OMICS A Journal of Integrative Biology
Volume 7, Number 1, 2003

Graph Data Management for Molecular Biology

FRANK OLKEN

IT IS OUR VIEW that labeled directed graph data models (simple, nested, or hypergraphs) can naturally and usefully capture a wide variety of biological data and queries. We believe that development of general purpose graph data management systems (GDMSs) could become major platforms for development of a wide variety of bioinformatics database systems spanning applications from DNA sequences, chemical structure graphs, to contact graphs and biopathways. The use of a common data model and query language across numerous biological applications is very attractive from both a development and user standpoint. Considerable research and development effort are still required to extend current systems to meet modern requirements.

GRAPH DATASETS

There are numerous biological datasets which lend themselves naturally to graph data models:

- Metabolic pathways: directed cyclic (multi)-graphs
- Signaling pathways: directed cyclic graphs
- Gene regulatory networks: directed cyclic graphs
- Protein interaction networks: undirected graphs (or hypergraphs)
- Taxonomies of proteins, chemical compounds, and organisms: directed acyclic graphs
- Partonomies: directed acyclic graphs
- Topological adjacency relations: directed graphs
- Data provenance graphs: directed acyclic graphs (possibly hypergraphs)
- Chemical structure graphs: chemical bond graphs—undirected cyclic multigraphs
- Sequence data and multiple sequence alignments: linear and directed acyclic (partial order) graphs, respectively
- Contact graphs: undirected graphs to represent three-dimensional structure of proteins
- Gene clusterings: usually directed trees, directed acyclic graphs if overlapping clusters allowed
- Bibliographic citations: directed graphs—mostly acyclic
- Hypertext: directed graphs

GRAPH DATA MODELS

Graph data models considered by the DBMS and bioinformatics communities vary by the class of graphs allowed: directed versus undirected, linear, tree, DAGs, and directed cyclic graphs. A number of authors have adopted nested graph data models to allow hierarchical biopathways, chemical structure graphs encapsulated inside chemical entity nodes. The Ozsoyoglu's at CWRU have adopted a hypergraph data model for biopathways, presumably due to advantages of conciseness. Note that commonly one needs to be able

Lawrence Berkeley National Laboratory, Computational Research Division, Berkeley, California.

to specify labels on both nodes and edges of the graph data model. Often it will be useful if nodes and/or edges are typed and if the node/edge labels can be organized as taxonomies.

GRAPH DATA REPRESENTATION

There are two major approaches to the representation of graphs: edge lists and adjacency matrices. Edge list are simply lists of pairs of nodes connected by an edge. Adjacency matrices are binary matrices (for graphs), which contain a 1 for element A(i,j) if there is an edge connecting node i to node j. Edge lists are preferred for sparse graph applications. Adjacency matrices are more efficient representations for dense graphs. Most graph data managements (e.g., in the database community, etc.) employ edge list notations.

GRAPH QUERIES

In biological settings, graph queries may be constructed from compositions of the "basic" query types enumerated below. Note that most work in the database community has focused on path queries, and more recently tree queries (e.g., for XML). More general graph matching has received less attention due to potential pitfalls with computational complexity. However, general graph matching, for example, labeled subgraph isomorphism is widely used in the chemical information retrieval communities for matching chemical structure subgraphs. Graft queries include:

- Fixed path queries, with predicates on edges and/or nodes
- Recursive path queries, for example, transitive closure queries, regular expressions on paths.
- Path existence queries, for example, for subsumption tests
- Shortest path queries
- Transitive reduction queries (the inverse of transtive closure)
- Subgraph isomorphism queries (i.e., structural matching of subgraphs)
- Subgraph homomorphism queries, similar to subgraph isomorphism, except that the labels on the query subgraph nodes are generalizations of the terms used to label nodes in the database (i.e., taken from a concept lattice or taxonomy
- Connected component labeling queries
- Boolean graph queries: graph intersection, difference, sum (disjunction)
- Graph majority, at least k-of-n queries
- Graph composition: on directed graphs
- Largest common subgraph (maximum common subgraph)
- Least common ancestor query: on directed acyclic graphs
- Approximate graph matching: akin to approximate string matching
- Neighborhood queries return the subgraph comprised of all portions of a graph that are within a specified distance (usually number of edges) of a query subgraph
- k-Core queries return a subgraph all of whose nodes have degree of at least k (to other nodes in the k-core).
- Frequent subgraphs queries
- Graph characterization queries often involve the computation of various statistics over sets of nodes, edges, graphs. Examples include computing the distribution of (in/out) degrees of nodes and a variety of topological graph metrics developed in chemical graph theory.
- Graph segmentation used in image segmentation, parallel numerical linear algebra, and clustering

