G

Gaifman-Locality

► Locality of Queries

Gazetteers

Linda L. Hill
University of California-Santa Barbara, Santa Barbara, CA, USA

Synonyms

Place names; Toponyms; Knowledge organization systems; Ontologies

Definition

A simple definition is that gazetteers are dictionaries of placenames. The digital gazetteer as a component of georeferenced information systems, however, is more formally modeled. A gazetteer is defined as a collection of gazetteer entries, each of which contains, at a minimum, the tuple *N, F, T* where *N* is a place name, *F* is a formal expression of geographic location – a *footprint*, and *T* is a place type expressed with a term (or code) from a typing scheme. Applications often require, in addition, relationships between gazetteer entries, documentation of time frames, and additional information (as described below). The gazetteer model is a type of knowledge organization system (KOS) – or ontology – which can be modified to represent other classes of spatial-temporal information, such as named time periods and named events [3].

Key Points

Gazetteers support bidirectional translation between *informal* georeferencing using names (e.g., Las Vegas) and *formal* georeferencing using coordinates (e.g., 36°10′30″N, 115°08′11″W) or similar mathematical reference within a geospatial framework. The need for such translation is common, For example, to calculate routes and distances and to support information retrieval by either place names or coordinates. A common use for gazetteers is to answer the "where is" question: "where is Baltimore?" can be answered by a map display showing the locations of places called "Baltimore" in the world.

In addition to place names and geospatial location, a third basic component of a gazetteer entry is a classification according to a typing scheme; that is, assigning a type term to a place to indicate that it is a populated place, a river, an island, a country, a bank, etc. There is no single typing scheme for named places. Instead, there are multiple typing schemes, usually unique for a particular application.

All of the descriptive data associated with named places have time dimensions. To deal with these temporal dimensions adequately, a gazetteer data structure must incorporate time ranges for places as well as for most of the elements of description.

Since different names can be used for the same place (e.g., in multiple languages), gazetteers must support multiple names and, ideally, document the source and time frame of each name. Supporting multiple footprints and information about a place from multiple sources is also needed for some applications.

A gazetteer is a collection of gazetteer entries [5]. Inherent spatial relationships exist between gazetteer entries. It can be determined which places are located within the footprint of another place – at least, theoretically. Actually, the ability to do this adequately depends on the quality of the footprints. Explicit relationships between gazetteer entries can also be added to the descriptive information. It can be documented, for example, that Chicago *is part of* Illinois and that Illinois *is part of* the US or that Topeka *is capital of* Kansas.

The International Organization for Standardization (ISO), the Open Geospatial Consortium, and the Alexandria Digital Library at the University of

California, Santa Barbara, have published gazetteer standards, protocols, and place typing schemes; see references [2,4–6].

Cross-references

- ▶ Browsing in Digital Libraries
- ▶ Georeferencing
- ▶ GIS
- ▶ Searching Digital Libraries

Recommended Reading

1. Alexandria Digital Library. Guide to the ADL Gazetteer Content Standard. University of California, Santa Barbara, CA, 2004.
2. Hill L.L. Feature Type Thesaurus. Alexandria Digital Library, 2002.
3. Hill L.L. Georeferencing: The Geographic Associations of Information. MIT Press, Cambridge, MA, 2006.
4. International Organization for Standardization (ISO) – Technical Committee 211. Geographic Information – Spatial Referencing by Geographic Identifiers (ISO 19112:2003), 2003.
5. Janée G. and Hill L.L. ADL Gazetteer Protocol, Version 1.2. Alexandria Digital Library, University of California, Santa Barbara, CA, 2003.
6. Open Geospatial Consortium Inc. Gazetteer Service Profile of the Web Feature Service Implementation Specification. OGC 05–035rl, version 0.9.1, 2006.

Gene Expression Arrays

Mehmet M. Dalkiliç
Indiana University, Bloomington, IN, USA

Definition

A gene expression array (or DNA microarray) is a miniature, massively parallel, and semi-automated process that measures levels of gene products (or mRNA or transcripts). The DNA of an organism (cell or cells) is a set of instructions used to direct its existence. A genetic message is transcribed from DNA into mRNA (messenger RNA), then typically translated into protein. By measuring the levels of mRNA, an indirect glimpse can be taken into the functioning of a cell. The relative and absolute levels of the transcripts are called expressions, and this instantaneous set of expressions is called the expression profile. Profiles can be used, for example, to indicate a disease state.

Historical Background

DNA and RNA are bio-molecules that *hybridize* bonding due to complementary structure. Hybridization has been used for several decades as means of identifying DNA and RNA present in an organism (cell or cells). Until microarray technology was invented, utilizing hybridization was slow and could only be perfomed on a relatively small scale. The microarry process is often associated with the Pat Brown laboratory in 1995 [6] and by Affemetrix [5].

Foundations

The fundamental purpose of gene expression arrays (or DNA microarrays) is to understand how an organism functions through measuring levels of proteins – molecules that perform a vast array of functions necessary for the life of the organism. Together with its genome (the DNA of an organism), DNA microarrays provide a global perspective into how regulation takes place, in other words, how the organism makes use of its DNA. The measurements are not directly made on the protein levels themselves, but intermediaries between the DNA and protein called RNA. Specifically, messenger RNA (mRNA) is first transcribed from the DNA, then translated into protein. The basic mechanism underlying microarray technology is *hybridization* – the chemical pairing of complementary molecules. DNA can hybridize to itself and to RNA, and this property has been used for decades in Southern and Northern blotting that measures amounts of hybridized DNA and RNA, respectively, typically tagged through radioactive labeling. The weakness of these two techniques is scale and time: only a small number of proteins can be measured requiring the period of a day or so. These blotting techniques have been miniaturized, made massively parallel, and semi-automated, so that $O(10^5)$ to $O(10^6)$ experiments can be conducted at a time. In DNA microarrays single stranded fragments of DNA are affixed to a substrate (called probes) in a grid and washed with a solution containing fluorescently tagged fragments of DNA (called targets). The concentration of DNA labeled fragments affects the likelihood of hybridizing to the probes. The intensities of each point on the grid are measured and result in a table of floating point numbers. It is from the intensity of labels that the levels of proteins are to be inferred. There are essentially two kinds of DNA microarrays: cDNA (spotted arrays) and high-density oligonucleotide arrays (oligos). In the former, fragments of a genome are used as probes. Oligos are synthesized sequences of nucleotides. cDNA arrays allow for matching against the genome, and Oligos for particular sequences of interest. Because the company Affemetrix developed the oligos, these are often referred to as Affy chips (or Affy arrays). The data produced from arrays is a very large set of intensities of electromagnetic radiation. Each intensity is associated with a two-dimensional location on the chip is called a spot. Microarrays are noisy, low resolution, and inconsistent; however, as the technology matures, there have been marked improvements. The data must be heavily pre-processed before analysis. This includes correcting for experimental design, biological factors, and specific mechanical idiosyncrasies themselves. One of the most significant pre-processing steps is called "normalization" which is an attempt to make different microarray data comparable. Currently, a consensus on how to achieve normalization does not exist, and a number of approaches can be taken. Extensive libraries exist in **R** (http://www.r-project.org) a feely available computing environment for statistical analysis; in particular, the OLIN [2] and Bioconductor packages [4] include normalization and dye correction. The main objective in microarray anaylsis is discovery patterns and outliers. Patterns are, crudely, sets of gene products that possess some discernable functional relationship. Typically, the relationship is between two expressions and linearity is examined. In particular, Pearson's correlation is often used to gauge the linearity of genes. Pearson's correction denoted $\rho(\cdot,\cdot)$ is a real-valued measurement on the interval $[-1,1]$ over two sets of data, X,Y. The score $|\rho(X, Y)| \approx 1$ if there is either a positive or negative unity if a strong linear relationship exists, tending toward zero otherwise. A zero value for Pearson's only indicates that linearity does not exist. Database practitioners should realize the microarray is significant when it can be thoroughly studied with extensive biological annotations – it is much more than a table of spots.

Key Applications

There are key applications for both bioinformatics and database. To give some indication of the amount of interest in the area, the number of publications found in PubMed (http://www.pubmed.gov) by searching with terms "`microarray`" and "`2007`" yields well over 10,000 entries (executed 01/20/2008). For bioinformatics, microarrays promise to yield cell states that can help determine function, disease states (as mentioned previously), guide areas of discovery (e.g., drugs). From a database perspective, there is an incredible wealth of data that needs to be managed. However, the empirical and complex nature of the processes underlying the microarray belie its apparent simplicity – usually rendered pictorially as an attractive rectangle grid of colored circular sports that range from green to yellow to red. Although no ANSI standard exists for microarrays, like much of bioinformatics there are different popular standards vying for dominance. Of these, the most well-known is MIAME (the minimum information about a microarray experiment) that specifies elements of experiments. Since this specification is at least recognized in name, it will briefly highlight its components: (i) Array information including platform (spotted, etc.), size and number of spots, protocols; (ii) Reporter information (sequences on the slide itself); (iii) Gene information; (iv) Information about control array control elements. Often cluster analysis is done on microarrays, rows are gene products and columns are experiment conditions. The clustering results in a tree (dendogram) that depicts the most similar genes on the set of conditions. A formal language for accessing, managing, creating these large data sets does not exist.

Queries associated with arrays vary with the intent of the user. This is significant because a good number of these queries rely on data completely external to the system. For example, a typical query might be, "What are the sequence annotations of gene g_1, g_2, g_3?" For this query, the system must link to sequence annotation databases to retrieve the information. The challenge is that there are (again) no official standards and that the data, contained in flat files or even XML, must be parsed to extract the meaningful information desired. To answer the growing needs to understand a group was formed near 2000 called the Microarray and Gene Expression Database (http://www.mged.org). The Microarray and Gene Expression Data (MGED) Society is a global consortium of scientists who are striving to provide some coherence in how to make data exchange easier. The scope of MGED includes other genome-wide data and so, provides additional information particular to data like binding. Although the MGED is working to establish standards for management and so forth, it is unclear how widespread the recognition and adoption is. As mentioned with the example query, one unusual aspect of managing gene expression data is associating it with other data housed and managed in other systems. The MGED provides a few standards: MIAME, MAGE-TAB, MAGE-ML, MGED, Ontology; it also provides information about the current status of these representations.

One of the most significant elements of database design for array databases is the ability to exchange and retrieve "foreign" data – data housed in different forms in different repositories likely due to the nature of the biological questions being asked. To this end, MAGE and MAGE-ML, are means of facilitating data exchange, the latter XML in spirit. MAGE is more popular and does associate with the MIAME standard, but the associations are not one-to-one. One significant difference between MIAME and MAGE is the latter provides elements of provenance and security, while MIAME has no mechanism at all.

ArrayExpress, a public European repository for array data, is housed at the European Bioinformatics Institute (EBI) and seems to be aiming more toward a data warehousing approach, archiving data, but allowing limited keyword searches, etc. accession number or author.

A significant challenge to management of array data is the dynamic nature of information. Since array data maps back to genomes, as genome drafts are updated, array data must reflect these changes. There is genomic draft versioning to help remap gene products to the updated genomic locations. To give an idea of the complexity, often drafts result in genes being split, coalesced, removed, or changed in length. This directly affects the interpretation of the results of array.

Semantics of terms in array database are, as described above, often imprecise. One approach that biologists have begun adopting is the use of ontologies. The most well known example in biology is the Gene Ontology [3] (GO) (*http://www.geneontology.org/*). By fixing a common vocabulary and set of relationships GO has become a common tool in analysis.

Future Directions

Microarrays are only one of many so-called high-throughput technologies that produce enormous amounts of data – these can easily range in size to terabytes. Many problems biologists have been able to effectively solve on small scales, do not scale with the data. There is much work to be done in how to analyze such a large and growing corpus of data. Another aspect is how to integrate the different types of high-throughput. Systems biology is a recent discipline that attempts to semantically merge disparate types of data to form global perspectives that no one piece of data would contain. One of the standard structures used in microarrays is the directed graph. Again, polynomial complexity algorithms become infeasible with such a large amount of data. Bayesian Belief Nets have recently become popular in analyzing these large graphs (or networks).

Data Sets

There are public data sets available through the internet. Some of the better known include:

NAME	URL
ArrayExpress	http://www.ebi.ac.uk/microarray-as/aer/
GEO	http://www.ncbi.nlm.nih.gov/geo/
Stanford Microarray Database	http://genome-www5.stanford.edu/

Cross-references

- ▶ Annotation
- ▶ Approximate XML Querying
- ▶ Biological Networks
- ▶ Biological Sequence
- ▶ Data Mining
- ▶ Data Provenance
- ▶ Data Structures and Models for Biological Data Management
- ▶ Data Visualization
- ▶ Graph Database
- ▶ Graph Management in the Life Sciences
- ▶ Information Integration
- ▶ Ontologies
- ▶ Ontologies and Life Science Data Management
- ▶ Ontologies in Scientific Data Integration
- ▶ Ontology
- ▶ Semantic Data Integration for Life Science Entities
- ▶ Text Mining of Biological Resources
- ▶ Uncertainty and Data Quality Management
- ▶ Visual Data Mining
- ▶ XML

Recommended Reading

1. Drăghici S. Data Analysis Tools for DNA Microarrays. Chapman & Hall/CRC, London, 2003.

2. Futschik M.E. and Crompton T. OLIN: optimized normalization, visualization and quality testing of two-channel microarray data. Bioinformatics, 21(8):1724–1726, 2005.
3. Gene Ontology Consortium. Creating the gene ontology resource: Design and implementation. Genome Res., 11:1425–1433, 2001.
4. Gentleman R., Carey V., Bates D., Bolstad B., Dettling M., Dudoit S., Gautier L., Ge Y., Gentry J., Hornik K., Hothorn T., Huber W., Iacus S., Irizarry R., Leisch F., Li C., Maechler M., Rossini A., Sawitzki G., Smith C., Smyth G., Tierney L., Yang J., and Zhang J. Bioconductor: open software development for computational biology and bioinformatics. Genome Biol., 5(10):R80,2004.
5. Lockhart D.J., Dong H., Byrne M.C., Follettie M.T., Gallo M.V., Chee M.S., Mittmann M., Wang C., Kobayashi M., Horton H., and Brown E.L. Expression monitoring by hybridization to high-density oligonucleotide arrays. Nat. Biotechnol., 14:1675–1680, 1996.
6. Schena M., Shalon D., Davis R.W., and Brown P.O. Quantitative monitoring of gene expression patterns with complementary DNA microarray. Science, 270:467–460, 1995.

Generalisation

► Abstraction

Generalization of ACID Properties

BRAHIM MEDJAHED[1], MOURAD OUZZANI[2], AHMED K. ELMAGARMID[2]
[1]The University of Michigan–Dearborn, Dearborn, MI, USA
[2]Purdue University, West Lafayette, IN, USA

Synonyms

Advanced transaction models; Extended transaction models

Definition

ACID (Atomicity, Consistency, Isolation, and Durability) is a set of properties that guarantee the reliability of database transactions [2]. ACID properties were initially developed with traditional, business-oriented applications (e.g., banking) in mind. Hence, they do not fully support the functional and performance requirements of advanced database applications such as computer-aided design, computer-aided manufacturing, office automation, network management, multidatabases, and mobile databases. For instance, transactions in computer-aided design applications are generally of long duration and preserving the traditional ACID properties in such transactions would require locking resources for long periods of time. This has lead to the generalization of ACID properties as Recovery, Consistency, Visibility, and Permanence. The aim of such generalization is to relax some of the constraints and restrictions imposed by the ACID properties. For example, visibility relaxes the isolation property by enabling the sharing of partial results and therefore promoting cooperation among concurrent transactions. Hence, the more generalized ACID properties are, the more flexible the corresponding transaction model will be.

Key Points

ACID is an important concept of database theory. It defines four properties that traditional database transactions must display: Atomicity, Consistency, Isolation, and Durability. *Atomicity* states that transactions must follow an "all or nothing" rule. Either all of the changes made by a transaction occur, or none of them do. The two-phase commit protocol (2PC) is generally used in distributed databases to ensure that each participant in a transaction agrees on whether or not the transaction should be committed. *Consistency* means that transactions always operate on a consistent view of the database and leave the database in a consistent state. A database is said to be consistent as long as it conforms to a set of invariants, called *integrity constraints*. *Isolation* gives the illusion that each transaction is executed alone. It ensures that the effects of a transaction are invisible to other concurrent transactions until that transaction is committed. Concurrency control protocol are generally implemented in database systems to preserve this property. *Durability* states that once a transaction is committed, its effects are guaranteed to persist even in the event of subsequent failures. This is usually achieved using database backups and transaction logs.

The limitations inherent to the original ACID properties and the peculiarities of advanced database applications has lead to the generalization of ACID properties Recovery, Consistency, Visibility and Permanence. *Recovery* refers to the ability to take the database to a state that is considered correct in case of failure. *Consistency* refers to the correctness of the state of the database that a committed transaction produces. *Visibility* refers to the ability of one transaction to see the results of another running transaction.

Permanence refers to the ability of a transaction to record its results in the database. The flexibility of a transaction model depends on the way ACID properties are generalized. An extensive coverage of advanced transaction models is presented in [1]. *Sagas*, *Nested Transactions*, and *Flex* are representative examples of models that generalize ACID properties while *ACTA* is an example of a formal framework to express these models and reason about them.

A *Saga* is a chain of transactions that is itself atomic. Isolation is relaxed at the level of a Saga and visibility is permitted at the component transaction boundaries. A saga itself is atomic but because of the relaxed visibility, it supports semantic consistency. That is, each transaction in the chain is assumed to have a semantic inverse, or compensation, transaction associated with it. If one of the transactions in the saga fails, the transactions are rolled back in the reverse order of their execution. Committed transactions are rolled back by executing their corresponding compensation transactions.

Nesting allows concurrency within a transaction and provides fine-grained and hierarchical control for failure handling. The original nested transaction model, which supports only closed sub-transactions, was extended to include open sub-transactions. Closed sub-transactions may not support consistency and durability. A closed sub-transaction commits its results to its parent. These partial results are externalized only after the top (root) transaction commits, thus ensuring atomicity and isolation of the whole transaction. Because of its relaxed visibility, open sub-transactions directly externalize their results and expose them to other transactions.

The *Flex Transaction Model* has been proposed to generalize ACID properties in multidatabase systems. It relaxes the atomicity and isolation properties of nested transactions to provide users increased flexibility in specifying their transactions. A Flex transaction may proceed and commit even if some of its sub-transactions fail. It also allows the specification of dependencies on sub-transactions as internal or external dependencies. Internal dependencies define the execution order of sub-transactions, while external dependencies define the dependency of a sub-transaction execution on events (such as the start/end execution times) that do not belong to the transaction. The Flex model also enables users to control the isolation granularity of a transaction through the use of compensating sub-transactions.

ACTA is a framework that facilitates the formal description of properties of extended transaction models. It defines constructs that facilitate the synthesis of extended transaction models by tailoring/combining existing models or starting from first principles. Different notions are introduced in ACTA to enable the generalization of ACID properties from different perspectives. For instance, the notion of delegation allows transactions to selectively abort some of the operations it has performed and yet commit.

Cross-references

▶ ACID Properties
▶ Concurrency Control
▶ Distributed
▶ Extended Transaction Models
▶ Extended Transaction Models and the ACTA Framework
▶ Parallel and Networked Databases
▶ Serializability
▶ System Recovery
▶ Transaction Management
▶ Two-Phase Commit

Recommended Reading

1. Elmagarmid A.K. (ed.). Database Transaction Models for Advanced Applications. Morgan Kaufmann, Los Altos, CA, 1992.
2. Gray J. and Reuter A. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, Los Altos, CA, 1993.

Generalized Search Tree

Joseph M. Hellerstein
University of California-Berkeley, Berkeley, CA, USA

Synonyms

GiST; GIST

Definition

The Generalized Search Tree (GiST) is an extensible, disk-based index structure for large data sets, enabling the easy design and implementation of domain-specific index structures. The GiST provides a general-purpose implementation of many of the difficult systems issues inherent in indexing (data storage and access, search, concurrency and recovery), with a compact extensibility interface sufficient for the specification of the domain-specific, algorithmic aspects of indexing (clustering of data into pages, labeling of subtrees, and prioritization of the search frontier).

