman-Ford is used primarily for graphs with negative edge weights.

2.4.1 David mod

This modification uses two correlated stacks.

2.5 Floyd Warshall

3 Clustering

3.1 Kernighan-Lin

This algorithm runs in $\Theta(n^2 log(n))$.

Kernighan-Lin is an heuristic algorithm for solving the graph partitioning problem. The algorithm has important applications in the layout of digital circuits and components in VLSI.

3.2 Number of triangles

Two algorithms counting the number of triangles in a graph are implemented in *Grph*.

3.2.1 Native

Implemented in C. Uses Matthieu Latapy (LIP6) source code.

3.2.2 Matrix-based

Developed by Gregoy Morel

3.3 List all triangles

3.4 List all paths

3.5 List all cycles

3.6 Finding cliques

Uses the C implementation of `Cliquer`, developped by Patric R. J. Östergård, Professor at Aalto University (Finland).
`http://users.tkk.fi/pat/cliquer.html`

4 Distances

4.1 Distance matrix

The graph distance matrix, sometimes also called the all-pairs shortest path matrix, is the square matrix $d_{i,j}$ consisting of all graph distances from vertex v_i to vertex v_j.

4.2 Eccentricity

The eccentricity $\epsilon(v)$ of a graph vertex v in a connected graph G is the maximum graph distance between v and any other vertex u of G. For a disconnected graph, all vertices are defined to have infinite eccentricity.

4.3 Radius

The radius of a graph is the minimum graph eccentricity of any graph vertex in a graph.

4.4 Diameter

The length $max_{u,v}d(u,v)$ of the "longest shortest path" (i.e., the longest graph geodesic) between any two graph vertices (u,v) of a graph, where $d(u,v)$ is a graph distance. In other words, a graph's diameter is the largest number of vertices which must be traversed in order to travel from one vertex to another when paths which backtrack, detour, or loop are excluded from consideration.

4.5 Center vertex

4.5.1 Minimum eccentricity

4.5.2 Two sweeps

Starts from any node. Performs one BFS and choose the farthest node as the new source. Performs another BFS and returns the node in the middle of the path between the source and the farthest node.

4.5.3 Distance matrix-based diameter

This algorithm runs in $\Theta(n^2)$. n being the number of vertices in the graph.

4.5.4 Two sweeps

Also called two-sweeps diameter approximation. Gives a lower bound.

This algorithm runs in $\Theta(2n)$.

4.5.5 Four sweeps

Also called two-sweeps diameter approximation. Gives a lower bound.

This is a contribution of Laurent Viennot.

4.6 Shortest cycle and girth

It is possible to compute a shortest cycle, and consequently the *girth* of a graph (defined as the length of a shortest cycle) by calling the methods `getShortestCycle()` and `getGirth()`.

5 Connected components

A connected component of an undirected graph is a subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices.

5.1 Set of connected components

5.2 Connected component containing a given vertex

6 Cuts

A cut is a partition of the vertices of a graph into two disjoint subsets. The cut-set of the cut is the set of edges whose end points are in different subsets of the partition. Edges are said to be crossing the cut if they are in its cut-set.

6.1 Test if two vertex sets form a cut in the graph

6.2 Test if a set of vertex sets form a cut in the graph

6.3 Computes the set of edges of a given cut

6.4 Computes the size of given cut

6.5 Sparse cut

Written by Issam Tahari

7 Navigation

7.1 Adjacency matrix

The adjacency matrix, sometimes also called the connection matrix, of a simple graph is a matrix with rows and columns labelled by graph vertices, with a 1 or 0 in position (v_i, v_j) according to whether v_i and v_j are adjacent or not. For a simple graph with no self-loops, the adjacency matrix must have 0s on the diagonal. For an undirected graph, the adjacency matrix is symmetric.

7.2 Incidence matrix

The incidence matrix of a graph gives the (0,1)-matrix which has a row for each vertex and column for each edge, and (v,e)=1 iff vertex v is incident upon edge e (Skiena 1990, p. 135). However, some authors define the incidence matrix to be the transpose of this, with a column for each vertex and a row for each edge.