Many of these graph queries are motivated by interest in comparative biology over biopathways and other graph data. Comparative biology has proven to be a fruitful approach when the comparisons concern sequence data (DNA, RNA, or protein sequences) or shape comparisons (e.g., of protein structures). We anticipate that as metabolic pathways, signaling pathways, and genetic regulatory networks are determined for many organisms there will be increasing interest in a variety of comparisons of the corresponding graphs.

OPEN RESEARCH QUESTIONS

- Which graph data model should we adopt? Simple, nested, or hypergraph?
- Which types of queries should be supported? Which are necessary? Which are too hard?
- How far into network optimization computations should we go in terms of query specification and answering?
- What should the query language look like? Should it be a functional query language?
- Should we adopt a query operator algebra? What kind?
- How should we do query optimization?
- How to do query optimization and processing in various settings?
  - Main memory
  - Disk
  - Parallel machines
  - Distributed database
- What sort of additional indexing (encoding) can be created to support efficient subgraph homorphism queries, that is, efficiently perform subsumption testing?
- Could adjacency matrix graph representations be usefully employed in a graph query language?
- How do we provide extensibility in the query language and query processing system, for example, to support various sorts of additional graph computations?
- What is the role of graph grammars and other graph transformation systems in the specification and implementation of graph data management systems?
- What is the relationship between graph data management and object-oriented data management?
- Can we naturally represent three-dimensional protein structures in graph databases?
- Should graph data management be integrated in relational DBMSs? If so, how?
- Are native graph data management systems desirable? Or should graph data management be either integrated or built atop relational, XML, or object-oriented DBMSs?
- How do we handle graph queries over unbounded graphs, for example, the World Wide Web?
- What is the relative expressivity (power) and tractability (computatioanl complexity) of graph data models (query languages) and logic databases? In what contexts is one approach preferred over the other?

PRACTICAL QUESTIONS

Is there enough commonality across applications and a sufficiently large market for graph data management systems in the following for graph data managements systems to ever become a real business, rather than just an exotic boutique class of data management systems?:

- Biology
- GIS systems (street maps)
- Network topology
- Hypertext
- Transportation trip routing (e.g., airline ticketing)
- Terminology management
- Knowledge management
- Mesh data management (e.g., for finite element codes, computational fluid dynamics)

FURTHER INFORMATION

For further information, literature citations, and more detailed explanations of queries on graph data management, see the web page, http://www.lbl.gov/~olken/graphdm/graphdm.htm, for our project, "Biopathways Graph Data Manager."

ACKNOWLEDGMENTS

This work is supported by (1) The Virtual Institute for Microbial Stress and Survival at Lawrence Berkeley National Laboratory, et al., funded by the U.S. Dept. of Energy program on Genomes to Life. This work was supported by the Director, Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098; (2) Sandia Genomes to Life project on Carbon Sequestration in Synechococcus Sp.: From Molecular Machines to Hierarchical Modeling funded by the U.S. Dept. of Energy program on Genomes to Life Program; and (3) Berkeley BIOSPICE Project at the Arkin Lab at LBNL, funded by the DARPA program on Bio-Computation (BIOCOMP) of DARPA IPTO (Information Processing Technology Office).

Address reprint requests to:
Dr. Frank Olken
Lawrence Berkeley National Laboratory
Computational Research Division
Building 50B3238
1 Cyclotron Road
Berkeley, CA 94720-8147

E-mail: olken@lbl.gov