Historical Background

A key research challenge in database systems in the 1980s and 1990s was to support an *extensible* set of abstract data types, beyond the alphanumeric types typically used in business data processing. One critical component of database extensibility is the ability to easily add new *access methods* customized to specific data types and query operators. Ideally, extensibility interfaces should enable the programmer extending the DBMS to focus on their domain expertise (e.g., geographic information, bioinformatics, etc.), and not have to reason about database systems issues like storage, buffering, query optimization, concurrency or recovery.

The Postgres system provided one of the first truly extensible access method interfaces, which allows new access methods to be registered with a query planner and executor. However, this interface leaves the implementation of data storage, concurrency and recovery in the hands of the extension programmer, which makes the task of writing a new access method extremely difficult, and results in inelegant software that replicates complex functionality across different index implementations.

The GiST design was intended to take care of as many generic database systems issues as possible, while still enabling a wide range of indexes to be easily written within its framework. At heart, the GiST is a generalization of the B+-tree, and one of its contributions is to demonstrate the degree to which many subsequent index inventions are specific variations of the basic B+-tree ideas.

In principle, the GiST can be used for any indexing *workload*, as defined by a family of query predicates over a particular data domain. This raises theoretical questions of both lower and upper bounds: which workloads are efficiently indexable, and how close to optimal is the performance of a particular index design on a given workload? The first of these questions inspired the *Theory of Indexability*; the latter led to a set of empirical performance metrics, which have been embodied in the *Access Method DeBugger (amdb)* toolkit.

Foundations

Like the B+-tree, the GiST is a height-balanced tree of variable fan out, with data stored at the leaves, and *search keys* associated with pointers in the internal nodes. The GiST search keys are the center of its extensibility API: a new index is implemented via a new search key class and associated methods. Search keys are parameterized by four user-defined methods:

1. The `consistent` method of a search key *k* takes a query predicate *p*, and returns *false* if it is not possible for *k* and *p* to describe the same data items. The index search algorithm only traverses a subtree if the consistent method returns true for the search keys above it.
2. The `union` method takes a set of search keys from a child node, and produces a single search key to be placed in the parent. This is used to update search keys whenever the contents of the tree change.
3. The `penalty` method of a search key takes a new data item to be inserted, and returns a cost for the insertion of that item into the subtree labeled by that search key. This guides the index insertion algorithm's descent to a leaf, affecting the clustering of data in the tree.
4. The `pickSplit` method takes a set of search keys, and partitions them into two subsets; this is used for choosing how to split an overflowing page during insertion into the tree. This method also affects the clustering of data in the tree.

In its original conception, the GiST was designed to support selection predicates, via search keys forming a set-containment hierarchy. Subsequently, it was observed that the GiST can naturally support richer queries, including near-neighbor search, as well as statistical methods for approximate query answering that treat the search keys as a multi-resolution synopsis of the data.

Key Applications

GiST has been implemented in a number of database systems and projects, most notably in the Informix Universal Server, and the open-source PostgreSQL system, both of which implement the full GiST concurrency and recovery algorithms.

Currently, GiST serves as the default framework for spatial and text indexing in PostgreSQL, via extensions corresponding to R-trees for spatial data, and Russian-Doll (RD) Trees for text data. The open-source

PostGIS Geographic Information System relies upon the PostgreSQL GiST implementation for its spatial indexing.

In addition to their widespread use for spatial and text workloads, GiST-based indexes have been used for applications in image retrieval, astronomy data, and genomic data.

URL to Code

1. http://www.postgresql.org
2. http://gist.cs.berkeley.edu

Cross-references

- ▶ B-Tree
- ▶ B+-Tree
- ▶ Rtree
- ▶ M-Tree
- ▶ Extensional Relational Database (ERDB)
- ▶ Postgres
- ▶ PostgreSQL
- ▶ Geographic Information System
- ▶ Spatial Indexing Techniques
- ▶ Informix Indexability Theory
- ▶ Index Concurrency
- ▶ Index Recovery

Recommended Reading

1. Aoki P.M. Generalizing 'search' in generalized search trees. In Proc. 14th Int. Conf. on Data Engineering, 1998.
2. Aoki P.M. How to avoid building dataBlades that know the value of everything and the cost of nothing. In Proc. 11th Int. Conf. on Scientific and Statistical Database Management, 1999, pp. 122–133.
3. Aref W.G., Ilyas I.F. SP-GiST: an extensible database index for supporting space partitioning trees. J. Intell. Inf. Syst., 17(2/3):215–240, 2001.
4. Hellerstein J.M., Koutsoupias E., Miranker D., Papadimitriou C., and Samoladas V. On a model of indexability and its bounds for range queries. J. ACM, 49(1):35–55, 2002.
5. Hellerstein J.M., Naughton J.F., and Pfeffer A. Generalized search trees for database systems. In Proc. 21st Int. Conf. on Very Large Data Bases, 1995, pp. 562–573.
6. Hellerstein J.M. and Pfeffer A. The RD-Tree: An Index Structure for Sets. University of Wisconsin Technical Report #1252, October 1994.
7. Kornacker M. High-performance generalized search trees. In Proc. 25th Int. Conf. on Very Large Data Bases, 1999.
8. Kornacker M., Mohan C., and Hellerstein J.M. Concurrency and recovery in generalized search trees. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 62–72.
9. Kornacker M., Shah M.A., and Hellerstein J.M. Amdb: a design tool for access methods. Data Eng. Bull., 26(2):3–11, June 2003.
10. Thomas M.C. and Hellerstein J.M. Boolean bounding predicates for spatial access methods. In Proc. Int. Conf. on Database and Expert Systems Applications, 2002, pp. 925–934.
11. Stonebraker M. Inclusion of new types in relational data base systems. In Proc. 2nd Int. Conf. on Data Engineering, 1986.

Generative Models

▶ Language Models

Genetic Algorithms

Colin R. Reeves
Coventry University, Coventry, UK

Synonyms

Evolutionary algorithms; Evolutionary computation

Definition

A genetic algorithm (GA) is one of a number of heuristic techniques that attempt to find high-quality solutions to large and complex optimization problems. The term *evolutionary algorithm* is sometimes used synonymously, but is generally used to denote a rather wider class of heuristics. All such algorithms use the notion of a sequence of cycles that employ mutation of, and subsequent selection from, a population of candidate solutions. While these features are also found in a GA, its most distinctive characteristic is the use of *recombination* (or *crossover*) to generate new candidate solutions. A secondary idea found in many, but not all GAs, is the existence of an encoding function that maps the original optimization problem into a space that is hoped to be more congenial to the application of the GA operators.

Historical Background

The term *genetic algorithm* was first used by John Holland, whose book [8], first published in 1975, was instrumental in creating what is now a flourishing field of research and application that goes much wider than the original GA. The area covers topics such as evolution strategies (ES), evolutionary programming (EP), artificial life (AL), classifier systems (CS), genetic programming (GP), and now the concept of evolvable hardware. These related fields of research are often grouped under the heading of *evolutionary computation* or *evolutionary algorithms* (EAs).

While Holland's influence in the development of the topic has been immense, from an historical perspective it is clear that several other scientists with different backgrounds were also involved in proposing and implementing similar ideas. In the 1960s in Germany, Rechenberg and Schwefel developed the idea of the *Evolutionsstrategie* (in English, *evolution strategy*), while – also in the 1960s – Lawrence Fogel and others in the USA implemented a similar idea that they called *evolutionary programming*. What these proposals had in common was their use of the ideas of mutation and selection that lie at the center of the neo-Darwinian theory of evolution. Relatively little attention was given to the use of recombination – the idea later placed at the heart of GAs by Holland. However, although some promising results were obtained, as an area of research the topic of evolutionary computing did not really catch fire. There may be many reasons for this, but the most obvious one is that the techniques tended to be computationally expensive – unless the applications were trivial – and far beyond the capabilities of the computers of the day. Nevertheless, the work of these early explorers is fascinating to read in the light of our current knowledge. David Fogel (son of one of these pioneers) has tracked down and collected some of this work [6].

The last 30 years has seen an explosion of interest in GAs and their applications, and there is now enormous literature on variations of the basic algorithm and developments in application areas. De Jong [4] – another early pioneer, particularly in relation to the GA's utility as an optimizer – has recently surveyed the most important developments from a unified perspective.

Theoretical analysis of GAs, and of EAs in general, has lagged far behind the development of novel practical applications. Holland's original work gave an account based on his idea of a *schema* that raised hopes that the GA's power was significantly greater than that of other heuristics. Sadly, these hopes have not been realized, and the rise of 'No-Free-Lunch' arguments [13] has put paid to the idea of some universal optimization algorithm that outperforms all others. In the case of GAs, Holland's schema-based theory has proved much less relevant than early accounts suggested. Several other theoretical approaches have been tried, with some success in accounting for large-scale GA behaviour, but something truly comprehensive is still awaited. Reeves and Rowe [10] provide the most recent description of the state of GA theory.

Foundations

Optimization problems arise in many forms, but the most obvious dichotomy is between continuous and discrete (or combinatorial) optimization. While GAs have been used successfully for continuous problems, the evolution strategy approach – although employing similar principles – is on the whole better suited to continuous problems. What is usually in mind for the application of a GA is the following:

Given a discrete search space $\mathcal{V}$, and a function

$$g : \mathcal{V} \mapsto \mathbb{R},$$

find

$$\arg\min_{v \in \mathcal{V}} g.$$

From a mathematical perspective, the decision variables of this problem are encoded as strings or vectors where each component is a symbol from an alphabet $\mathcal{A}$. These strings are then concatenated to provide a *chromosome*. That is, the vector v is represented by a string x of symbols drawn from $\mathcal{A}$, using a mapping

$$c : \mathcal{A}^{\ell} \mapsto \mathcal{V}.$$

where ℓ, the length of the string, depends on the dimensions of both $\mathcal{V}$ and $\mathcal{A}$. The elements of the string correspond to "genes", and the values those genes can take to "alleles". The position of a gene in the string is its "locus". The encoding function c is often designated as the *genotype–phenotype mapping*. The space $\mathcal{A}^{\ell}$ – or, quite often, some subspace $\mathcal{X}$ of it – is known as the genotype space, while the space $\mathcal{V}$, representing the original definition of the problem, is the phenotype space.

Thus the optimization problem becomes one of finding

$$\min_{x \in \mathcal{X}} g(\boldsymbol{x}),$$

where (with a slight abuse of notation)

$$g(\boldsymbol{x}) = g(c(\boldsymbol{x})).$$

In practice the distinction between genotype and phenotype is often blurred, and $g(\boldsymbol{v})$ or $g(\boldsymbol{x})$ is used according to context. Finally, a monotonic transformation of g

is often used to provide what is usually termed a *fitness function*, which can be regarded as

$$f : \mathcal{X} \mapsto \mathbb{R}^+,$$

so that fitness is always a positive value. It should also be noted that the genotype search space $\mathcal{X} \subseteq \mathcal{A}^\ell$, which implies that some strings may represent invalid solutions to the original problem. This can be a source of difficulty for GAs, since GA operators may generate new strings in $\mathcal{A}^\ell$ that are not in $\mathcal{X}$. The choice of encoding is usually dictated by the type of problem that is being investigated. Examples and further details on such matters can be found in standard references such as [4] or [10].

With this background in mind, a basic version of a GA can now be presented as in Fig. 1.

Initial populations are usually generated randomly, although there are many variations that incorporate some aspects of problem-specific knowledge. Given a population, the next step is to choose parent chromosomes for reproduction. This is generally carried out by some means of *fitness-proportional* selection. Implementations of this idea vary widely: roulette-wheel, rank-based and tournament selection schemes are all popular and have associated advantages and disadvantages.

Reproduction takes place using crossover and mutation. Having selected parents in some way, they must be recombined – the process called *crossover*. This is simply a matter of replacing some of the alleles in one parent by alleles of the corresponding genes of the other. Suppose there are two strings ***a*** and ***b***, consisting of seven symbols, i.e.,

$$(a_1, a_2, a_3, a_4, a_5, a_6, a_7) \quad \text{and} \quad (b_1, b_2, b_3, b_4, b_5, b_6, b_7),$$

each representing a possible solution to a problem. To implement *one-point* crossover a *crossover point* is chosen at random from the numbers 1,...,6, and a new solution produced by combining the pieces of the original "parents". For instance, if the crossover point was 2, then the "offspring" solutions would be

$$(a_1, a_2, b_3, b_4, b_5, b_6, b_7) \quad \text{and} \quad (b_1, b_2, a_3, a_4, a_5, a_6, a_7).$$

Many other forms of crossover have been invented, with varying degrees of success.

The concept of mutation is even simpler than crossover: a gene (or subset of genes) is chosen randomly and the allele value of the chosen genes is changed. This can easily be represented as a bit-string mask such as

```
0100010
```

generated using a Bernoulli distribution – i.e., at each locus there is a small probability μ of a mutation occurring (indicated in the mask by a `1`). The above example would thus imply that the second and sixth genes are assigned new allele values.

Finally, a new population has to be selected from the joint set of old and newly generated members. Again there are many ways of carrying this out. At the extremes, there are the generational and steady-state approaches. In the first of these, the old population is completely replaced by the new. In the second, a single new candidate is generated, and some member of the old population (often the least fit one) is displaced by it. Again, many other techniques have been tried.

This is an extremely condensed overview of the possibilities available for implementing a GA. A more comprehensive picture of some of the most popular ideas, and a discussion of their effects on GA performance, can be found in [4] or [10].

```
Choose an initial population of chromosomes;
while termination condition not satisfied do
    repeat
        if crossover condition satisfied then
        {select parent chromosomes;
        choose crossover parameters;
        perform crossover;}
        if mutation condition satisfied then
        {select chromosome(s) for mutation;
        choose mutation points;
        perform mutation;}
        evaluate fitness of offspring
    until suffcient offspring created;
    select new population;
endwhile
```

Genetic Algorithms. Figure 1. A genetic algorithm template. This is a fairly general formulation, accommodating many different forms of selection, crossover and mutation. It assumes user-specified conditions under which crossover and mutation are performed, a new population is created, and whereby the whole process is terminated.

Key Applications

The number of NP-hard combinatorial optimization problems continues to grow, and heuristic methods offer the only realistic route to high-quality solutions. The use of GAs is thus motivated by the existence of computational complexity in many aspects of technological development. In the area of database systems, there were early applications to problems of database design [3] and query optimization [11]. Other areas of interest include data clustering methods [7] and data partitioning [2,1]. Genetic algorithms have also been used for rule discovery in data mining applications, seminal work in this area being reported in [5]. Data reduction by means of instance selection using GAs for other techniques such as decision trees, neural nets or nearest neighbour methods has also been reported [9]. Also, several companies claim – without revealing very many details – to use GAs in spam detection software. A recent published example of such work is [12].

Experimental Results

Given the lack of authoritative theoretical models of GAs, assessment of their performance in any individual case is necessarily experimental. Standard methods of evaluation include the *best-so-far curve* – a time series plot of the fitness of the best solution found so far (y-axis) against the number of generations or the number of individuals generated. This is sometimes called the *offline performance* measure, in contrast to the *online performance* measure, which is a time series plot of current (average) population fitness.

Another aspect of experimental evaluation is assessment of convergence – or equivalently, of population diversity. This can be assessed simply by plotting a time series of the coefficient of variation of the current population fitnesses, or more expensively, by means of a locus-wise count of allele percentages in successive populations.

Since GAs are stochastic algorithms, it is obvious that results will differ from one run of the algorithm to the next. In any experimental work it is essential to apply the algorithm several times in order to have some confidence that the results obtained are not freakishly good (or bad).

Cross-references

▶ Data Clustering
▶ Database Design
▶ Data Mining
▶ Query Optimization
▶ Spam Detection

Recommended Reading

1. Acar A.C. and Motro A. Intensional encapsulations of database subsets via genetic programming. In database and expert systems applications, K.V. Andersen, J. Debenham and R. Wagner (eds.). Lecture notes in computer science 3588. Springer, Berlin, 2005, pp. 365–374.
2. Cheng C.H., Lee W.K., and Wong K.F. A genetic algorithm-based clustering approach for database partitioning. IEEE Trans. Syst. Man Cybernet., 32:215–230, 2002.
3. Corcoran A.L. and Hale J. A genetic algorithm for fragment allocation in a distributed database system. In Proc. 1994 ACM Symp. on Applied Computing, 1994, pp. 247–250.
4. De Jong K.A. Evolutionary Computation: A unified approach. MIT Press, Cambridge, MA, 2006.
5. Flockhart I.W. and Radcliffe N.J. A genetic algorithm-based approach to data mining. In Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, 1996, pp. 299–302.
6. Fogel D.B. Evolutionary Computation: The Fossil Record. IEEE Press, New York, 1998.
7. Hall L.O., Özyurt I.B., and Bezdek J.C. Clustering with a genetically optimized approach. IEEE Trans. Evol. Comput., 3:103–112, 1999.
8. Holland J.H. Adaptation in Natural and Artificial Systems, 2nd edn. MIT Press, Cambridge, MA, 1992.
9. Reeves C.R. and Bush D.R. Using genetic algorithms for training set selection in RBF networks. In Instance selection and construction for data mining. H. Liu and H. Motoda (eds.) Kluwer, Dordecht, 2001, pp. 339–356.
10. Reeves C.R. and Rowe J.E. Genetic Algorithms: Principles and Perspectives. Kluwer, Dordecht, 2002.
11. Wang J.C., Horng J.T., and Liu B.J. A genetic algorithm for set query optimization in distributed database systems. IEEE Conf. Syst. Man Cybernet., 3:1977–1982, 1996.
12. Wang H.B., Yu Y., and Liu Z. SVM classifier incorporating feature selection using GA for spam detection. In Proc. Int. Conf. embedded and ubiquitous computing, 2005, pp. 1147–1154.
13. Wolpert D.H. and Macready W.G. No free lunch theorems for optimization. IEEE Trans. Evol. Comput., 1:67–82, 1997.

Geographical Information Retrieval

Christopher B. Jones[1], Ross S. Purves[2]
[1]Cardiff University, Cardiff, UK
[2]University of Zurich, Zurich, Switzerland

Definition

The provision of facilities to retrieve and relevance rank documents or other resources from an unstructured or partially structured collection on the basis of queries specifying both theme and geographic scope.

Historical Background

Geographical information associates things and events with locations. Since everything one does and everything that happens on Earth happens somewhere, there is a vast amount of information that can be regarded as geographical in some sense. This information is represented within a variety of media, including digital and analogue versions of text documents, images, videos and maps, as well as the structured spatial databases that are employed within most geographical information systems (GIS). In text documents the association with location is often qualitative with place names and natural language terminology (such as in, at, and near) that provide spatial context. This is in contrast to the use in GIS of, for example, digital topographic datasets in which location is described quantitatively with numerical coordinates. Over the past few decades, information technology for accessing geographical information has focused on the latter, quantitative representation of geographic space that characterizes GIS. It is only in recent years that much attention has been paid to the development of computer systems to retrieve geographically-specific information from the relatively unstructured but immense resource of digital documents, many of which can be found on the worldwide web (e.g., [7,9,11]).

Because of the importance of geographical context, it is not surprising that requirements for information on the web often have a geographical focus (it has been estimated that between 13–15% of web queries submitted to traditional search engines contain a place name) [6,12]. It is this need for public access to geographical information on the web that has been a major motivation for the growth of the academic field of geographical information retrieval (GIR).

Much current research in GIR can be regarded as an extension of the field of information retrieval (IR). The products of work in IR have provided the foundation for current search engine technology, with its focus on access to primarily textual documents that are relevant to user queries taking the form of a phrase or set of key words. This latter type of query is in contrast to the more formal approach common in spatial databases, where georeferenced data objects are retrieved from structured tables in response to queries that can stipulate precise spatial constraints. IR methods usually identify documents that contain query terms and rank the retrieved documents using statistical methods that are intended to highlight the most relevant to the user's needs. For some forms of geographical search, particularly when looking for common resources within relatively large geographic extents (e.g., hotels in London), this approach can work well but it is fraught with limitations.