7.3 Degrees

Consider in/out edge/vertex degrees.

7.4 Clustering coefficient

A clustering coefficient is a measure of degree to which nodes in a graph tend to cluster together. Evidence suggests that in most real-world networks, and in particular social networks, nodes tend to create tightly knit groups characterised by a relatively high density of ties.

7.5 Set of edges connecting two given vertices

7.6 Set of edges incident to two given vertices

7.7 Headless/tailless edges

7.8 In/out neighbors

7.9 Isolated vertices

Returns the set of vertices that have no incident edge.

7.10 Loops

Returns the of edges that are self-loops.

7.11 Set of vertices at a given distance

Returns the set of vertices at the given distance.

7.12 Set of vertices incident to a given edge

7.13 Set of vertices with a given in/out degree

7.14 Inclusion

Test if a given graph is included in another graph.

7.15 Equality

Test if a given graph equals another graph.

7.16 Difference detection

Reports the difference with two given graphs.

7.17 Cloning

Builds a clone of the given graph. Topology and properties are duplicated.

8 Structural tests

8.1 Connected

; A graph which is connected in the sense of a topological space, i.e., there is a path from any point to any other point in the graph. A graph that is not connected is said to be disconnected. This definition means that the null graph and singleton graph are considered connected, while empty graphs on n¿=2 nodes are disconnected.

8.2 Complete

A complete graph is a graph in which each pair of graph vertices is connected by an edge.

8.3 Null

The term "null graph" is used both to refer to any empty graph and to the empty graph on 0 nodes.

8.4 Reflexive

A reflexive graph is a pseudograph such that each vertex has an associated graph loop.

8.5 Regular

A graph is said to be regular of degree r if all local degrees are the same number r. A 0-regular graph is an empty graph, a 1-regular graph consists of disconnected edges, and a 2-regular graph consists of disconnected cycles. The first interesting case is therefore 3-regular graphs, which are called cubic graphs.

8.6 Trivial

8.7 Simple

A simple graph, also called a strict graph, is an unweighted, undirected graph containing no graph loops or multiple edges.

8.8 Tree

8.9 Chordal

8.10 Hypergraph

An hypergraph is a graph in which generalized edges (called hyperedges) may connect more than two nodes. A connection between two or more vertices of a hypergraph. A hyperedge connecting just two vertices is simply a usual graph edge.

8.11 Multigraph

; The term multigraph refers to a graph in which multiple edges between nodes are either permitted (Harary 1994, p. 10; Gross and Yellen 1999, p. 4) or required.

8.12 Irreflexive

8.13 Acyclic

8.14 Hamiltonian

8.14.1 Ore's theorem

8.14.2 Dirac's theorem

8.15 Anti-graph

An anti-graph is an orientation of a graph where every vertex is a source or a sink.

8.16 Bipartiteness

9 Topology generators

http://sndlib.zib.de/home.action

9.1 GNM

Erdos and Renyi GNM model: adds m edges to a graph of n vertices, connecting random pairs of vertices.

The maximum number of edges is $x = n(n-1)/2$. If $m > x$ then the generator starts by constructing a clique of all vertices from which it then randomly removes $x - m$ edges.

9.2 GNP

Erdos and Renyi GNP model: connect each pair of vertices according to a given probability.

9.3 Clique

Connect the vertices in the graph as a clique. Vertices are connected to all others. Previously existing edges are ignored.

9.4 Bus

Connect the vertices in the graph as a bus: they are all included in a new undirected hyper-edge. Previously existing edges are ignored.

9.5 Chain

For instance, the chain topology model can be used to model in-line networks following railways, enabling train passengers to connect to the internet.

9.6 Star

9.7 Grid

Connect the vertices in the graph as a grid.

9.8 Inet

Inet, currently at version 3.0, is an Autonomous System (AS) level Internet topology generator.