From the user's perspective these limitations can be manifested in a failure to distinguish between different instances of the same place name, a lack of ability to deal with spatial qualifiers such as near or north, a lack of methods to rank and explore results with respect to their spatial relevance and the non-retrieval of resources which are spatially relevant but use a place name different from that specified in the query. GIR is therefore concerned with improving the quality of geographically-specific information retrieval with a focus on access to unstructured documents such as those found on the web. There are several types of functionality that are distinctive to GIR and which constitute significant challenges. These include:

- Detecting geographical references in the form of place names and associated spatial natural language qualifiers within text documents and in user queries.
- Disambiguating place names to determine which particular instance of a name is intended.
- Geometric interpretation of the meaning of vague place names, such as the "Midlands," and of vague spatial language such as "near".
- Indexing documents with respect to their geographic context as well as their non-spatial thematic content.
- Ranking the relevance of documents with respect to geography as well as theme.
- Developing effective user interfaces that help users to formulate effective queries and identify relevant results.
- Developing methods to evaluate the success of GIR.

The following sections elaborate upon these issues.

Foundations

Detecting Geographic References

Place names (or toponyms) can be used to refer to places on Earth, but they also occur frequently within the names of organizations and as part of people's names. It is also the case that place names may be used *metonymically*, for example to refer to administrative entities as in "talks with Washington." The process of geo-parsing is concerned with analyzing text to identify the presence of place names and other spatial language and distinguish the genuine geographical occurrences of place name usage from those where they are being used to refer to some other entity. This process is often treated as an extension of Named Entity Recognition that is a standard part of linguistic analysis in natural language processing (NLP). Various techniques can be employed to identify a place name, including the use of rule-based approaches and machine learning. In order to recognize that a word is a potential place name, it is common practice to employ a list of known place names as is found in a gazetteer. Within gazetteers, each name is usually accompanied by at least one parent region, a classification for the place and a coordinate-based footprint providing a quantitative location.

Disambiguating Place Names

Once it has been established that a place name is being used in a geographic sense, the problem remains of uniquely determining the place to which the name refers. Note that the combination of detecting place names and disambiguating them is referred to as geoparsing. There are many names that are shared between different places. For example there are over 70 "Springfields" in the US. When humans read a document with a place name in it, they will tend to resolve ambiguity using knowledge gained from contextual clues within the document. Automatic resolution of geographic scope, and hence disambiguation of place names, attempts to mimic the methods humans use, for example by considering together all of the place names in the document. If a place name occurs in association with a set of other place names, several of which refer to neighboring places or are instances of places within the same parent region, then that provides evidence to distinguish which meaning is implied. Equally if the text mentions a parent or child region of an instance of a particular name, then that can help to determine the sense that is intended [14].

Vague Geographic Terminology

Many of the place names that users employ when searching on the web are of an informal or vernacular nature, often without precise boundaries. Examples include the Pennines and the Borders in the UK, the south of France, the Swiss Mittelland and the midwest in the US. Existing geographical search facilities make use of gazetteers that are based largely on the administrative names of places and which do not, in general, include any representations of vernacular names. There have been a few studies of methods of determining what may be a fuzzy extent for such places (e.g., [8]) but there is still much to be done. Recently, web mining methods have been used to determine associations between a vernacular place name and the names of other places within it or in the vicinity [4]. Spatial extent can then be modeled using, for example density surfaces, or Delaunay triangulation based methods [2].

The spatial language (such as near, close, between and north of) that accompanies place name terminology can be as vague as some of the place names. Being able to interpret such terms will help in analyzing the geographic context of documents and in interpreting user queries that employ vague spatial language. There have been previous studies of the meaning of natural language qualitative spatial relations, while [13] show how nearness can be interpreted quantitatively by analyzing the use of phrases such as "walking distance" in combination with knowledge of the coordinates of the places that are referred to.

Spatial and Textual Indexing

When web documents have been categorized according to their geographical context they must be indexed in a way that enables them to be found quickly in response to user queries. Techniques for indexing documents according to the words that they contain are well established. Typically an inverted file of documents is created in which each word is associated with a list of the documents that contain the word. This text index can be combined with a spatial index that records which documents relate to particular regions of space. Building a spatial index of documents can be done if each document has one or more document footprints that represent the regions of geographic space to which the document refers. Each document footprint may correspond to the spatial extent of a geographic reference that occurs in the document. If there are many such references an effort may be made to establish the main geographic foci of the document as represented by a smaller number of footprints [1,14]. These footprints can then be indexed in the same way that any other piece of geometry would be indexed in a conventional GIS. A challenge remains, however, to find efficient ways to combine text and spatial indexes (see [15] for a summary of some approaches to spatio-textual indexing).

Retrieval of relevant documents requires matching the query specification to the characteristics of the indexed documents. As mentioned above, in conventional web search engines this starts by finding those documents that contain the query terms, before ranking the resulting documents. For geographical search, there is a need to match the geographical component of the query with the geographical context of the documents as represented by the document footprints. GIR queries can be characterized as a triplet of <theme> <relationship> <location> composed of a topic of interest in combination with a place name qualified by a spatial preposition such as near, in, or north of. The combination of place name (after disambiguation) and spatial preposition can be used to generate a query footprint representing, for example, the interpretation of an expression such as "near Bristol." This query footprint can then be used to access the relevant part of the spatial index and hence find document footprints that intersect the query footprint. The retrieved documents will then be the members of this latter set of geographically relevant documents that also contain the thematic (text) query terms in the query.

Geographical Relevance Ranking

Having found a set of potentially relevant documents they should then be ranked by some measures of their estimated degree of relevance to the query. Relevance with respect to the thematic part of the query and the retrieved documents can be represented by a score that takes into account factors such as the frequency of occurrence of query terms within retrieved documents as a function of the overall frequency of query terms within the whole collection. The spatial score can be some measure of the geometric match between the query footprint and the document footprints. These two scores can then be combined to find an overall relevance. [16] has described some methods for doing this in which the text and spatial scores are normalized to values between 0 and 1 before calculating their distance from an ideal combined score in the two-dimensional space of text and spatial scores.

User Interfaces

Retrieval of documents using GIR also provides a variety of research challenges in the development of user interfaces. Query formulation generally requires, as indicated above, specification of a triplet of <theme> <spatial relationship> <location>. This can easily be facilitated by a simple structured interface, but this assumes sufficient geographic knowledge to specify relevant place names. Other approaches to query formulation allow users to sketch a region of interest on a map, and enter a related concept within a textbox [11]. This approach presupposes that users are interested in a spatial relationship representing containment, but to date very little progress has been made in developing map-based interfaces that allow users the option of graphically specifying other spatial relationships.

The results of a query to a GIR system can be treated in an identical manner to those of a traditional search engine, and simply displayed as a ranked list. In practice, the nature of geographic search and the pervasiveness of map-based web services means that the overlaying of results on a map has become a natural and expected visualization mechanism. Several challenges exist in displaying the results of GIR searches through a map interface. Many relevant documents may have the same geographic footprint, and techniques are required to allow the aggregation of these relevant documents, while summarizing or filtering duplicate content. Equally, document footprints often have scopes which are not sensibly represented at all scales as a point (e.g., London) and methods are required to allow users to explore documents with extensive geographic scopes in meaningful ways. Finally, there is much room for techniques from the Geovisualization community to be applied in exploring the typically large sets of documents that are returned by GIR.

User Studies and Evaluation

In developing GIR techniques, it is very important to take cognizance of user needs and to develop techniques to evaluate the quality of approaches to GIR. With respect to user needs, work is required to analyze where gaps exist in current approaches and to analyze query logs, firstly to identify the types of geographic search users undertake, and secondly to assess where users fail or have difficulty in search [6,12].

Evaluation is a key part of IR. In general, evaluation strategies within IR either take the form of system-focused or user-centered studies [5]. The former are based around the use of test collections, the measuring of the relevance of documents to specific queries and the calculation of standard IR measures such as precision and recall for a variety of systems and settings. Such methods require substantial resources to implement, particularly since relevance judgments must be performed manually and the numbers of judgments required are large. Within IR, the long-running TREC experiments have provided a mechanism for the pooling of resources to the mutual benefit of the research community. In GIR, GeoCLEF [3] has gone some way towards addressing this need by mounting a campaign to compare performance of a range of approaches to GIR across a set of multilingual queries performed on document collections based on newspaper articles. However, there is much room for further research on relevance judgment and measures for GIR. Current techniques are based on a single dimension of relevance using standard IR measures, which in general do not take account of the G (geography) in GIR [10]. To date, very little user-centered evaluation has been performed in GIR.

Key Applications

Geographical information retrieval is applicable in all areas where there is a need to specify geographical constraints when searching for information in document collections such as those found in digital libraries and on the World Wide Web. This applies in particular to spatially-aware (or geographical) search engines.

▶ *This Chapter derives from the following article which was originally published in the International Journal of Geographical Information Science vol 22, 2008, pp 219–228.*

▶ *The article "Geographical Information Retrieval" and journal International Journal of Geographical Information Science are copyright of Taylor & Francis.*

Cross-references

▶ Gazetteers
▶ Geographic Information System
▶ Georeferencing
▶ Human-Computer Interaction
▶ Information Retrieval
▶ Inverted Files
▶ Spatial Indexing Techniques

Recommended Reading

1. Amitay E., Har'el N., Sivan R., and Soffer A. Web-a-where: geotagging web content. In Proc. of the 27th Int. Conf. on Research and Development in Information Retrieval, 2004, pp. 273–280.
2. Arampatzis A., van Kreveld M., Reinbacher I., Jones C.B., Vaid S., Clough P., Joho H., and Sanderson M. Web-based delineation of imprecise regions. Comput. Environ. Urban Syst., 30:436–459, 2006.
3. Gey F.C., Larson R.R., Sanderson M., Bischoff K., Mandl T., Womser-Hacker C., Santos D., Rocha P., Di Nunzio G.M., and Ferro N. GeoCLEF 2006: the CLEF 2006 Cross-language geographic information retrieval track overview. In Proc. Workshop on the Cross Language Evaluation Forum, 2007, pp. 852–876.
4. Jones C.B., Purves R.S., Clough P.D., and Joho H. Modelling vague places with knowledge from the web. Int. J. Geogr. Inf. Sci., 22:1045–1065, 2008.
5. Jones K.S. and Willett P. (eds.). Readings in Information Retrieval. Morgan Kaufmann, San Francisco, CA, 1997.
6. Jones R., Zhang W.V., Rey B., Jhala P., and Stipp E. Geographic intention and modification in web search. Int. J. Geogr. Inf. Sci., 22(3):229–246, 2008.
7. Larson R. Geographic information retrieval and spatial browsing. In GIS and Libraries: Patrons, Maps and Spatial Information, L. Smith, M. Gluck (eds.). University of Illinois, Urbana-Champaign, 1996, pp. 81–124.
8. Montello D., Goodchild M., Gottsegen J., and Fohl P. Where's Downtown?: Behavioral methods for determining referents of vague spatial queries. Spatial Cogn. Comput., 3:185–204, 2003.
9. McCurley S.K. Geospatial mapping and navigation of the web. In Proc. 10th Int. World Wide Web Conference, 2001, pp. 221–229.
10. Purves R.S. and Clough P. Judging spatial relevance and document location for geographic information retrieval, extended abstract. In Proc. 4th Int. Conf. on Geographic Information Science, 2006, pp. 159–164.
11. Purves R.S., Clough P., Jones C.B., Arampatzis A., Bucher B., Finch D., Fu G., Joho H., Syed A.K., Vaid S., and Yang B. The design and implementation of SPIRIT: a spatially aware search engine for information retrieval on the internet. Int. J. Geogr. Inf. Sci., 21:717–745, 2007.
12. Sanderson M. and Kohler J. Analyzing geographic queries. In Proc. of the ACM SIGIR 2004 Workshop on Geographic Information Retrieval, 2004.
13. Schockaert S., De Cock M., and Kerre E.E. Location approximation for local search services using natural language hints. Int. J. Geogr. Inf. Sci., 22(3):315–336, 2008.
14. Silva M., Martins B., Chaves M., Cardoso N., and Afonso A.P. Adding geographic scopes to web resources. Comput. Environ. Urban Syst., 30:378–399, 2006.
15. Vaid S., Jones C.B., Joho H., and Sanderson M. Spatio-textual indexing for geographical search on the web. In Proc. 9th Int. Symp. Spatial and Temporal Databases. 2005, pp. 218–235.
16. van Kreveld M., Reinbacher I., Arampatzis A., and van Zwol R. Multi-dimensional scattered ranking methods for geographic information retrieval. Geoinformatica, 9:61–84.

Geographic Information Services

▶ Location-Based Services (LBS)

Geographic Information System

MICHAEL F. GOODCHILD
University of California-Santa Barbara, Santa Barbara, CA, USA

Synonyms

Geospatial information system; Spatial information system

Definition

A geographic information system (GIS) is a computer application designed to perform a wide range of operations on geographic information. Geographic information is defined as information about locations on or near the surface of the Earth, and may be organized in a variety of ways. Thus a GIS includes functions to input, store, visualize, export, and analyze such information. Commercial off-the-shelf GIS software is today capable of virtually any conceivable operation on geographic information, and capable of recognizing hundreds of different formats. GISs are used in a wide range of applications, from the management of the distributed assets of utility companies to emergency response. Their scientific applications are found in any discipline that deals with phenomena distributed over the surface of the Earth, from ecology to criminology. Increasingly GIS technology is encountered by ordinary citizens, in the form of map-making sites based on Google Maps, wayfinding sites such as MapQuest, and hotel-finding sites such as Expedia.

A wide range of introductory textbooks on GIS are available with various levels of sophistication, covering the fundamentals of representation, analysis, and application. Clarke [2] provides a general introduction; Longley et al. [4] give a somewhat more advanced and comprehensive perspective; and Worboys and Duckham [6] provide a computational viewpoint.

Historical Background

GIS has several roots, and today's technology represents a convergence among several independent developments. In the mid 1960s, the Government of Canada developed the Canada Geographic Information System (CGIS), the first to use the term GIS, as a means of generating products from the Canada Land Inventory. CGIS implemented a very small set of functions, including map overlay and area calculation, and in its initial form included no capabilities for map output, yet it solved many fundamental problems. Another set of developments centered around the desire to automate the process of making paper maps, and yet another was stimulated by the need to manage complex sets of data in support of transportation planning and census data collection. By the late 1970s, the commonalities in these various threads had been recognized, and the first commercial software products began to emerge. Foresman [3] provides a comprehensive history of GIS.

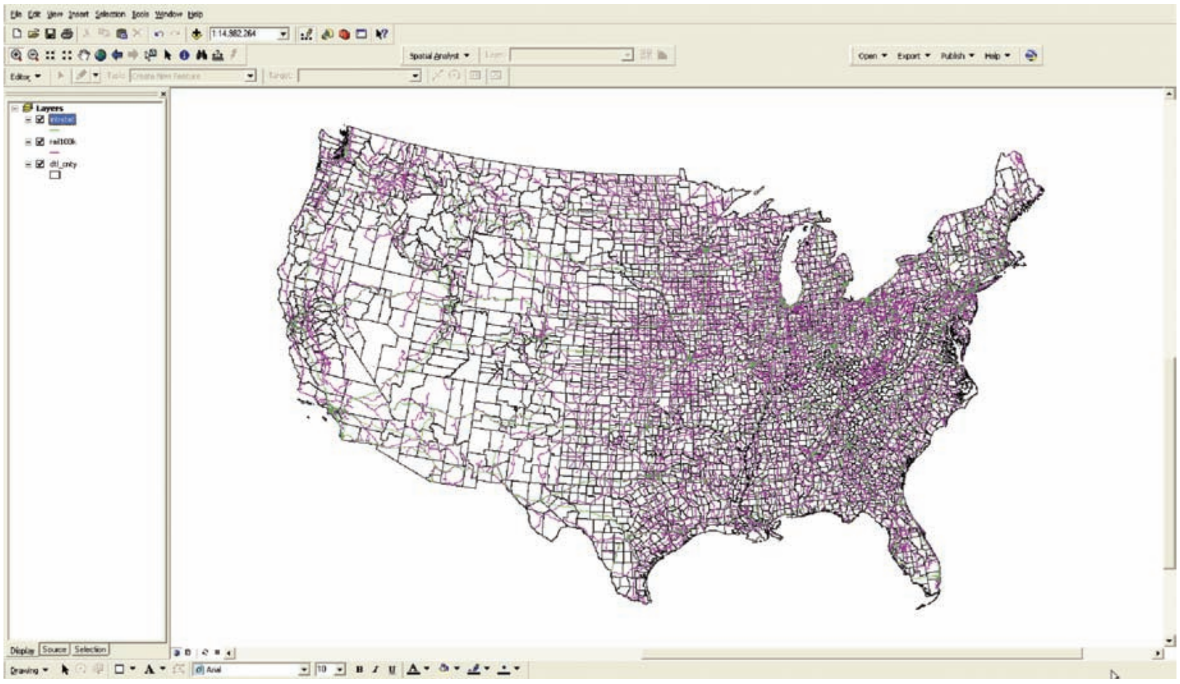
The early efforts to develop GIS were based on idiosyncratic data models and formats. By the late 1970s, the value of relational databases had been recognized, and GIS developers had begun to represent complex geographic features as interrelated collections of elements. A map of counties and their attributes, for example, could be represented as tables of nodes, edges, and faces; and the same approach could be used to represent a road network or a map of land cover types. Nevertheless, early relational database software was not easily adapted to the storage of the variable numbers of vertices needed to represent each edge, and so a hybrid approach had to be adopted in which some information was stored in a relational database and the remainder in a unique, proprietary structure. By the late 1980s, major problems of lack of interoperability began to impact the industry and led to the formation of the Open Geospatial Consortium.

By the 1990s, GIS designers had begun to adopt object-oriented database principles [1], and database technology and computing power had advanced to the point where hybrid architectures were no longer necessary. Greater flexibility in data modeling had opened the possibility of applying GIS tools to phenomena that were never traditionally associated with maps, such as events, transactions, movements and flows, and dynamic phenomena in general. GIS is no longer seen as a container of maps, but as an engine for the representation and analysis of spatio-temporal phenomena. The only limitation is that the information they contain be tied to specified locations on or near the surface of the Earth.

Figure 1 shows a screen shot of a typical application of a contemporary GIS, in this case ESRI's ArcMap.

Foundations

Two fundamentally distinct ways of conceptualizing spatial variation are recognized in GIS. In the discrete-object view, the Earth's surface is analogous to an empty table-top, on which are distributed a countable collection of features. The features may overlap, but between them is emptiness. The features might be represented as points, lines, or areas; and in some cases the third spatial dimension will be important and features may be represented as volumes. In the continuous-field view variation over the Earth's surface is described by a series of functions f of location $\mathbf{x}$, where f may be a measure, as in the case of elevation or temperature, or a class or name, as in the case of maps of soils or counties.



Geographic Information System. Figure 1. Screen shot of ESRI's ArcMap GIS. Three layers are displayed, as listed in the upper left: the county boundaries of the US (black), the railroads (blue), and the interstate highway network (green). Pull-down menus and toolbars provide access to a wide array of manipulation and analysis functions, and many more are available as third-party extensions. The display uses the Lambert Conformal Conic projection.

Location **x** may have two or three spatial dimensions, and may also include time in the case of dynamic phenomena. Phenomena distributed over networks, such as vehicle densities or pavement quality, can be conceptualized as fields distributed over a one-dimensional space that is in turn embedded in two- or three-dimensional space.

These conceptualizations can be implemented in GIS in two ways, as raster or vector structures. In a raster structure the set of possible locations is finite, being defined by a grid that is normally rectangular (though more complex grid geometries are needed when the Earth's curvature is important). In a vector structure, every feature is located using an appropriate number of coordinates. Areas and lines are normally represented as ordered sets of coordinates connected by straight lines, and known as polygons and polylines, respectively. These methods are straightforward in the case of discrete objects, but the representation of continuous fields requires another step of discretization, with six commonly used alternatives in the case of two-dimensional variation: (i) a set of irregularly spaced sample points, as used for example in capturing and representing weather observations; (ii) a set of regularly spaced sample points, as used in representing terrain; (iii) representation of the isolines of the field, as used in contour maps; (iv) a triangular mesh, with linear variation within the triangles and continuity of value across triangle edges; (v) a set of rectangular raster cells, as used in capturing and representing images; and (vi) a set of irregularly shaped, non-overlapping, and space-exhausting areas represented as polygons, as used in mapping land cover types or aggregated census data.

Whether raster or vector, any GIS data set must be referenced to the Earth's surface using some form of coordinate system. In the case of raster data this is usually achieved by registering the corner points of the raster; while in the case of vector data every feature is independently positioned in the coordinate system. Many alternative coordinate systems exist, including latitude and longitude plus numerous systems based on projections of the Earth's curved surface onto a plane. Some have been officially adopted by countries, such as the UK's National Grid, while others have been adopted by international agreement through agencies such as NATO, or by individual states in the case of the US State Plane Coordinate system. Unfortunately this leads to complexity and lack of interoperability, and any GIS must offer a comprehensive set of functions for managing coordinate systems. Moreover, the definitions of latitude and longitude are not entirely standard, being dependent on the mathematical figure adopted to represent the shape of the Earth, otherwise known as the datum, which adds another dimension of complexity. One approach to dealing with interoperability has been through the development of metadata standards, or standardized descriptions of data sets that include specification of such properties as coordinate systems. Metadata are now widely used by portals, libraries, and warehouses of geographic data that offer massive information resources to users over the Internet.

A GIS database is normally conceptualized as a collection of layers, each registered to the Earth's surface. A layer may contain the representation of a field, such as a map of land cover type, or it may contain a collection of discrete objects, such as buildings. This layer-based view of the world is often used as an icon, since it is so fundamental to GIS, and has appeared on the cover of more than one textbook. Raster-based layers will likely describe the geographic variation of only a single property, such as county name or land cover type, and because this property will exist for every cell and will have only a single recorded value per cell there is clearly a natural affinity between raster layers and continuous-field conceptualizations. Vector-based layers on the other hand may represent either collections of discrete objects with any number of associated attributes, or the variation of a single property conceptualized as a continuous field. Unfortunately today's GIS designs do not recognize the field/object distinction, with the result that no automatic procedures exist to avoid violating the constraints that apply in the field case. For example, a GIS will treat a set of digitized isolines representing a field as if they were discrete, independent polyline objects, and will not object if a user attempts to edit them in such a way as to make two isolines cross.

Many GIS applications employ some form of tiling, in which the geographic area covered by the database is divided into smaller units, often in the interests of computational efficiency. Tiles correspond to the map sheets of cartography, and are often defined on a rectangular basis. In some cases they may be evident to the user, who must develop explicit strategies for handling queries or analyses that involve multiple tiles; in other cases the tiling scheme may be transparent. In addition, a GIS will employ one or more systems of indexing, again in the interests of computational efficiency. Van Oosterom [5] provides a comprehensive survey of spatial database indexing options.

GIS implementations vary from the stand-alone desktop system designed to support the work of a single user, to departmental solutions that integrate the work of several people around a common database, to enterprise-wide solutions that organize the entire operation of a company or agency around its GIS. Increasingly, GIS functions are available on-line from servers, either based on data held at the server, or on data supplied by the user. For example, many Web sites now offer the GIS function of geocoding, which takes a list of postal addresses as input and returns their corresponding geographic coordinates.

A wide range of GIS software products are available to match these various implementation strategies, ranging from versions for hand-held devices through desktop systems to server-side GIS. The use of component-based software architectures has allowed vendors to build different products from the same collection of discrete elements. The trend in recent years has been away from the single-user desktop to integrated client-server architectures, and increasing reliance on data downloaded from Internet portals. Open-source GIS is growing steadily in popularity, and a number of low-cost options have recently appeared to challenge the marketplace dominance of the industry leaders, ESRI and Intergraph. Idrisi, originating from Clark University, continues to provide a software option with very strong support for decision making. Niche products also exist, particularly in transportation where the products of Caliper offer some powerful capabilities, and products from developing countries, particularly China, are increasingly competitive.

Key Applications

As noted above, the earliest applications of GIS were in isolated domains: land resource management; automated cartography; and transportation. By the late 1970s a broader vision of GIS had emerged as a general-purpose software application that could be used to solve a vast range of problems, all dealing in one way or another with the surface and near-surface of the Earth. Nevertheless, the first software products that emerged in the early 1980s found their most important applications in resource management, and were heavily adopted by forest-management companies and agencies, where they were used for compiling inventories, planning harvesting, and management of road access.

This viewpoint largely ignores the role of the military in GIS development, however. By the late 1950s the U.S. was heavily engaged in classified programs aimed at acquiring imagery from space, and in developing the systems needed to assemble, interpret, and analyze their products. Civilian satellites were first deployed in the early 1970s, and led quickly to a generation of GIS software, largely raster-based, that found useful applications in agriculture. Today the military and intelligence agencies are among the heaviest users of GIS, and many weapons systems and warfighting strategies depend on ready access to current, digital geographic information.

One of the most successful areas of GIS application has been in facilities management, and more generally in the management of distributed assets. Public and private utilities, planning departments of governments, transportation agencies, and managers of complex facilities such as university campuses will all use GIS routinely to inventory assets, manage and schedule maintenance, and address problems. Investments by individual companies may run to the millions of dollars, and involve hundreds of GIS workstations.

Of rapidly growing importance are GIS applications that impact the daily lives of the general public. These include route guidance, provided perhaps by an in-vehicle navigation system or by a Web service, and map-based services such as Zillow that provide current information on property values. The general public is increasingly familiar with GIS data sources, including remote sensing, and with concepts such as geo-tagging that permit information in sources such as Wikipedia to be georeferenced and thus linked to maps. The term neogeography is being used to describe such novel applications of GIS.

Future Directions

The days when GIS was an exotic computer application familiar only to a small elite, and accessible only after extensive training, are long gone; today virtually everyone with access to the Internet is familiar with at least some of its capabilities. The meaning of the term itself has become confused: some would reserve it exclusively for the single-user desktop application, while others would extend it to cover what is now a vast array of activities, for which the adjective geospatial has emerged recently as an alternative umbrella term. In discussing future directions, then, it is possible to take either the first, somewhat narrow perspective, or the second broader one.

From the narrow perspective, GIS software will continue to evolve, particularly in the direction of greater support for dynamics. Some application areas are currently in their infancy, and will likely grow in importance in the next few years. They include human health, the understanding of disease transmission and diffusion, and the assessment of health service outcomes; and business, whose mantra "location, location, location" feeds directly into GIS applications. Related areas of logistics, real estate, and insurance are also ripe for greater use of GIS.

A rich research agenda has evolved over the past decade, and will likely produce results that will in turn impact future generations of GIS software. The University Consortium for Geographic Information Science has taken the lead in this context, with short- and long-term agendas that address topics ranging from spatial cognition (How can GIS interfaces be made easier to use?) to the social impacts of GIS technology (Who gains, who loses?). The topic of uncertainty remains a thorny issue, with much useful research completed on the sources and management of uncertainty in GIS but little adoption in mainstream software. As a result, the user is left with little choice but to accept the results of GIS analysis at face value, while knowing that the data on which they were based is inevitably imperfect and uncertain.

From the broader perspective, GIS functions will continue to become more available in Web services. In the past year a very interesting series of developments have occurred that are enabling citizens to create their own geographic information, and to integrate and publish it on the Internet. Sites such as OpenStreetMap are enabling communities to create their own maps; Wikimapia is collecting descriptions of features on the Earth's surface from vast numbers of volunteers; and Flickr is assembling a vast collection of geo-referenced photographs. This process of volunteering geographic information is having a profound impact on citizens' knowledge of the planet, and on their engagement with geography and GIS.

Cross-references

▶ Data Models
▶ Field-Based Spatial Modeling
▶ Spatial Indexing Techniques
▶ Spatial Network Databases
▶ Spatial Data Analysis

Recommended Reading

1. Arctur D. and Zeiler M. Designing Geodatabases: Case Studies in GIS Data Modeling. ESRI Press, Redlands, CA, 2004.
2. Clarke K.C. Getting Started with Geographic Information Systems. Prentice Hall, Upper Saddle River, NJ, 2003.
3. Foresman T.W., ed. The History of Geographic Information Systems: Perspectives from the Pioneers. Prentice Hall, Upper Saddle River, NJ, 1998.
4. Longley P.A., Goodchild M.F., Maguire D.J., and Rhind D.W. Geographic Information Systems and Science. Wiley, Chichester, UK, 2005.
5. van Oosterom P. Spatial access methods. In Geographical Information Systems, vol. 1, P.A. Longley, M.F. Goodchild, D.J. Maguire, and D.W. Rhind (eds.). Wiley, New York, NY, 1999, pp. 385–400.
6. Worboys M.F. and Duckham M. GIS: A Computing Perspective. CRC Press, Boca Raton, FL, 2004.

Geographic Information

▶ Geography Markup Language

Geographic Web Search

▶ Geo-Targeted Web Search

Geographical Analysis

▶ Spatial Data Analysis

Geographical Data Analysis

▶ Spatial Data Analysis

Geographical Databases

▶ Semantic Modeling for Geographic Information Systems

Geographical Metadata

▶ Geospatial Meta Data

Geography Markup Language

JAYANT SHARMA, JOHN HERRING
Oracle Corporation, Nashua, NH, USA

Synonyms

ISO 19136; Geographic information; Geography markup language

Definition

Geography Markup Language (GML) (http://www.opengis.net/gml/) is an XML vocabulary that can be used as the basis for the modeling, transfer, communication, or storage of geographic feature information. It is defined and maintained by members of the Open Geospatial Consortium (OGC), a voluntary consensus standards organization that develops standards for geospatial information handling and location based services. It has been accepted as one of and harmonized with the standards of ISO TC 211: Geographic information (http://www.isotc211.org/). The official copies of all versions of the GML specification and related OGC standards can be found at http://www.opengeospatial.org/standards/. The most recent version of GML, version 3.2.1, is identical with ISO 19136 (http://www.isotc211.org/Outreach/Overview/Factsheet_19136.pdf).

The primary purpose and rationale for GML, as described by the OGC, is to provide an open, vendor-neutral format for the definition of geographical entities used by applications, in various domains, for geospatial information processing and exchange. The enhanced ability of organizations to, efficiently and effectively, re-use, share, integrate, and consequently extract more value from, their extremely valuable geospatial information across operational units and processes is one immediate benefit of using GML.

Historical Background

GML (version 1.0) was first submitted to the OpenGIS Consortium, now Open Geospatial Consortium in 1999, as a "request for comment" (RFC) by authors from CubeWerx, Compusult, Galdos Systems, MapInfo, and Oracle. It was accepted by OGC and published in May 2000. GML 2, the first widely accepted version, was published in February 2001. GML 2.1 is still commonly used today.

GML 3.1 was published by OGC in February 2004, and submitted to ISO TC 211: Geographic Information, for consideration as an ISO standard. This would make it acceptable under the laws of some countries as the basis for national standards. It has since been accepted and ISO 19136 (GML, version 3.2.1), became an International Standard in 2007.

Foundations

GML is based on a feature and property model. Any complete GML file contains a collection of features, and each feature contains a collection of properties, and each property contains its value, which may either be direct (lexically contained within the property as an XML sub element) or indirect (pointed to by the property using an xlink).

Features represent real-world entities, such as roads or parks. Some real-world entities are composed of other entities and hence are represented as collections of features. For example, a city may be represented as a collection of roads, parks, land parcels etc.

Features have properties that have a name and type. For example, a road has a name (i.e., a property called "name") and its value is of type "String." Features may also have named geometry-valued properties. For example, a road might have a geometry property named "centerLineOf" that is of type "Linestring."

GML implements a subset of the ISO 19107 standard geometry model that defines the primitive geometry types and their hierarchy. GML 2 was concerned with simple features that only have a simple two-dimensional geometry property. GML 3 addresses requirements of applications that need to model more complex geographic phenomena that are dynamic (e.g., coverages or sensors), and have non-linear (e.g., circular or parametric arcs) or 3-D (cubes, prisms) geometry, 2-D topology, or temporal properties. To be useful, such feature models must include spatial and temporal reference systems, units of measure, metadata, and default styling (or portrayal) information.

Base and Application Schema

In order to enable an open, interoperable, vendor-neutral means of modeling and exchanging geographic information, GML includes types for features, geometry, topology, coordinate reference systems, grids, measures, observations, and temporal reference system/values.

Feature types form the basic logical construct and all other types are used to express property values for features. Geometry types are used to describe the basic spatial attributes of a feature. Topology types can be used to describe the spatial relations between those geometries. Coordinate and coordinate reference systems describe the manner in which point locations are described for geometry (and related purposes). Grids are commonly used for "image-like" raster structures for distributing measurements across space. Measure and observation types are used for describing measurable attributes and their units. Temporal values and their reference systems are used for expression of time.

Applications define and describe a "feature schema" as defined in ISO 19109: Rules for application schema. These schemata control the types of features that can actually be stored in a GML file. Each schema uses the basic GML structure as defined above and created "feature types" which determine the name of the feature types and which properties a particular feature type must (mandatory) or may (optional) contain, and with what cardinality (maximum and minimum occurrence). Because these application schemata are restricted by ISO 19109 and by the syntax of XML schema, they are single classification objects as is common in most programming languages.

Profiles

GML is quite comprehensive and consequently complex, even its early versions. To compensate for this for particular information communities (informal groups of users with similar data content needs), "profiles" are often defined as subset of the full base schema for use within their applications.

One profile is the "point profile" which simply describes the way geographic point location can be expressed. It is used as the basis for geographic extension of other standards that only require a geographic point as a location reference and hence do not use, or require, the GML feature and geometry model.

The most common profile is the one used for another OGC standard, Simple Feature Access that includes a data architecture volume (http://www.opengeospatial.org/standards/sfa). It was defined first in 1999, and became an ISO standard soon thereafter. The GML simple features profile was developed to support another OGC specification, namely, the OpenGIS Web Feature Service (WFS) Implementation Specification. The primary purpose is to define a useful subset of GML that (i) "lowered the implementation bar," and (ii) supported the Simple Features for SQL (SF-SQL) geometry model and hence the WFS specification. So this profile "prescribes the encoding of GML application schemas in sufficient detail that client applications do not need to deal with the entire scope of XML-Schema and GML but only need to understand a restricted subset of both specifications in order to be able to interpret schema documents generated or referenced by servers offering data encoded in GML." As a consequence, this document is just 49 pages long compared to over 800-page length of the full GML specification.

Other relevant profiles are the common CRS, and common dictionary profiles. These define encodings of commonly used coordinate reference systems and conversions, and simple dictionaries respectively.

Key Applications

The WFS (Web Feature Service) implementation specification defines an interface for retrieving or updating, geographical features encoded in GML, using HTTP.

RSS (real simple syndication, http://www.rssboard.org/rss-specification) is a news feed mechanism for the web and GeoRSS (http://www.rssboard.org/rss-specification) uses the geometry of the simple feature profile to geographically locate the information in the news. See http://www.georss.org/gml.html for some examples of GML usage in GeoRSS.

JPEG2000 (http://www.jpeg.org/jpeg2000/) is a digital image format that allows "XML" data to be stored with the image data. GML is the preferred format for geographic information and the combination of image and geographic information allows JPEG2000 to be used for geographically referencible images such as satellite and aerial imagery.

CityGML (http://www.citygml.org/) is an application schema for GML that integrates building information, at various levels of detail, into 3-D city and landscape models. The basic concept is to create a full 3-D model capable of supporting a wide variety of applications associated to such fields as environment, telecommunications, security, navigation and others requiring highly accurate and detailed building and landscape models. Some examples of GML as used in CityGML are at http://www.citygmlwiki.org/index.php/Examples_and_WFSs. A free viewer for CityGML can be found at http://www.3dgeo.de/citygml.aspx.

The IETF (Internet Engineering Task Force – http://www.ietf.org/) has several "rfc" specifications that use GML for location information. In particular, *IETF RFC 4119 – A Presence-based GEOPRIV Location Object Format and IETF RFC 3863 – Presence Information Data Format (PIDF)*. Each uses a profile of GML for location information.

Schemas for GML

All versions of GML have their schemas posted for public use in some subdirectory of http://schemas.opengis.net/gml/. For example http://schemas.opengis.net/gml/3.2.1/contains the schema for the latest version of GML version 3.2.1 [5], which is the same as ISO 19136.

Examples

Some fairly simple examples of GML can be found in the Wikipedia article located at http://en.wikipedia.org/wiki/Geography_Markup_Language.

Cross-references

► Abstraction
► Data Models with Nested Collections and Classes
► Document Databases
► Geographical Information Retrieval
► Geographic Information System
► Georeferencing
► Geospatial Metadata
► Object Data Models
► Resource Description Framework (RDF) Schema (RDFS)
► Spatial and Spatio-Temporal Data Models and Languages

▶ Spatial Data Types
▶ Spatio-Temporal Data Warehouses
▶ Unified Modeling Language
▶ W3C
▶ XML
▶ XML Metadata Interchange
▶ XML Parsing, SAX/DOM
▶ XML Programming
▶ XML Schema
▶ Xpath/Xquery
▶ XSL/XSLT

Recommended Reading

1. Alberto B., Negri M., and Pelagatti G. An ISO TC 211 conformant approach to model spatial integrity constraints in the conceptual design of geographical databases. In Advances in Conceptual Modeling – Theory and Practice, 2006, pp. 100–109.
2. Bacharach S. The GML simple features profile and you, directions magazine.
3. Lake R., Burggraf D., Trininic M., and Rae L. Geography Markup Language: Foundation for the Geoweb, Wiley, New York, 2004.
4. Lu C., Santos R.F. Jr., Sripada L.N., and Kou Y. Advances in GML for geospatial applications. GeoInformatica., 11(1):131–157, 2007.
5. Portele C. (ed.). OpenGIS Geography Markup Language (GML) Encoding Specification 3.2.1, OGC Document #07–036.
6. Vretanos P. A. (ed.). OpenGIS Simple Features Profile (1.0.0), OGC Document #06–049r1.
7. Warnill C. and Bae H. A specification of a moving object query language over GML for location-based services. In Proc. 6th Asia-Pacific Web Conference on Advanced Web Technologies and Applications, 2004, pp. 788–793.
8. Zhong-Ren P. and Zhang C. The roles of geography markup language (GML), scalable vector graphics (SVG), and web feature service (WFS) specifications in the development of internet geographic information systems (GIS). J. Geogr. Syst., 6(2): 95–116, 2004.

Geometric Data Types

▶ Spatial Data Types

Geometric Mean Average Precision

▶ GMAP

Geometric Stream Mining

Cecilia M. Procopiuc
AT&T Labs, Florham Park, NJ, USA

Definition

Let $P = \{p_1, p_2, \ldots\}$ be a stream of points in the metric space (X, L_q). Usually, $X = \mathbb{R}^d$ or $X = \{1, \ldots, U\}^d$ (discrete case), and $L_q = L_2$ is the Euclidean distance. The set P is called a *spatial data stream*. Geometric stream mining algorithms compute the (approximate) answer to a geometric question over the subset of P seen so far. For example, the *diameter* problem asks to maintain the pair of points that are farthest away in the current stream. A more comprehensive list of problems is presented later.

Historical Background

Geometric algorithms in the offline setting have been extensively studied over the past decades. Their applications encompass many fields, such as image processing, robotics, data mining, or VLSI design. For an introduction to computational geometry, refer to the book [8]. On the other hand, research on spatial data streams is a recent development. Shortly after the first results on numeric data streams appeared, a slew of papers argued that in many applications numeric streams exist in metric spaces, and that the capabilities of such systems can be expanded by taking into account the underlying metric, and by supporting geometric queries. Queries can be similar to offline databases (e.g., range queries), but they can also be about the shape and extent of the stream (e.g., diameter). Most examples of spatial streams come from sensor networks, traffic monitoring, and satellite data, all of which have location information.