9.9 Wireless backhaul

Degenerates a grid in the following manner: the farther is an edge from the initial node, the more probability it has to be deleted. An edge is never deleted if deleting it would result in disconnecting the graph.

9.10 Ad hoc network

A wireless ad hoc network is a decentralized wireless network.[1] The network is ad hoc because it does not rely on a preexisting infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks. Instead, each node participates in routing by forwarding data for other nodes, and so the determination of which nodes forward data is made dynamically based on the network connectivity.

9.11 Random Newman Watts Strogatz

From the SageMath documentation: First create a ring over n nodes. Then each node in the ring is connected with its k nearest neighbors. Then shortcuts are created by adding new edges as follows: for each edge u-v in the underlying '$n-$ring with k nearest' neighbors'; with probability p add a new edge $u - w$ with" + "randomly-chosen existing node w.

9.12 Random tree

9.13 Ring

Connect the vertices in the graph as a ring. Previously existing edges are ignored.

9.14 Chordal

9.15 Brite

Uses the external Brite topology generator. BRITE supports multiple generation models including models for flat AS, flat Router and hierarchical topologies.

9.16 GLP

9.17 R-mat

The R-mat topology generator is implemented in C.

10 Input/Output

10.1 DOT/GraphViz

DOT is a plain text graph description language. It is a simple way of describing graphs that both humans and computer programs can use. DOT graphs are typically files that end with the .gv (or .dot) extension. Various programs can process DOT files. Some, like dot, neato, twopi, circo, fdp, and sfdp, will read a DOT file and render it in graphical form. Others, like gvpr, gc, accyclic, ccomps, sccmap, and tred, will read a DOT file and perform calculations on the represented graph. Finally, others, like GVedit, KGraphEditor, lefty, dotty, and grappa, provide an interactive interface.

10.2 Dimacs

10.3 LAD

10.4 GraphML

GraphML is an XML-based file format for graphs. The GraphML file format results from the joint effort of the graph drawing community to define a common format for exchanging graph structure data. It uses an XML-based syntax and supports the entire range of possible graph structure constellations including directed, undirected, mixed graphs, hypergraphs, and application-specific attributes.

10.5 GML

Graph Modelling Language (GML) is a hierarchical ASCII-based file format for describing graphs. It has been also named Graph Meta Language.

10.6 DGS (read-only)

DGS is the native format used by the GraphStream graph rendering tool. DGS is suited to the description of dynamic graphs.

10.7 Grph binary

This format is native to the *Grph* project. It encodes graphs in a compact sequence of bytes. It is not human-readable.

10.8 Grph text

This format is native to the *Grph* project. It encodes graphs in a compact sequence of ASCII characters. It is meant to be compact and easily editable by humans.

11 Unclassified

11.1 Subgraph isomorphism

Written by Nathann Cohen

11.2 Graph label-based pattern matching

This algorithm performs graph pattern matching, on the basis of vertex/edge labels.

A graph pattern is given on the form of a set regular expressions. Each regular expression denotes a path in the graph. More precisely, a regular expression matches (or not) the concatenated labels of the elements (vertices and/or edges) found in a given path.

The algorithm performs as follows:

1. it enumerates all paths in the graph;
2. it computes the textual representation for every path;
3. it constructs a dictionary associating to every regular expression the set of paths which match it;
4. it computes all combinaisons of paths matching the set of regular expression and the sub-graph corresponding to each combinaison.

Each sub-graph has a minimal set of paths matching the input regular expressions.

11.3 Transitive closure

Written by Julien Fighera

11.4 Line graph

11.5 Pruning

11.6 Evolutionary creation of graph instances

11.7 Random-walk based centrality

Based on a random walk.

11.8 Graph isomorphism

Integration of Brendan McKay C code (Nauty).

11.9 Subgraph isomorphism

LAD is a program for solving subgraph isomorphism problem. It is written by Christine SOLNON, Professor at LIRIS, INSA (Lyon, France).