Foundations

Stream Models

Online algorithms come in two flavors. In the first model, points can only be inserted in the stream. In the second, both insertions and deletions are permitted. The latter case is usually referred to as the *turnstile* or *dynamic* model. Allowing deletions introduces distinct challenges. Frahling et al. [5] proved that certain computations over a dynamic stream from $\{1, \ldots, U\}^d$ require $\Omega(U^d)$ space in any deterministic approach, but only $O(d \log U)$ space if randomization is allowed. This contrasts with the insertions-only case, for which there are deterministic space-efficient methods.

Further classification of stream models depends on whether the metric space X is continuous or discrete. In the following, *data stream* refers to insertions-only streams from a continuous metric space, unless stated otherwise.

Classes of Problems

Some geometric stream problems are closely related, i.e., they can be solved by the same methods. Such a classification is presented below. The details of the techniques are deferred to the next subsection.

Range Counting and Robust Statistics *Range counting* asks for the number of points that lie inside an arbitrary axis-parallel hyper-rectangle. The query hyper-rectangle is not known a priori. The *Tukey depth* of a point p is the minimum proportion of points among all the half-spaces that contain p (the larger the depth, the more "central" the point). The *simplicial depth* of p is the proportion of simplices that contain p, among all simplices that are convex hulls of $d + 1$ data points. These problems can be approximated using a so-called *ε-approximation* of the data.

Extent Measures The *diameter* of a point set is the maximum distance between two points in the set. The convex hull is the smallest convex region that contains the set. The *width* is the minimum distance between two parallel hyper-planes so that the set lies between them. The *minimum enclosing cylinder, respectively sphere* is the cylinder, respectively sphere, of minimum radius containing the points. The *minimum cylindrical, respectively spherical, shell* is the region of minimum width between two concentric cylinders, respectively spheres, that contains the points. These problems can be approximated by computing appropriate *ε-kernels*. More efficient methods exist for the diameter and convex hull, but they do not extend to the other problems.

Geometric Graphs Many graph problems have a natural equivalent when the graph is embedded in a metric space, and the distance between points is given by that metric. Such problems include the *minimum weight spanning tree*, *minimum match*, and *TSP*. In the online version, the solution cannot be stored explicitly, so only the cost is computed. A general technique is to reduce the problems to vector computations over streams, and use existing algorithms from that area. Another approach is to maintain appropriate ε-coresets.

Nearest Neighbor and Skyline The classical nearest neighbor problem asks for the data point closest to a query point. This cannot be computed in sub-linear space, as any data point can be the answer to some query. Thus, several modifications have been proposed for streams. Korn et al. [7] studied *reverse nearest neighbors*, in which the data set is known a priori, and the stream consists of query points. However, some data points may be inactive at times, and the query is only with respect to active points. An indexing structure is pre-computed on the data points (in one dimension, this involves sorting the points and putting a geometric grid between each pair of consecutive points). A different framework is studied by Böhm et al. Queries and points have time spans in which they are active, and the k-nearest neighbors are computed from among the active data points during the query's entire span. This can be reduced to the *skyline* problem, defined as follows. The skyline of a dataset with respect to a fixed quadrant contains those points p_i for which the quadrant translated with the origin in p_i has no other points. For example, for the lower right quadrant, p_i is in the skyline if all points whose x-coordinate is larger than p_i also have larger y-coordinate. For nearest neighbors, the data stream is transformed into new coordinates, with the x-axis being expiration time, and the y-axis being the distance to the query. Then the nearest neighbors are the points on the skyline. The skyline can be maintained exactly, but may have $\Omega(n)$ complexity. The authors propose heuristics for maintaining approximate skylines in that case.

General Techniques

Most geometric problems cannot be solved exactly in sublinear space, so approximate solutions are sought. Three techniques for designing approximation algorithms for spatial data streams are presented below.

1. Merge and Reduce A popular approach in geometric algorithms is to compute a "sketch" of the dataset, so that the optimal solution can be approximated by a computation over the sketch. The sketch is a small subset of the input, whose size usually depends only on the approximation error. Given its space efficiency, this approach is a good candidate for stream algorithms. To maintain a sketch online, the stream is divided into substreams, and a sketch is independently computed for each substream. As more points arrive, the algorithm merges older sketches and replaces them by a smaller sketch.

Figure 1 illustrates this process when the stream is divided based on the *logarithmic method*: Let n denote the number of points seen so far. Let $0 \leq i_1 < i_2 < ...i_k \leq \log n$ be the 1-bits in the binary representation of n, i.e., $n = 2^{i_k} + ... + 2^{i_2} + 2^{i_1}$. Divide the stream into disjoint substreams of consecutive points $P_{i_k},..., P_{i_1}$, such that P_{i_j} arrives before $P_{i_{j-1}}$ for all $2 \leq j \leq k$, and $|P_{i_j}| = 2^{i_j}$. When a new point arrives, assign it to P_0. If $|P_0| = 1$, i.e., P_0 was empty before, return. Otherwise, $|P_0| = 2$ violates the cardinality condition. Merge P_0 into P_1, and set $P_0 = \emptyset$. Continue the process until all sets have the correct cardinality (if necessary, create a new set $P_{i_{k+1}}$).

Since P_{i_j} cannot be stored exactly, its sketch S_{i_j} is stored instead. Merging two substreams is simulated by merging their sketches – this is the *merge step*. \To maintain the space bound, a sketch of the merged sketches is computed – the *reduce step*. Finally, a sketch S of the whole stream is computed as a sketch of all S_{i_j}, and the original optimization problem is solved over S. To guarantee correctness, sketches must observe two key properties:

1. If S_1, S_2 are sketches of P_1, P_2, then $S_1 \cup S_2$ is a sketch for $P_1 \cup P_2$ (correctness of merge step).
2. If S is a sketch of P and T is a sketch of S, then T is also a sketch of P, possibly with larger error (correctness of reduce step).

Let σ be the size of one sketch; σ is independent of n, but depends on ε. Then the overall space is $O(\sigma \log n)$. However, errors usually increase during the reduce step. To guarantee that the final sketch has error at most ε, a smaller error ε_i is used for each S_i. This increases the space by logarithmic factors. A recent result replaces the logarithmic method by a geometric strategy, eliminating all log n factors.

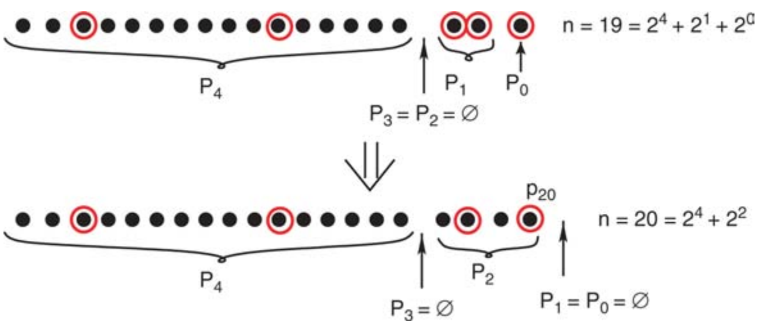
Two types of sketches apply to a wide range of problems. The first, called ε-approximations, was defined by Vapnik and Chervonenkis, in the context of set systems with finite VC dimension. For an introduction to this theory see, e.g., the book. Some classes of set systems to which the theory applies are specified below. The second type of sketches, called ε-coresets, were first introduced by Agarwal et al. and have emerged as a powerful tool in geometric approximation [1].

ε-approximations: A subset $S \subseteq P$ is an ε-approximation of P if for any *range* R from a fixed family of ranges of bounded VC-dimension

$$\frac{|R \cap S|}{|S|} = \frac{|R \cap P|}{|P|} \pm \varepsilon.$$

Examples of families of ranges of bounded VC-dimension include the set of all hyper-rectangles, and the set of half-spaces. Thus, a range counting query can be answered as follows. Let Q be the query hyper-rectangle and S be the current ε-approximation of the stream P. Then the number of points inside Q is approximated by $|Q \cap S| \cdot \frac{|P|}{|S|}$, and the error is $\pm\, \varepsilon|P|$.

Any set system of bounded VC-dimension admits an ε-approximation of size $O(\varepsilon^{-2}\log(\varepsilon^{-1}))$. Matousek designed a deterministic offline algorithm for computing such an ε-approximation. The algorithm was extended to the online setting by Bagchi et al. [5] using the logarithmic method as above.



Geometric Stream Mining. Figure 1. Merge and reduce via logarithmic method: A sketch of size at most two (*circled points*) is stored for each P_i. When p_{20} arrives, the sketch of P_2 is computed from p_{20} and the previous sketches of P_0,P_1.

ε-coresets and ε-kernels: Let $\mu : 2^X \rightarrow \mathbb{R}^+ \cup \{0\}$ be the measure to be optimized. For example, $\mu(P)$ can be the width, or the cost of the minimum weight spanning tree of P. Assume that μ is monotone, i.e., for any $P_1 \subseteq P_2$, $\mu(P_1) \leq \mu(P_2)$. A subset $S \subseteq P$ is an ε-coreset of P if

$$(1 - \varepsilon)\mu(P) \leq \mu(S).$$

Many geometric problems have ε-coresets of small size, i.e., $|S| = 1/\varepsilon^{O(1)}$.

The related notion of ε-kernels provides a blueprint for computing coresets for several measures. For any unit vector $v \in \mathbb{R}^d$, let $\mu_v(P)$ denote the *directional width* of P along v. Formally, $\mu_v(P) = max_{p \in P}\langle p, u\rangle - min_{p \in P}\langle p, u\rangle$, where $\langle \cdot,\cdot \rangle$ is the inner product. See Fig. 2a. A subset $S \subseteq P$ is an ε-kernel of P if it simultaneously ε-approximates all directional widths, i.e., for all unit vectors v,

$$(1 - \varepsilon)\mu_v(P) \leq \mu_v(S).$$

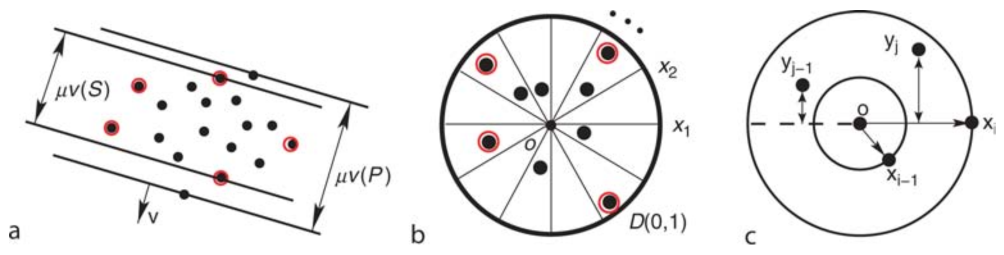
Figure 2b illustrates the offline algorithm by Agarwal et al. [1] for computing an ε-kernel in two dimensions. The algorithm assumes that all points of P lie inside the unit disk centered at the origin $\mathcal{D}(o, 1)$. It also assumes that P is "fat," i.e., there exists a constant $0 < \alpha < 1$ such that $\mu_v(P) \geq \alpha diam\ (P)$ for any direction v. Both conditions are enforced by applying an affine transform. The algorithm pre-defines a set of directions $\overrightarrow{ox_1}, \ldots, \overrightarrow{ox_r}$ and computes the extremal points in each direction. Let $S \subseteq P$ be the set of at most $2r$ extremal points. The authors show that one can choose $r = O(1/\sqrt{\varepsilon})$ directions, so that S is an ε-kernel. For the d-dimensional case, $r = O(\varepsilon^{-(d-1)/2})$.

Maintaining extremal points along a fixed set of directions extends to the online setting. However, computing the affine transform that insures that P is fat requires first scanning the whole set. Thus, the logarithmic method is used for the online version. Different affine transforms are applied to different substreams.

A significant improvement is due to Chan [3], who eliminates the log$n$ factors from the space bound. He partitions the data stream based on significant changes in the "shape" of the current substream, rather than its number of points. As before, a different affine transform applies to each substream, but the transform can be computed from the first three substream points.

In two dimensions, he uses the following affine transform. For any set P, let $o \in P$ be a fixed point, and let $x, y \in P$ such that $||op|| \leq 2||ox||$ and $d(p, ox) \leq 2d(y, ox)$ for all $p \in P$. Translate P to make o the origin. Rotate and scale P so that $x = (1/2,0)$. Map each $(a,b) \in P$ to $(a, b/2d(y, ox))$. Let τ be the resulting transform. Then $\tau(P)$ is fat and $\tau(P) \subseteq \mathcal{D}(o, 1)$.

Let o be the first point in the stream; see Fig. 2c. Then o is assigned to all substreams (which are no longer disjoint). Divide the stream into sequences called *epochs*. An epoch E_i starts with a stream point x_i, and contains any subsequent point p so that $||op|| \leq 2||ox_i||$. Once a point q arrives so that $||oq|| > 2||ox_i||$, epoch E_{i+1} starts with $x_{i+1} = q$. Each epoch E_i is further divided into *subepochs* as follows. The jth subepoch $E_{i,j}$ which started with point $y_j \in E_i$ contains all subsequent points $p \in E_i$ for which $d(p, ox_i) \leq 2d(y_j, ox_i)$. When a point q arrives so that $d(q, ox_i) > 2d(y_j, ox_i)$, and q does not start a new epoch, then subepoch $E_{i,j+1}$ starts with $y_{j+1} = q$. A substream is a subepoch $E_{i,j}$, plus o and the point x_i that starts epoch E_i. An affine transform as above is computed at the beginning of each $E_{i,j}$, using points o, x_i and y_j.



Geometric Stream Mining. Figure 2. Computing an ε-kernel: (a) an ε-kernel S (circled points) approximates the directional width for all directions υ; (b) offline algorithm: kernel = extreme points along fixed directions (circled points); (c) online algorithm: x_{i-1}, x_i start epochs $i - 1$, resp. i, where $d(o, x_i) > 2d(o, x_{i-1})$; y_{j-1}, y_j start subepochs $j - 1$, resp. j, inside epoch i, where $d(y_j, ox_i) > 2d(y_{j-1}, ox_i)$.

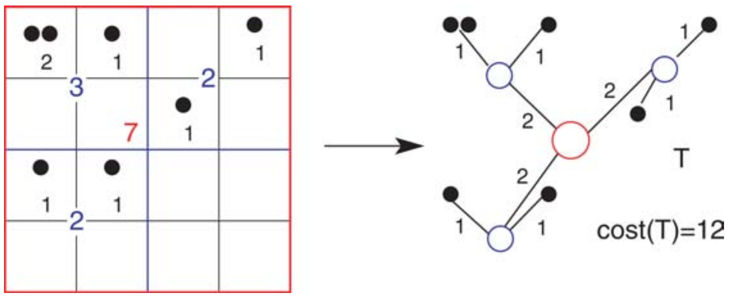
The number of subepochs can be $\Omega(n)$. However, Chan proves that it is sufficient to maintain kernels only for the most recent $\log(1/\varepsilon)$ epochs. Older epochs are so close to o that they can be represented by only two points. A similar argument applies to sub-epochs. The overall space is $O((1/\sqrt{\varepsilon})\log^2(1/\varepsilon))$, which was recently improved to $O(1/\sqrt{\varepsilon})$. In higher dimensions, a slightly different algorithm requires $O([(1/\varepsilon)\log(1/\varepsilon)]^{d-1})$ space.

2. Reduction to Vector Problems This technique has been applied to dynamic spatial streams, as the previous approach cannot handle deletions. The method was introduced by Indyk [6] and was used to approximate several geometric graph problems. It assumes a discrete space $\{1,\ldots,U\}^d$ and imposes a hierarchical decomposition on it. Statistics on the number of points in each cell is maintained under both insertion and deletion, using previous results from numerical data streams.

Figure 3 illustrates the method in two dimensions. The space decomposition consists of $k = \log U$ nested square grids $G_0,\ldots,G_k$ with side lengths $2^0,\ldots,2^k$, randomly shifted along each axis. This decomposition induces a tree structure, where a cell of G_i has at most four children corresponding to the non-empty cells of G_{i-1} contained in it. Data points are stored in leaves. The edge between a cell of G_i and its child from G_{i-1} has weight 2^{i-1}. Conceptually, the graph problem is solved on this tree, where the distance function is the shortest path. Indyk proves that, for randomly shifted grids, the expected cost of the tree solution is within $O(\log U)$ from the cost of the optimal solution. An $O(1)$ approximation of the tree solution can be computed by only maintaining statistics on the number of points in each non-empty cell. For the minimum spanning tree problem, the optimal solution on the induced tree is the tree itself, whose weight is $\sum_{i=0}^{k-1} 2^i n_i$, where n_i is the number of non-empty cells of G_i. Thus, for each i, it is sufficient to estimate the 0-norm of the vector defined over the cells of G_i under increment and decrement operations (i.e., when $p \in P$ is inserted, the counter of the cell containing p in G_i is increased by 1, and when p is deleted, the counter is decreased by 1). Thus, the original problem is reduced to a vector problem over data streams, which has already been studied. Other geometric graph problems can similarly be reduced to estimating vector norms of streams.

3. Random Sampling Several offline algorithms have a single-pass flavor and could work online, except that they need to select random samples from the data. If one can maintain random samples over a stream, then the algorithms extend online. When the stream is insertions-only, a random sample can be maintained via the reservoir sampling technique by Vitter [9]: Select the ith stream point p_i with probability r/i, where r is the size of the sample. If p_i is selected, it replaces a previous sample point, chosen uniformly at random.

For dynamic discrete streams, Cormode et al. [4] proposed a more general technique (a similar approach was independently designed by Frahling et al. [5]). The stream points are hashed to levels, such that the likelihood of being hashed to level l decreases exponentially in l. More exactly, if h is a universal hash function and $x \in \{1,\ldots,U\}$ is the stream point, then x is assigned to the level l for which $2C^{l+1}U \leq h(x) < 2C^{l}U$ ($0 < C < 1$ is a constant). Hence, there are $O(\log U)$ levels. Each level l has two counters $sum[l]$ and $count[l]$, initialized to zero, as well as a collision detection structure. If x is assigned to level l, then



Geometric Stream Mining. Figure 3. Cell count statistics (e.g., how many are non-zero) lead to approximate solution. Points can be inserted and deleted multiple times.

for each insertion of x, $sum[l] \leftarrow sum[l] + x$ and $count[l] \leftarrow count[l] + 1$; and for each deletion of x, $sum[l] \leftarrow sum[l] - x$ and $count[l] \leftarrow count[l] - 1$. If $count[l] > 0$ and there is only one point assigned to level l (as determined by the collision detection structure), then $sum[l]/count[l]$ is chosen in the random sample (note that $sum[l]/count[l] = x$, where x is the point assigned to l). The authors prove that there is a constant probability of finding such a level l, and that each point present in the stream has equal probability of being assigned to l. (A point is present in the stream if it has been inserted more times than it has been deleted.) To choose a random sample of size r, $O(r)$ independent copies of this structure are maintained.

Since a random sample of size $O(1/\varepsilon^2 \log(1/\varepsilon))$ is an ε-approximation with high probability, this provides a different method for maintaining ε-approximations. Frahling et al. [5] used the technique for the minimum spanning tree problem, improving the $O(\log U)$-approximation to $(1 + \varepsilon)$-approximation.

Key Applications

Most streaming applications whose data are embedded in a metric space require some analysis of the "geometry" of the stream. Three main areas, to which the above algorithms apply, are mentioned below.

Sensor networks Basic analysis of such systems computes information about the shape and extent of some sensor-covered area. For example, a system of sensors that monitor air quality can track the spread of a chemical spill, by maintaining the convex hull of the sensors that signal. In other applications, such as herd monitoring, the sensors are mobile, and the application tracks the distribution and density of the herd. In this case, it is useful to answer range counting and statistics queries.

Fixed wireless telephony This is an application of reverse nearest neighbor searching. Fixed wireless base stations are installed so that residential customers connect to the closest one that can handle the traffic. To optimize the performance, stations may be turned on or off depending on the stream of calls. To insure service quality, the system monitors station loads, as well as the maximum distance between a customer and the nearest station.

Mobile networks Such networks have a dynamic topology, in which connections between nodes appear and disappear based on their relative location. In this case, maintaining information on the network topology requires solving geometric graph problems.

Cross-references

- ▶ Approximate Query Processing
- ▶ Clustering on Streams
- ▶ Clustering Overview and Applications
- ▶ Sensor Networks
- ▶ Spatial Data Mining
- ▶ Stream Mining
- ▶ Stream Models
- ▶ Stream Sampling

Recommended Reading

1. Agarwal P.K., Har-Peled S., and Varadarajan K.R. Approximating extent measures of points. J. ACM, 51(4):606–633, 2004.
2. Bagchi A., Chaudhary A., Eppstein D., and Goodrich M.T. Deterministic sampling and range counting in geometric data streams. In Proc. 20th Annual Symposium on Computational Geometry, 2004, pp. 144–151.
3. Chan T.M. Faster core-set constructions and data-stream algorithms in fixed dimensions. Comput. Geom., 35(1–2):20–35, 2006.
4. Cormode G., Muthukrishnan S., and Rozenbaum I. Summarizing and mining inverse distributions on data streams via dynamic inverse sampling. In Proc. 31st Int. Conf. on Very Large Data Bases, 2005, pp. 25–36.
5. Frahling G., Indyk P., and Sohler C. Sampling in dynamic data streams and applications. In Proc. 21st Annual Symposium on Computational Geometry, 2005, 142–149.
6. Indyk P. Algorithms for dynamic geometric problems over data streams. In Proc. 41st Annual ACM Symp. on Theory of Computing, 2004, pp. 373–380.
7. Korn F., Muthukrishnan S., and Srivastava D. Reverse nearest neighbor aggregates over data streams. In Proc. 28th Int. Conf. on Very Large Data Bases, 2002, 814–825.
8. Preparata F.P. and Shamos M.I. Computational Geometry: An Introduction, 3rd edn. Springer, Berlin Hiedelberg New York, Oct. 1990.
9. Vitter J.S. Random sampling with a reservoir. ACM Trans. Math. Software, 11(1)37–57, 1985.

GEO-RBAC Model

Yue Zhang, James B. D. Joshi
University of Pittsburgh, Pittsburgh, PA, USA

Synonyms

LoT-RBAC

Definition

The central idea of the GEO-RBAC model is the role schema, and instances of roles, as defined below [1]:

- Role Schema: *A Role Schema is a tuple* $<r, ext, loc, m_{loc}>$, *where r is the name of the role, ext is the feature type of the role extent, loc is the feature type of the logical positions and mloc is the mapping function that maps a real position into a logical position of type loc.*
- Role Instance: *Given a role schema* r_s, *an instance* r_i *of* r_s *is a pair of* $<r, e>$ *where* $r = r_s.r$ *and* $e \in F$, *such that* $FT\_Type\ (e) = r_s.ext$.

The notion of permissions, users, sessions, permission-role assignment and user-role assignment in the GEO-RBAC are similar to those in the RBAC model. The role enabling status in GEO-RBAC depends on whether the user's physical location maps the logical location specified in the role schema, as defined below:

- Enabled Roles: *Enabled session roles are defined as the function*: $EnabledSessionRoles: SES \times RPOS \rightarrow 2^{RI}$ *such that* $EnabledSessionRoles\ (s,\ rp) = \{<r,\ e> \in R_I \mid <r,\ e> \in SessionRoles\ (s),\ lpos = SchemaOf(<r,\ e>).mloc(rp),\ Contrains\ (LocObj(e),\ LocObj\ (lpos)) = \text{TRUE}\}$

Finally, the control of the authorization in GEO-RBAC is defined as follows [1]:

- Authorization control function: An access request is a tuple $ar = <s, rp, p, o>$, where s is the session of the user, rp is the real positions of the user, p is permission requested, and o is the object of the permission. ar can be satisfied at position rp if

$$(p, o) \in \bigcup_{y \in EnabledSessionRoles(S, rp)} I\_PrmsAssignment^*(y)$$

where $I\_PrmsAssignment^*\ (y)$ represents those permissions that are directly assigned to role instance y or assigned to the role schema corresponding to y.

Key Points

The widespread deployment of location-based services and mobile applications, as well as the increased concerns for the management and sharing of geographical information in strategic applications like environmental protection and homeland security have resulted in a strong demand for fine-grained location based access control models and mechanisms. Efforts have been expended to extend existing RBAC models to develop fine-grained specification and enforcement framework for location based access control requirements.

The GEO-RBAC model, proposed by Bertino et al. [1], supports location constraints to address such location based access control needs. GEO_RBAC is based on the notion of *spatial role* that is a geographically bounded organizational function. The boundary of a role is defined as a geographical feature, such as a road, city, or hospital, and specifies the spatial extent in which the user has to be located in order to use the role. The location constraints and the authorization are integrated through enabled roles. If the user is located inside the spatial boundary of the role that has been selected (activated) during the session, the role is said to be enabled. The user can acquire all the permissions that are assigned to the enabled roles from all the roles available in his/her sessions.

As RBAC, GEO-RBAC encompasses a family of models. Core GEO-RBAC includes the basic concepts of the model, thus the notion of spatial role, role schema, real/logical position, and activated/enabled role. Hierarchical GEO-RBAC extends the conventional hierarchical RBAC model by introducing two distinct hierarchies – one over role schemas and one over role instances. Constrained GEO-RBAC supports the specification of separation of duty (SoD) constraints for spatial roles and role schemas. Since exclusive role constraints are important to support the definition and maintenance of access control policies in mobile contexts, SoD constraints are extended to account for different granularities (schema/instance level), dimension (spatial/non-spatial), and different verification time (static, dynamic at activation time, dynamic at enabling time).

Another model that addresses location based access in conjunction with temporal constraint is the Location and time-based RBAC (LoT-RBAC) model.

Cross-references

► General Based Role Based Access Control
► LoT-RBAC
► Role Based Access Control
► Temporal Access Control

Recommended Reading

1. Bertino E., Catania B., Damiani M.L., and Perlasca P. GEO-RBAC: a spatially aware RBAC. In Proc. 10th ACM Symp. on Access Control Models and Technologies, 2005, pp. 29–37.

Georeferencing

Jordan T. Hastings, Linda L. Hill
Department of Geography, University of California-Santa Barbara, Santa Barbara, CA, USA

Synonyms

Geospatial referencing; Spatial referencing

Definition

Georeferencing is the name given to the process of geospatially referencing data and information objects – datasets, text documents, maps, photographs and imagery, etc. – to their proper locations on Earth. The vast majority of such objects derive from measurements and observations phenomena that are inherently georeferenceable, because humans are largely confined in their activities to the near-surface of the Earth and "Almost everything that happens, happens somewhere" [3].

Georeferencing can be accomplished in two main ways [2]: formally, by assigning geospatial coordinates directly to data and information objects; and informally, by relating such objects to one or more pre-existing ones for which georeferences have already been established. In everyday life, the latter approach predominates, using the mechanism of *place*. A geographic place is a real-world location, perhaps vague, with a recognizable name and type, and optionally other attributes, e.g., "Northern California," Golden Gate Park. A *gazetteer* is a dictionary of such places, which is essential for translating back and forth between informal and formal georeferences.

Numerous problems and subtleties attend the formal, or direct, georeferencing process. The Earth is not a sphere, in fact, but a lumpy body that is cannot be exactly described geometrically. Indeed, multiple definitions exist for the fundamental geospatial coordinates – longitude, latitude, and surface elevation – all of which depend on a definition of the Earth's center. Map projections further confound matters. No projection of the globe onto a flat sheet, to create a map, can simultaneously preserve both area and shape, even locally; thus, all projections introduce distortions, which can be severe. In addition, most projections involve transcendental mathematical functions, which are only approximated by computers.

Other problems apply to informal georeferencing using named places, e.g., "Fisherman's Wharf" or "5 miles south of San Francisco." Except for officially designated and defined administrative regions, geographic places are inherently subjective, based in geospatial cognition and mediated by linguistic, historical, and social conventions. Places are therefore approximate in location, shape and size, and further subject to various typing schemes. What are the distinctions among "metropolis," "city," and "town," for example; or between "lake" and "pond"? A single name may apply to many places – for example, there are many Lake Genevas in addition to the famous water body in Switzerland; some are lakes, in fact, others are cities. Also, a single place can have multiple names and types – for example, New York City is colloquially known as "The Big Apple."

Historical Background

Geospatial referencing originated in marine navigation, where it is essentially a 2-D exercise, using knowledge of celestial mechanics and time. Latitude can be determined from a sighting of the sun at local noon, knowing the day of the year. Determining longitude requires a reliable shipboard time-of-day clock [5] as well. Positional accuracies on the order of kilometers, i.e., within range of sight, are typical.

On land, longitude and latitude are relatively unimportant in people's everyday affairs, supplanted by linear measurements of distance and direction along constrained routes, e.g., riverways and roadways. Even these measurements are subordinate in many cases to the observation of salient places, also known as landmarks, along the routes: topographic features, human settlements, water bodies, etc. Again, relatively low-technology tools suffice: a wheel to measure distance, a compass to indicate direction, and optionally a barometer to estimate elevation. Positional accuracies on the order of $\sim$100 m are suitable for most purposes.

In space, the position of every satellite must be meticulously tracked. A constellation of $\sim$30 satellites has been deployed specifically as a global positioning system (GPS). Several of these satellites are visible from most points on Earth at all times, although the particular satellites come and go throughout the day. GPS satellites continuously broadcast their positions on special radio frequencies. A companion GPS receiver on Earth that picks up the signal from three or more of these satellites can back-calculate its own horizontal position by trilateralization (a variant of triangulation); with four or more satellites, vertical position can be determined as well. From multiple "fixes" on position over time, the GPS receiver also can work out its speed and direction of movement, if any. Overall, this system provides positional accuracies of ∼5 m horizontally and ∼20 m vertically, which is more than sufficient for most georeferencing purposes.

Foundations

Technical Background

The Earth is a globe, but only approximately spherical, with an equatorial radius of 6,378,137.0 m and a polar radius of 6,356,752.3 m, by modern measurement (WGS84). This ellipsoidal description is also approximate, as the Northern hemisphere is squatter than the Southern hemisphere; in addition, both hemispheres are incised by oceans and elevated by continents. The exact 3-D "figure" of the Earth defies analytic mathematical description, in fact. For precise geodetic work, affecting rocket and satellite trajectories among other things, a gridded model of the Earth's shape is used. However, for most mapping applications, an ellipsoidal approximation is satisfactory [1].

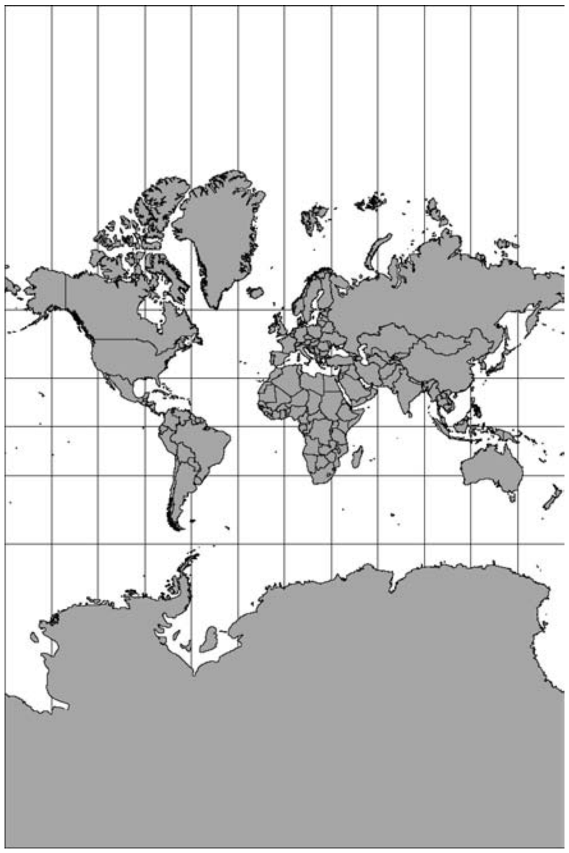
A *datum* is a set of parameters that define an approximate figure for the Earth, typically as an ellipsoid. The horizontal datum defines longitude and latitude with regard to this figure, and the vertical datum defines elevation in relation to an average sea level on it. Until the satellite era, the Clarke 1866 datum was sufficient and stable; since then a progression of datums (Table 1) have been adopted. A *map projection* depicts the globular figure of the Earth on a flat sheet of paper. In the projection process, lines of longitude and latitude (circles on the Earth) become other shapes on paper; inevitably, distortions are introduced. For example, the well known Mercator projection (Figure 1) depicts both longitudes and latitudes as straight lines, resulting in extreme distortion at the poles. No projection preserves both shape and area throughout the map, and many projections preserve neither (but still are useful).

Digital geospatial data representations are numerous. Three well established categories of such representation are: *vectors*, discrete shapes, applying to object-like phenomena; *rasters*, regular grids, best suited to field-like phenomena; and TINS (triangulated irregular networks), for 2½-D surfaces. Vector data further subdivides into points, lines, and polygons. With the exception of points and the simplest polygons – triangles and rectangles – none of these representations fits easily into a single relational database table; rather, table(s) containing open-ended "blob" (binary large object) structures, or a group of tables, are frequently used. A wide variety of proprietary structures exist for geospatial data transcribed to files. The Geographic Markup Language (GML) [4] is an application of XML designed to remedy this complexity by encoding

Georeferencing. Table 1. Selected world datums

Name	Semi-major axis (m)	Semi-minor axis (m)	Flattening ratio
Clarke 1866 (NAD27)	6378206.4	6356583.8	294.979
GCS 80 (NAD83)	6378137.0	6356752.3	298.257
WGS 84[a]	6378137.0	6356752.3	298.257
GCS 87 (Europe)	6378388.0	6356911.9	297.000

[a]Differs imperceptibly from GCS80, for most purposes



Georeferencing. Figure 1. Mercator projection of the world, with 30-degree graticule.

both vectors and rasters (but not TINS), including their associated attributes, in a portable text format.

Georeferencing Techniques

Georeferencing spans a variety of techniques, determined in part by the content and format of the underlying data and information objects. Some objects must be formally georeferenced with longitude and latitude coordinates. Aerial photographs and satellite imagery, for example, are usually located and oriented geospatially by the coordinates of their center and/or corner points. By contrast, text documents are typically georeferenced either by assigned geographic subject headings or by place names (in titles, indexes, and the text itself). Subsequently, using a gazetteer, these informal place references can be translated to coordinates. Hardcopy maps usually can be georeferenced in both ways, because they are drawn in a coordinate framework and also have place names embedded in titles, marginalia, map features, etc. Tabular materials represent another special case, because they can contain both place names and coordinates explicitly. Photographs, works of art, biological specimens, and other such objects are most often georeferenced informally in accompanying notes, although some digital cameras are now equipped with GPS and capable of recording formal coordinates directly with the image.

Following is a synopsis of common georeferencing techniques.

Datasets Scientific and technical data are frequently presented in tables, recorded in spreadsheets or databases. Such tables may include georeferences as coordinates, place names, or both. For statistical data, place codes may appear in lieu of place names for administrative or other formally defined areas, such as census tracts or postal zips. Data in tables may be georeferenced formally, informally using gazetteers, or both.

Documents Textual materials commonly contain geospatial references as place names, which may serve to locate the subject matter more or less precisely, viz. "Washington Mall" versus "apex of Washington Monument". Place names also may appear adjectively ("Washington Redskins' stadium"), prepositionally ("suburbs of Washington"), or they may be the subject itself ("Washington, DC"). A single text document may contain many place names, some relevant to the subject matter, some not, viz. "unlike its neighboring states of Maryland and Virginia, Washington, DC is a federal district". A technique called geoparsing is used to recognize salient place references in text documents and to associate coordinates with them via gazetteers.

Maps Maps are, by definition, depictions of geospatial phenomena in a coordinate framework, i.e., graphical displays of geospatial data. A significant portion of the geography curriculum at all levels concerns locating coordinate pairs on maps and conversely extracting the coordinate locations of mapped points. Linear and areal features are typically represented by sequences of points, which can be intricate. A GIS (below) is, to first approximation, a map in digital form. Map content is inherently georeferenced, but the maps themselves often require external, informal georeferencing.

Photographs It is useful to distinguish between photographs taken *in* the environment – "snapshots" – and photographs taken *of* the environment, typically from an aerial platform such as an airplane. The former may be georeferenced, either formally or informally, as objects at mapped points; the later cover a quasi-rectangular area on a map, indicated by a sequence of points. With aerial photographs, comes the additional issue of othro-rectification: removing the dilation or "bloom" in portions of the photograph except at the single point (if any) focused straight down.

Imagery Satellite imagery is an electronic rather than film product – so-called "digital numbers" (DNs), representing the brightness of a scene in a gridded pattern of picture elements (pixels). Images are taken from sufficient distance above the Earth that dilation issues are small, but not negligible. The satellite's orbit coupled with the Earth's rotation typically results in an image swath that tracks diagonally across the land surface, producing trapezoidal pixels. For ease of use, the trapezoids are usually interpolated back to square pixels during post-processing of the satellite data, in conjunction with ortho-rectification. Georeferencing then proceeds as for aerial photographs.

Key Applications

Georeferencing makes data and information objects more useful for many purposes. In a geographic information system (GIS), for example, vector objects, raster fields and imagery are presented in layers. When such layers are georegistered to a common standard, their content appears superimposed – as on an electronic light table. Users are frequently surprised by even slight mis-registrations of the layers. Hence, the intrinsic accuracy of geospatial measurements relating to the layers also must be considered.

Geographic information retrieval (GIR) is an extension of textual information retrieval (IR) that compares coordinate-based queries to georeferenced data and information objects to find the best matches. In a geospatially-enabled digital library, for example, the query "find information about Washington, DC" can be translated into coordinates for the area, from which a wide variety of materials can be retrieved, including satellite images, maps, and books. The spatial matching operations usually include containment, overlap, and proximity within a specified distance. As with GIS, issues of georeferencing accuracy as well as intrinsic errors must be considered.

A third, hybrid application, is geospatial modeling and analysis, in which GIS layers, are manipulated numerically to yield additional insights and results. When the physical laws governing the behavior of environmental phenomena are known, or hypothesized, these also can be integrated with the GIS layers to produce diagnostic and forecast models of environmental behavior. The purpose of such models, beyond scientific curiosity, is usually to inform political policy and decision-making.

Cross-references

► Gazetteers
► Geographic Information Retrieval
► Geographic Information Systems
► Spatial Operations and Map Operations

Recommended Reading

1. Clarke K.C. Getting Started with GIS, 4th edn. Prentice-Hall, Upper Saddle River, NJ, 2004.
2. Hill L.L. Georeferencing: The Geographic Associations of Information. MIT Press, Cambridge, MA, 2006.
3. Longley P.A., Goodchild M.F., Maguire D.J., and Rhind D.W. Geographical Information Systems and Science, 2nd edn. Wiley, New York, 2005.
4. Open Geospatial Consortium (OGC). Geography Markup Language (version 3.1.1).
5. Sobel D. Longitude: The True Story of a Lone Genius Who Solved the Greatest Scientific Problem of his Time. Penguin, New York, NY, 1992.