`http://liris.cnrs.fr/csolnon/LAD.html`

11.10 Full-featured subgraph isomorphism

Based on LAD. Support direction and properties.

11.11 Maximum clique

The computation of the maximum clique is delegated to program AntClique. AntClique is a program for solving maximum clique problems. It is written by Christine SOLNON, Professor at LIRIS, INSA (Lyon, France).

`http://liris.cnrs.fr/csolnon/AntClique.html`

On request of the maximum clique, Grph runs AntClique in a new process and interact with it using standard input/output streams. If the executable file of AntClique is not found (which is very likely at the first call), it will compile the source code on-the-fly.

11.12 Topological sort

11.13 Complement

The complement (or inverse) of a graph G is a graph H on the same vertices such that two vertices of H are adjacent if and only if they are not adjacent in G.

12 Set operations

Inclusion, suppression, addition, difference, union.

13 Value distribution

The result of methods returning set of values (one value per vertex) can be further computed so as to generate distribution of these values.

14 Covering and packing

14.1 Minimum Vertex Cover

The Minimum Vertex Cover problem aims at finding a set S of vertices of minimum cardinality such that each edge has at least one endpoint in S.

14.1.1 BruteForceMinimumVertexCoverAlgorithm

This algorithm simply tests the 2^n subsets of vertices of the graphs (where n is the number of vertices of graphs), and thus offers poor performance.

14.1.2 BranchingMinimumVertexCoverAlgorithm

This algorithm, of complexity $O(1.47^n)$, is a refinement of the previous one, based on the simple observation that either a vertex v belongs to the cover, or all its neighbors must belong to. It is described in details in Niedermeier's book *Invitation to fixed-parameter algorithms*, p.90.

14.1.3 NiedermeierMinimumVertexCoverAlgorithm

This is the $O(1.33^k)$ Bounded Search Tree algorithm, as explained in Niedermeier's book (see above for a reference), pp. 98–101. It offers good performances in practice.

14.2 Maximum Matching

The Maximum Matching problem aims at finding a set S of edges of maximum cardinality such that no two edges of S share a common endpoints. Edmonds has provided the well-known blossom algorithm for solving this problem, so showing it is polynomial.

14.2.1 BipartiteMaximumMatchingAlgorithm

Algorithm based on flows for the special case where the graph is bipartite.

14.2.2 LPBasedMaximumMatchingAlgorithm

Solves the Maximum Matching problem by formulating it as an Integer Program, then calling the specified solver.

14.3 Maximum Independent Set

The Maximum Independent Set problem aims at finding a set S of vertices of maximum cardinality such that at most one endpoint of each edge belongs to S. Thus, an independent set is the complement of a vertex cover, and we can use Minimum Vertex Cover algorithms to solve Maximum Independent Set problem.

14.3.1 LPBasedMaximumIndependentSetAlgorithm

Solves the Maximum Indepensent Set problem by formulating it as an Integer Program, then calling the specified solver.

14.3.2 FominGrandoniKratschMaximumindependentSetAlgorithm

This is an implementation of the algorithm proposed by Fomin, Grandoni and Kratsch for computing a maximum independent set, based on *foldable vertices* and *mirroring*, in F.V. Fomin, F. Grandoni, and D. Kratsch, *A measure & conquer approach for the analysis of exact algorithms*, Journal of the ACM (JACM), vol.56-5, p.25, 2009, ACM.

15 Coloring

15.1 Chromatic number

When used without any qualification, a coloring of a graph is almost always a proper vertex coloring, namely a labelling of the graphs vertices with colors such that no two vertices sharing the same edge have the same color. The smallest number of colors needed to color a graph G is called its chromatic number.

15.2 BipartitenessAlgorithm

Implementation of the bipartite testing algorithm. The algorithm traverses the graph with depth-first search and tries to color vertices properly with two colors. If it fails, the graph is not bipartite.

16 Coloring

16.1 OGDF

References