Geoscientific Information System

► Three-Dimensional GIS and Geological Applications

Geospatial Information System

► Geographic Information System

Geospatial Metadata

Bugra Gedik
IBM T.J. Watson Research Center, Hawthorne, NY, USA

Synonyms

Geographical metadata

Definition

Geospatial metadata are a type of metadata used to describe geospatial data. Data described by geospatial metadata relates to objects that have an explicit or implicit geographic extent or position on the surface of the Globe. Geospatial metadata can be used to create metadata catalogues or directories that describe geographic features of data stored in any environment, ranging from data stored in geographic information systems (GIS) to simple documents, datasets, images, or even services. Geospatial metadata captures the basic characteristics of geospatial data and represents the what, the when, the where, and the who of the geospatial data. A typical geospatial metadata record includes catalog elements such as title, abstract, and publication data; geographic elements such as geographic extent and projection information; and database elements such as attribute label definitions and attribute domain values.

Historical Background

The interest in geospatial metadata grew steadily over the 1980s and 1990s, and as a result several national and international organizations, as well as communities of practice, started to develop their own formats and conventions for collecting and organizing geospatial metadata. For instance, United States National

Aeronautics and Space Administration's (NASA) Directory Interchange Format (DIF) [2] was developed in 1987, and formally approved for adoption in 1988. A similar geospatial metadata development effort was undertaken by the United States Federal Geographic Data Committee (FGDC) throughout 1992 to 1994 [5], which led to FGDC's release of Content Standard for Digital Geospatial Metadata (CSDGM) standard in 1998 [3]. Similarly, in 1996 Australia and New Zealand Land Information Council (ANZLIC) released its metadata guidelines [1]. The emerging need for consolidating various formats and standards developed by different communities of practice was addressed by ISO/TC 211 over the years 1999 to 2002. Consequently, ISO 19115 standard was released in 2003 under the title "Geographic Information – Metadata." Following the release of the ISO 19115 standard, national and international organizations, as well as communities of practice have started the process of fitting their previously adopted metadata standards as profiles or recommended subsets of the ISO 19115 standard, via formal extensions to it. The increasing popularity and adoption of the Extensible Markup Language (XML) for sharing data across different information systems over the Internet led to the development of mechanisms for exchanging geographic metadata on the Web during the 1990s. Consequently, the Geography Markup Language (GML), an XML grammar for expressing geospatial features and metadata, was released in 2004 by the Open Geospatial Consortium (OGC). With the growth of the Semantic Web in the 2000s, several organizations have started to develop ontologies for representing semantic geospatial metadata. Some examples include the Hydrology and Administrative Geography ontologies [4] developed by the Ordnance Survey in the United Kingdom.

Foundations

Digital geospatial data are often used to create a model of the real world geographical objects for use in computer analysis and digital display of geospatial information. Hence, it is an abstraction of the real world geographical objects that are being described. This abstraction often involves various approximations, simplifications, and exclusions. Given that this digital representation is seldom perfect or complete, the assumptions and limitations affecting its accuracy must be fully documented to ensure that such data are not misused. Geospatial metadata allows a data producer to describe a geospatial dataset fully, so that the users can understand the assumptions and limitations associated with the dataset and evaluate its applicability for their intended use. In summary, geospatial metadata provides data producers with appropriate information to characterize their geospatial data, facilitates the organization and management of geospatial data, enables users to apply geospatial data in the most efficient way by knowing its basic characteristics, facilitates data discovery, retrieval and reuse, and enables users to determine whether geospatial data at hand will be of use to them.

Geospatial metadata consists of a schema required for describing geographic information and services. It provides information about the identification, quality, spatial and temporal extent, spatial reference, and distribution of geospatial data. The set of geospatial metadata elements is often large and typically only a subset of the full number of elements is used. For instance, ISO 19115 standard includes a list of core metadata elements – a minimum set of basic elements required to identify a dataset for cataloging purposes. These core metadata elements answer the following questions:

- Does a dataset on a specific topic exist? ("what" aspect)
- For a specific place? ("where" aspect)
- For a specific date or period? ("when" aspect)
- With a point of contact to learn more about the dataset? ("who" aspect)

GML: The Geography Markup Language

The Geography Markup Language (GML) is an XML grammar used to express geospatial features. GML serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. Originally developed by the Open Geospatial Consortium (OGC), it is later adopted as ISO 19136 standard in 2007.

Geospatial Metadata Catalog

A geospatial metadata catalog is a collection of geospatial metadata entries that are managed together. A geospatial metadata catalog is often accompanied by a catalog service, which responds to requests for metadata in the catalog and help browse or search geospatial data described by the collection of geospatial metadata managed in the catalog. In summary, the main goal of a geospatial catalog is to support a wide range of users in discovering relevant geospatial information from heterogeneous and distributed repositories of geospatial data.

Cross-references

▶ Geographic Information System
▶ Meta Data Management System

Recommended Reading

1. ANZLIC Metadata Working Group. ANZLIC Metadata Guidelines: Core metadata elements for geographic data in Australia and New Zealand, Version 2. http://www.anzlic.org.au/get/2358011755.
2. Major G. and Olsen L. A short history of the DIF. On GCMD website, http://gcmd.nasa.gov/User/difguide/.
3. Metadata Ad Hoc Working Group Federal Geographic Data Committee. FGDC website, http://www.fgdc.gov/metadata/csdgm/.
4. Ordnance – National Mapping Agency of Great Britain. Ordnance Survey Ontologies. Ordnance Website, http://www.ordnancesurvey.co.uk/oswebsite/ontology/.
5. Wolfe R. MIT Libraries website, http://libraries.mit.edu/guides/subjects/metadata/standards/fgdc.html.

Geospatial Referencing

▶ Georeferencing

Geo-Targeted Web Search

Torsten Suel
Yahoo! Research, Sunnyvale, CA, USA

Synonyms

Geographic web search; Local web search

Definition

Geo-Targeted Web Search (Geo Search) describes a set of techniques used in web search engines that aim to return more relevant results to users by exploiting geographic knowledge. Most commonly, a Geo Search Engine (a search engine using Geo Search) tries to infer a geographic area of interest for a user query by considering the query itself, the current location of the user, or the search history of the user. The Geo Search Engine then tries to return documents that are relevant to the query search terms and that also match the geographic area of interest of the query, as determined through analysis of the web page itself, any geographic meta tags (geo tags), link or site structure, or other data. As an example, a Geo Search Engine may return to a mobile user a list of restaurants close to the user's current location given a query "restaurants." Or a user may explicitly ask for information about tourism attractions in Florida, by adding a keyword such as "florida" or pointing to a map. Geo Search techniques can be integrated into standard broad-based or specialized search engines, or deployed as separate search facilities with special interfaces that allow more convenient query input or result display.

Historical Background

Geo Search developed over the last decade as a response to several challenges faced by search engines. First, given the size of the web, there are often too many potentially relevant results, and geography can provide a natural way of filtering such results for those the user really wants. Second, much of commerce is still local in nature, and Geo Web Search can provide a platform for supporting local advertising models on the web. Third, it was found that treating geographic terms in queries just as normal terms does not give the best possible results, and that some amount of geographic domain knowledge should be added to engines to improve results. For example, by simply matching on the term "manhattan" in a query, one might miss out on results referring to New York in general or to a particular neighborhood in Manhattan, or results about Brooklyn that are close enough to be of interest. As a result, there have been significant efforts by major search engine companies and start-ups to develop Geo Search Engines, and there has also been increasing interest in the academic community.

Even before the emergence of the web, researchers in Information Retrieval studied how to exploit geographic information embedded in text documents for better text search and analysis; see Larson [10] for earlier work and Jones and Purves [8] for a recent overview of the slightly more general area of *Geographic Information Retrieval* (GIR). This article focuses on geographic web search technology, which currently dominates the area of GIR in terms of number of users and total economic investments, but there are of course important applications outside the web domain. Initial work on Geo Search on the web appears in [5,6,13], and in recent years a significant amount of research has addressed this new challenge. A lot of the work has studied the problem of reliably identifying geographic references in documents in the presence of noise and ambiguity [1,11–13]; for example, many common place names (e.g., Washington) can also refer to persons, hotels, or other entities, while others can refer to several different places (e.g., Paris, Texas vs. Paris, France). Recent work has also focused on indexing and query processing techniques for Geo Search [2,15,17], and on analyzing and studying user queries in order to better understand the types of geographic search needs that real users have [7,9,14,16].

Foundations

This discussion begins with an outline of the main tasks in Geo Search. As in conventional web search engines, these tasks can be divided into four groups: (i) data acquisition (i.e., web crawling), (ii) preprocessing and data analysis, (iii) index building, and (iv) processing of user queries using these index structures. Geo Search engines that are limited to a particular geographic area need to be able to efficiently crawl sites and documents related to this area; this problem is called *geo crawling*. In the preprocessing phase, the collection has to be analyzed to extract geographic information that can be used for indexing. This can include, e.g., complete addresses, phone numbers, town names, or informal references (such as "big apple" for New York). The output of this phase may be either sets of geographic terms, or sets of points, polygons, or other geometric objects. During index building, additional index structures may have to be built, e.g., spatial index structures for the geometric objects obtained from the previous phase. Finally, during query processing, queries and their contexts are analyzed to determine if the user might be interested in a particular geographic area, and then the index structures is used to find relevant documents.

Geo Parsing and Geo Coding

A significant amount of work has focused on (ii), the analysis of documents for geographic information [1,11–13]. In particular, *geo parsing* is the process of identifying likely geographic references (such as town names, addresses, etc.) in the document text or meta data. Most commonly, this is performed by matching terms in the documents with databases of geographic names (gazetteers) obtained from outside sources, such as the GNIS database made available by the US Geological Survey for the US. If databases of business listings (yellow pages) are available, these can also be used to extract additional geographic references. In addition, techniques for named entity recognition can be applied. When databases of geographic names are available, the problem may appear simple but is in fact not so. Consider the many ways in which city or state names may appear as part of names of persons, businesses, foods, or many other items. Moreover, Washington can be a city or a state, LA can refer to Los Angeles or Louisiana, and there are dozens of towns named Lexington. These problems are referred to as geo/non-geo ambiguity and geo/geo ambiguity, respectively. In addition, it can be desirable to also detect informal place names (e.g., "big apple" for New York).

An alternative proposal is to introduce *geo tags* that allow authors to attach appropriate geographic terms or coordinates to pages [3] or other objects such as images or business listings or reviews, or allow communities of users to supply tags. This avoids many of the challenges of automatic extraction, but potential problems are lack of participation and possible manipulation.

Once the geographic terms have been extracted, there are two main approaches for further processing. One approach performs *geo coding* to map the geographic references into longitude-latitude coordinates, and then uses spatial or geometric algorithms and data structures for further processing. Alternatively, the geographic terms can be kept in textual form and related to a geographic ontology, in the simplest case just a hierarchical grouping of geographic terms representing simple spatial facts (e.g., that San Jose is in Santa Clara County which is in California), that is used for subsequent processing.

The geo coding approach is now discussed in more detail. In this step, the goal is to infer for each document its *geographic focus* [13], which is the area that the document relates to. The estimate of this area will be stored as a spatial data structure called the document's *footprint*. To compute the footprint of a document, any references to countries, cities, counties, or maybe businesses in the document are first translated into coordinates, using coordinate data available as part of many gazetteers. The output is a set of points, rectangles, polygons, or other spatial objects, possibly with appropriate weights or amplitudes. For example, a business may be mapped to a precise coordinate, while a city could be modeled by a circle or rectangle with some diameter, or a polygon approximating city boundaries. Weights might model the importance of a geographic term in the document or the extraction confidence for this term. In general, the geographic footprint of a document might assign an amplitude to every location on a map, with a value of zero for areas that are not relevant to the document, and very high values for areas that are referenced multiple times in the document. Efficient processing will usually require approximation of this data by a small number of polygons or bounding rectangles.

Note that both geo parsing and geo coding could potentially be improved by using site and link structure [12,13], such that pages with ambiguous or missing geographic references are analyzed using information on neighboring pages. It may also be useful to assign footprints on a per-site basis. Finally, note that the geographic focus of a page is related but not identical to the *geographic scope* introduced in [5], which is the geographic area of the readership addressed by a page. For example, a page about travel to Paris in a local newspaper or site in Seattle might have a geographic focus around Paris, but a geographic scope around Seattle.

Indexing and Query Execution

After analyzing the documents for geographic references, it is necessary to build appropriate index structures that allow efficient query processing. Recall that in the analysis phase there were two main approaches, one that keeps extracted geographic references as terms that can be related to some spatial ontology or other knowledge base, and one that translates the extracted geographic terms into a spatial footprint structure based on polygons or rectangles. The former approach then uses textual index structures and techniques such as query expansion on the geographic terms in the user's query [4] in order to find the best results.

Lot of recent work, e.g., [2,15,17], follows the latter approach, which is now discussed in more detail. Assume that a query to a geo search engine consists of two parts, a set of terms as in a standard engine, and a *query footprint* in the same basic format as the document footprints. Such a query footprint could be obtained from the user in a number of ways, e.g., by having the user input a geographic term or point and click in a map, or by using the user's current location or past queries to derive a likely area of interest. Then a possible definition of query processing in geo search engines is as follows:

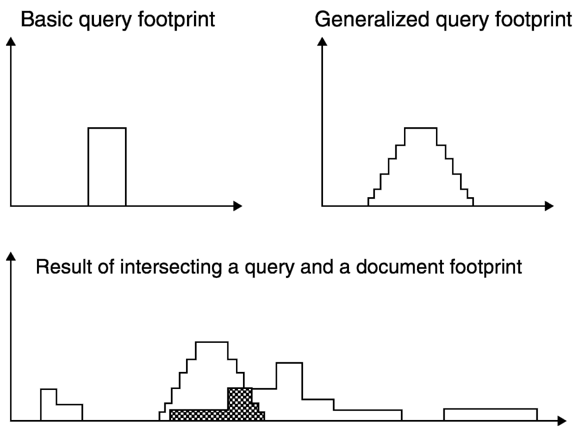
Definition: Given a collection of documents, each consisting of a document footprint and a set of terms, and a query consisting of a query footprint and a set of terms, the top-k query processing problem in geographic search engines is to return k documents such that:

- The footprint of each returned document has a nonempty intersection with the query footprint, i.e., there exists a location where both footprints are non-zero.
- Each returned document contains all terms in the query, or alternatively satisfies some weaker Boolean filter on the query terms (e.g., an OR).
- The returned documents are the k highest scoring documents according to a ranking function of the form $h(s_t,s_g)$ where s_t is a term-based score computed on the query and document terms, s_g is a geographic score obtained from the query and document footprints, and h is a monotone function for aggregating the two scores.

Some of the above assumptions can be relaxed, for example by allowing s_t to also contain scores based on link analysis or page visit frequency. One interesting choice for s_g is a dot product over all locations of the two footprints (or a corresponding integral in the continuous case). As shown in Fig. 1, by properly preprocessing the footprints it is possible to achieve objectives such as scoring documents higher if they are closer to the center of the query footprint or if they overlap with locations that are of particular interest to the user.

Thus, for fast query processing, it is necessary to have index structures that support execution of the spatial and textual filtering conditions, and an efficient way to score the remaining documents. In typical web collections, the textual data to be indexed is larger than the extracted spatial data, and thus efficient text index structures, such as inverted lists, are crucial. The size of the spatial data depends on the format used for the footprints. If the average footprint consists only of a few bounding rectangles, then this data may be only a few percent of the text data and thus may fit in main memory. If a footprint contains a more precise representation involving polygons or amplitudes, then the spatial data may only be moderately smaller than the text data.

In any case, to perform query processing, both textual and spatial index structures are required, and the challenge is to integrate these efficiently. A number of algorithms for this problem were first proposed in [15]. Two recent optimized approaches [2,17]


Basic query footprint
Generalized query footprint
Result of intersecting a query and a document footprint

Geo-Targeted Web Search. Figure 1. [12] An illustration of query and document footprints in a single spatial dimension. The top shows a query footprint with a distance threshold (*left*), and a footprint for a query that gives a lower score for documents that are farther away from the center (*right*). The bottom shows an intersection between the query footprint and a document footprint.

address this issue by combining inverted indexes with spatial structures such as R*-trees and space-filling curves. The approach in [17] assumes that each footprint is a set of rectangles; this results in a fairly small amount of spatial data and very fast query processing as most spatial data will be in memory. Work in [2] assumes that footprints are approximated by a set of bounding rectangles that is used during the filtering phase, but that a more precise representation of the footprints is also stored and must be fetched in order to compute precise values for s_g. The precise footprints are treated as binary objects (BLOBs) whose detailed structure is not important as long as a score can be computed from them; they might for example contain amplitudes or additional context about the geographic terms that were parsed from the documents. Compared to [17], this approach results in more spatial data that needs to be accessed, and thus more disk transfers, but it is more general in terms of the types of ranking functions that can be supported. However, it is not clear to what degree this generality is actually needed, and the approach in [17] may suffice in many practical scenarios.

Geographic Search Queries

In order to provide useful geographic search services, it is necessary to study how users search on the web. Several recent studies have looked at how to automatically identify queries that have a geographic intent and could thus benefit from geographic search technology [7,9,16]. Automatic identification allows a proper query footprint to be assigned to such queries. It may also allow better processing of queries that have no explicit geographic terms. For example, when searching from mobile devices or sometimes also from desktops, users may prefer results close to their current location without explicitly specifying so. Other researchers have studied the basic properties of such queries [14]. It is estimated that about 10% to 15% of queries submitted to standard search engines contain geographic terms. Work in [9] looks at how users modify the geographic terms in query sessions in order to get better results.

Key Applications

All the major search engine companies have developed and deployed geographic search technology as part of their offerings. In particular, these engines provide local search options that allow users to search for businesses and organizations using keywords, geographic terms, and map-based interfaces. In addition, geographic search technology is also used in their standard web search engines to provide better results for queries that are likely to have a geographic intent. Closely related and of particular commercial interest to the search engines are techniques for local advertising on the web.

Other applications on the web are in the automatic compilation of local information for city or neighborhood portals, and in providing better interfaces and search for real estate, car trading, or classified sites where users tend to search in a certain geographic area. Finally, although this discussion has focused on the case of the web, Geographic Information Retrieval has many applications on other textual collections where geographic search, browsing, and analysis are useful.

Future Directions

There is continuing research on many aspects of geographic web search, which is still in its infancy. This includes work on better extraction of geographic terms from documents and queries using techniques from Natural Language Processing and Information Retrieval. Ideally, techniques should be general enough to be applicable across languages, and it would be desirable to also identify implicit geographic intent in queries and informal place names.

Further studies of search logs and user behavior are needed to better understand user intentions and search strategies, particularly for queries that go beyond the types of searches for local businesses that are already handled fairly well by current local search services. Growing interest in technologies for local advertising on the web is also expected.

An intriguing future direction in geographic search involves online virtual worlds, such as Second Life as well as models of the real world such as Google Earth. As such models become more popular and users become more comfortable in navigating them, significant textual content is expected to migrate to or be mapped into these spaces, resulting in yet unexplored research problems related to geographic web search.

Cross-references

▶ Geographic Information System
▶ Information Retrieval
▶ Inverted Index
▶ Named Entity Extraction
▶ Rtree
▶ Search Log Analysis
▶ Space-Filling Curves
▶ Spatial Network Databases
▶ Web Search Engines

Recommended Reading

1. Amitay E., Har'El N., Sivan R., and Soffer A. Web-a-where: geotagging web content. In Proc. 27th ACM SIGIR Conf. on Research and Development in Information Retrieval, 2004, pp. 273–280.
2. Chen Y., Suel T., and Markowetz A. Efficient query processing in geographic web search engines. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2006, pp. 277–288.
3. Daviel A. Geographic registration of HTML documents, Apr 2001.
4. Delboni T., Borges K., and Laender A. Geographic Web search based on positioning expressions. In Proc. Workshop on Geographic Information Retrieval, 2005, pp. 61–64.
5. Ding J., Gravano L., and Shivakumar N. Computing geographical scopes of web resources. In Proc. 28th Int. Conf. on Very Large Data Bases, 2000, pp. 545–556.
6. Egenhofer M. Toward the semantic geospatial web. In Proc. 10th ACM Int. Conf. on Advances in Geographic Information Systems, 2002, pp. 1–4.
7. Gravano L., Hatzivassiloglou V., and Lichtenstein R. Categorizing web queries according to geographical locality. In Proc. 12th ACM International Conference on Information and Knowledge Management, 2003, pp. 325–333.
8. Jones C. and Purves R. Geographic information retrieval. Int. J. Geo-Graph. Inform. Sci., 22(3):219–228, March 2008.
9. Jones R., Zhang V., Rey B., Jhala P., and Stipp E. Geographic intention and modification in web search. Int. J. Geograph. Inform. Sci., 22(3):229–246, 2008.
10. Larson R. Geographic information retrieval and spatial browsing. In GIS and Libraries: Patrons, Maps and Spatial Information. Linda Smith and Myke Gluck (eds.), 1996, pp. 81–124.
11. Leidner J. Toponym resolution in text: Which Sheffield is it? In Proc. 27th ACM SIGIR Int. Conf. on Research and Development in Information Retrieval, 2004, pp. 602–602.
12. Markowetz A., Chen Y., Suel T., Long X., and Seeger B. Design and implementation of a geographic search engine. In Proc. 8th Int. Workshop on the Web and Databases, 2005.
13. McCurley K. Geospatial mapping and navigation of the web. In Proc. 10th Int. World Wide Web Conference, 2001, pp. 221–229.
14. Sanderson T. and Kohler J. Analyzing geographic queries. In Proc. of the Workshop on Geographic Information Retrieval, 2005.
15. Vaid S., Jones C., Joho H., and Sanderson M. Spatio-textual indexing for geographical search on the web. In Proc. Nineth Int. Symp. on Spatial and Temporal Databases, 2005.
16. Wang L., Wang C., Xie X., Forman J., Lu Y., Ma W., and Li Y. Detecting dominant locations from search queries. In Proc. 28th ACM SIGIR Int. Conf. on Research and Development in Information Retrieval, 2005.
17. Zhou Y., Xie X., Wang C., Gong Y., and Ma W. Hybrid index structures for location-based web search. In Proc. Int. Conf. on Information and Knowledge Management, 2005, pp. 155–162.

GIS

▶ Semantic Modeling for Geographic Information Systems

GIST

▶ Generalized Search Tree (GiST)

GiST

▶ Generalized Search Tree (GiST)

Global Query Optimization

▶ Multi-Query Optimization

Glyphs

▶ Iconic Displays

GMAP

Steven M. Beitzel[1], Eric C. Jensen[2], Ophir Frieder[3]
[1]Telcordia Technologies, Piscataway, NJ, USA
[2]Twitter, Inc., San Fransisco, CA, USA
[3]Georgetown University, Washington, DC, USA

Synonyms

Geometric mean average precision

Definition

The Geometric Mean Average Precision (GMAP) is the geometric mean of the average precision values for an information retrieval system over a set of n query topics. GMAP is expressed as follows (from [1]):

$$\mathrm{GMAP} = \sqrt[n]{\prod_{n} AP_n}$$

where AP represents the Average Precision value for a given topic from the evaluation set of n topics. Please refer to the entry on *Average Precision* for the complete definition of AP. An alternate calculation method expresses GMAP as an arithmetic mean of logs (also from [1]):

$$\mathrm{GMAP} = \exp \frac{1}{n} \sum_{n} \log AP_n$$

Key Points

The Geometric Mean Average Precision evaluation metric was first introduced in the Robust track at the 2004 NIST Text Retrieval Conference (TREC). A stated goal of the robust track was to examine and evaluate information retrieval systems with a view towards achieving consistently good performance over all query topics in the evaluation set. The geometric mean was considered for this purpose because it tends to better illustrate large relative improvements in the scores of poorly performing topics between a set of runs while suppressing the effect of minor fluctuations in the scores for topics that achieve good performance. By contrast, the traditional Mean Average Precision metric, which uses the arithmetic mean of average precision scores, gives equal weight to absolute changes in per-topic scores, regardless of the relative size of the change.

As an example, consider a run with five topics, each having the following Average Precision Scores:

	Topic #1	Topic #2	Topic #3	Topic #4	Topic #5
Run 1 AP	0.05	0.10	0.50	0.50	0.75
Run 2 AP	0.10	0.30	0.45	0.45	0.60

In this example, Run 1 and Run 2 have the same Mean Average Precision (0.38), whereas Run 1 has a GMAP of ~0.25 and Run 2 has a GMAP of ~0.33. If the evaluator is interested in a measure of consistency and collective performance across all topics, GMAP is a good choice for the evaluation metric.

Cross-references

▶ Average Precision
▶ Effectiveness Involving Multiple Queries
▶ Mean Average Precision

Recommended Reading

1. National Institute of Standards and Technology. TREC-2004 common evaluation measures.

Google Bombing

▶ Web Spam Detection

Grammar Induction

▶ Grammar Inference

Grammar Inference

Matthew Young-Lai
Sybase iAnywhere, Waterloo, ON, Canada

Synonyms

Automata induction; Grammatical inference; Grammatical induction; Grammar induction; Automatic induction; Automatic language induction

Definition

Grammar inference is the task of learning grammars or languages from training data. It is a type of inductive inference, the name given to learning techniques that try to guess general rules from examples.

The basic problem is to find a grammar consistent with a training set of positive examples. Usually, the target language is infinite, while the training set is finite. Some work assumes that both positive and negative examples are available, but this is not true in most real applications. Sometimes probability information is attached to each example. In this case, it is possible to learn a probability distribution for the strings in the language in addition to the grammar. This is sometimes called *stochastic* grammar inference.

A grammar inference algorithm must target a particular grammar representation. More expressive representations are more difficult and expensive to learn. For example, context-free grammars are more expressive than regular languages, but much harder to learn. The target representation for stochastic grammar inference can be viewed as separate structure and probability components. The structure component is a standard grammar. The probability component defines the probabilities associated with individual strings. One way to define this association is to assign a probability to every production of the grammar, or for regular grammars, to every transition of the finite automaton. The probability of a string is then the product of the production or transition probabilities used in its generation.

As with all learning problems, a solution is not necessarily acceptable just because it is consistent with the training set. Infinitely many grammars are consistent with any finite sample. Inference must construct a possible grammar, and also choose from among the possibilities. Usually the grammar must generalize outside the available examples in some useful way. What is useful varies with the application. Some applications require a result that a human can interpret. Others require a grammar that assigns accurate probability predictions to strings.

Historical Background

Grammar inference has been studied by researchers in many fields including information theory, formal languages, automata theory, language acquisition, computational linguistics, machine learning, pattern recognition, computational learning theory, neural networks, and data compression. Several conferences and workshops focus specifically on the subject. One example is the International Colloquium on Grammatical Inference (ICGI) which has been held in 1993, 1994, 1996, and every 2 years through 2006 so far. There have also been associated language learning competitions such as the Abbadingo competition which focused on learning deterministic finite automata and the Omphalos competition that focused on learning context free languages.

The first studies of grammar inference date back to the 1960s. One formalization is called *language identification in the limit* [7]. This assumes two sets of strings: $R+$ is the set of positive examples, and $R-$ is the set of negative examples. A language is said to be *identifiable in the limit* if it can be learned by *some method* for *sufficiently large* $R+$ and $R-$. Put another way, adding new examples to $R+$ and $R-$ must only produce a finite number of changes to the hypothesized model.

There are two decidability results in this formalization. The first is negative and says that no infinite language can be identified in the limit from only positive examples. This is intuitively a consequence of over-generalization since adding more positive examples can never imply that a string mistakenly included in the model should subsequently be removed. In fact, a consistent generalization of any set of positive examples is a language containing all finite strings constructed from the alphabet.

The second decidability result is positive and states that any member of an enumerable class of recursive languages (context-sensitive and below) is identifiable in the limit given both positive and negative data. However, this result says nothing about how large a sample is needed and therefore has limited application. Even if negative examples are available, which is often not the case, the learning problem may be intractable. For example, finding the smallest finite automaton compatible with given positive and negative example sets is NP-hard [3,8]. A polynomial time algorithm can construct a *non-minimal* solution to this problem in the limit [11]. However, the problem remains that there is no bound on the size of the required training data.

Within the standard Chomsky hierarchy from automata theory [10], grammar inference research has focused mainly on regular and context free grammars. Restricted grammars, referred to as *characterizable subclasses*, are also commonly targeted. For example, *k*-reversible languages [4], generalize from 0-reversible languages which are those generated by finite automata that remain deterministic if all of their arcs are reversed. The main justification for using restricted language classes is to reduce the time complexity of the inference process. The defining restrictions are not themselves relevant to real applications.

Much work has been done on stochastic grammar inference because of its applicability to real applications. Early work in information theory used probability information to discover rules. Shannon proposed that language learning can be seen as a problem of estimating probabilities of sequences of *n*-grams as *n* approaches infinity. Well known techniques such as the Baum-Welch algorithm [5] for learning hidden Markov models can be considered stochastic grammar inference since hidden Markov models are equivalent to stochastic finite automata. Many learning algorithms for data compression and neural networks perform stochastic grammar inference, although not always using representations that map to standard grammars.

Foundations

A common grammar inference paradigm starts with a model constructed to accept the finite language composed of exactly the strings in the training set. The inference process then transforms this *de-facto* model by applying generalization operations. This is analogous to a natural learning process that starts with a complete memory of specific examples and then combines groups of similar examples into general rules.

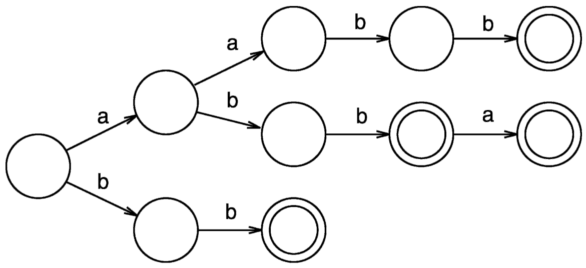
The form of the de-facto model depends on the target representation. The training set {*abb*, *bb*, *aabb*, *abba*}, for example, can be represented by the de-facto grammar in Fig. 1, the de-facto stochastic finite automaton in Fig. 2, or by appropriately chosen models in other representations.

Generalization operations can usually be viewed as merging sub-components of the model. Finite automata, for example, are generalized by merging states, thus creating new loops or paths. Grammars are generalized by merging productions. This expands the language so that it includes new strings with characteristics similar to existing strings. Operations that arbitrarily expand the language (e.g., by adding symbols to the alphabet or adding strings that have nothing in common with any of the training examples) are also possible, but not generally useful.

While the paradigm of moving from specific to more general models is common, the opposite strategy is also possible. This involves starting with a completely general model and then modifying it to describe specific

$$
\begin{aligned}
S &\rightarrow abb\\
S &\rightarrow bb\\
S &\rightarrow aabb\\
S &\rightarrow abba
\end{aligned}
$$

Grammar Inference. Figure 1. A de-facto grammar.



Grammar Inference. Figure 2. A de-facto stochastic finite automaton.

rules or patterns. This is done in the previous example by starting with the finite automaton having a single state and transitions back to that state on input of *a* and *b*. If it is then observed that *b*s always occur in pairs, then a new state can be split off to express this rule.

Generalization from a specific model and specialization from a general model are two paradigms for moving through the infinite space of all languages consistent with a given training set. Also needed is a specific control strategy that defines the order of generalization or specialization operations and when the process terminates. If a metric is available for evaluating intermediate results, then the task can be seen as a search problem, and many well known control strategies such as greedy search are applicable. However, many grammar inference algorithms are not explicitly formulated in this way. Instead, they choose the control strategy, the movement operations, or the representation in some way that is computationally convenient. The use of characterizable subclasses mentioned above is one example of this.

Stochastic grammar inference works with training data that can be viewed as a statistical sample of an unknown distribution. This means that candidate models can be evaluated in a relatively objective manner using either statistical tests or metrics based on information theory. In particular, it is possible to evaluate strings that are generated by the model but are not present in the training data. Probability information thus makes up for the lack of negative examples. Such unseen strings must be assigned low enough probability to fit the assumption that the training set is a random sample of the language generated by the model. Consider the training set $T = \{(a, 20), (aa, 10), (aaa, 5)\}$ consisting of three (*string*, *frequency*) pairs for a total frequency of 35. One possible model is $\{a^n | p(a^n) = 0.5^n\}$ which assigns a total probability of 0.125 to strings of four or more *a*s even though no such strings are present in the sample. A simple statistical test can verify that since T contains only 35 strings it could in fact have realistically been generated by the model. For example, a χ^2 test shows that the chance of getting T or any less likely sample from the model is somewhere between 0.20 and 0.50 – probably not unlikely enough to conclude that the model is a bad one. For the set $T' = \{(a, 200), (aa, 100), (aaa, 50)\}$, on the other hand, the probability is less than one in a thousand. The reason is that a random sample of 350 strings from the model should have included at least a few longer strings such as *aaaa* or *aaaaa*. Alternative evaluation criteria can be based on things other than strict probability of occurrence. Other approaches include Bayesian comparison to prior probability distributions [14] and divergence comparisons based on information theory [12].

Key Applications

The traditional application domain of grammar inference has been syntactic pattern recognition. This assumes that pattern classes can be represented by grammar models, and it is useful to learn these models automatically. Other application areas include speech and natural language processing, optical character recognition, information retrieval, gene analysis, sequence prediction, and cryptography.

In the database context, most recent work on selectivity estimation for query optimization in XML databases can be classified as stochastic grammar inference. Grammar inference also applies to data mining and knowledge discovery. Stochastic grammar inference is a key part of many compression techniques where better models directly correspond to better compression ratios.

For structured document collections, grammars are a natural representation to use as a database schema. This is also true of semi-structured and XML databases. Grammar inference is applicable to such data if it has no explicitly-created schema and there has been some work on grammar inference in this context. Fankhauser and Xu [6] describe a system that learns a grammar for use in automatic markup. Shafer [13] describes a C++ library, the GB (Grammar Builder) Engine which produces a grammar by generalizing examples marked up in an SGML-style. Work by Ahonen et al. [1,2] uses a more classical grammatical inference approach based on a characterizable subclass of regular languages that they call $(k{-}h)$ contextual languages. Goldman and Widom [9] describe the use of automatically constructed DataGuides for browsing database structure and formulating queries. Young-Lai and Tompa [15] look at applying stochastic grammar inference to the problem.

Cross-references

▶ Hidden Markov Models
▶ Inductive Inference
▶ Information Theory
▶ Machine Learning
▶ Markov Models
▶ Schema
▶ XML Selectivity Estimation
▶ XML Database

Recommended Reading

1. Ahonen H., Mannila H., and Nikunen E. Generating grammars for SGML tagged texts lacking DTD. In Proc. of the Workshop on Principles of Document Processing, 1994.
2. Ahonen H., Mannila H., and Nikunen E. Forming grammars for structured documents: an application of grammatical inference. In Lecture Notes in Computer Science, 862. R. Carrasco, J. Oncina (eds.), 1994, pp. 153–167.
3. Angluin D. On the complexity of minimum inference of regular sets. Inf. Control, 39:337–350, 1978.
4. Angluin D. Inference of reversible languages. J. ACM, 29:741–785, 1982.
5. Baum L.E., Petrie T., Soules G., and Weiss N. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. Ann. Math. Statist., 41(1):164–171, 1970.
6. Fankhauser P. and Xu Y. MarkItUp! An incremental approach to document structure recognition. Electron. Publ. Orig. Dissem. Des., 6(4):447–456, December 1993.
7. Gold E.M. Language identification in the limit. Inf. Control, 10:447–474, 1967.
8. Gold E.M. Complexity of automaton identification from finite data. Inf. Control, 37:302–320, 1978.
9. Goldman R. and Widom J. DataGuides: enabling query formulation and optimization in semi-structured databases. In Proc. 23th Int. Conf. on Very Large Data Bases, 1997, pp. 436–445.
10. Hopcroft J.E. and Ullman J.D. Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading, MA, 1979.
11. Oncina J. and García P. Inferring regular languages in polynomial updated time. In Pattern Recognition and Image Analysis. N.P de la Blanca, A. Sanfeliu, E. Vidal (eds.). World Scientific, Singapore, 1992, pp. 49–61.

12. Sánchez J.A. and Benedí J.M. Statistical inductive learning of regular formal languages. In Lecture Notes in Computer Science, 862. R. Carrasco, J. Oncina (eds.), 1994, pp. 130–138.
13. Shafer K. Creating DTDs via the GB-engine and Fred, 1995.
14. Stolcke A. and Omohundro S. Inducing probabilistic grammars by Bayesian model merging. In Lecture Notes in Computer Science, 862. R. Carrasco, J. Oncina (eds.), 1994, pp. 106–118.
15. Young-Lai M. and Tompa F.W. Stochastic grammatical inference of text database structure. Mach. Learn., 40(2):111–137, 2000.

Grammatical Induction

► Grammar Inference

Grammatical Inference

► Grammar Inference

Graph

Hans Hinterberger
ETH Zurich, Zurich, Switzerland

Synonyms

Graph; Chart; Plot

Definition

Graph theory: A set of nodes (also called **points** or **vertices**) connected by links called **lines** or **edges** or **arcs**. In an **undirected graph**, a line from point ***A*** to point ***B*** is considered to be the same thing as a line from point ***B*** to point ***A***. In a **directed graph**, the two directions are counted as being distinct **arcs** or **directed edges**.

Mathematics: A diagram exhibiting a relationship, often functional, between two or more sets of numbers as a set of points having coordinates determined by the relationship. Also called a plot.

Computer science: A data structure representing relationships or connections in lists, trees, and networks.

Data visualization: Any pictorial device such as a point graph, surface graph or symbol graph used to display numerical relationships. Also called a chart because graphs constitute one of the major categories of charts. Graphs combine two or more straight or circular axes, utilizing one or more quantitative scales. Straight axes are typically drawn perpendicular to each other, sharing a common origin. The most notable exceptions are triangular graphs, nomographs, and parallel coordinates.

Triangular (Trilinear) Graphs place the axes either along the sides or the along the angle bisectors of an equilateral triangle.

Nomographs (Nomograms) arrange the axes in such a way that a straight index line connecting known values on two axes passes through the solution value on another axis. The most common nomographs consist of three parallel axes, some place the axes at an angle to each other, others employ one or more curved axes.

Circular axes are based on polar coordinates, using angle or radius or both to represent values.

Key Points

Because graphs can be powerful tools to reveal the structure of data they are the preferred method to visualize data in science and technology, particularly as a tool for exploratory data analysis.

The use of graphical images to show data has evolved during the past 250 years, reaching unprecedented popularity with computer supported methods. The basic ideas behind good graphical design, however, transcend technology as they follow principles dictated by the human visual system.

W.S. Cleveland [1] has studied methods and basic principles that must be observed if a data analyst wants to realize a graph's potential for visualization. A classic on issues of graphical practice, including a theoretical approach to data graphics has been published by E.R. Tufte [3]. R.L. Harris [2] has assembled a comprehensive overview of different graphs by organizing them categorically according to their type of configuration (e.g., line graph, contour graph), area of application (e.g., technical, business), type of data plotted (e.g., original data, derived data), number of axes, and by purpose (e.g., analysis, monitoring, presentation).

Graphs are not only used to visualize data, they also serve as a tool for data acquisition and storage carried out by mechanical plotting devices or, in the form of nomograms, function as calculating devices, usually designed to perform a specific calculation.

Cross-references

► Chart
► Data Visualization

▶ Exploratory Data Analysis
▶ Parallel Coordinates

Recommended Reading

1. Cleveland W.S. The Elements of Graphing Data (revised edn.). Hobart Press, Summit, NJ, 1994.
2. Harris R.L. Information Graphics: A Comprehensive Illustrated Reference, Oxford University Press, Oxford, 1999.
3. Tufte E.R. The Visual Display of Quantitative Information. Graphics Press, Cheshire, CT, 1983.

Graph

▶ Chart
▶ Diagram

Graph Data Management in Scientific Applications

Amarnath Gupta
University of California-San Diego, La Jolla, CA, USA

Synonyms

Graph theory; Graph data structure; Graph database

Definition

In mathematics and computer science, graphs are mathematical structures used to model pairwise relations between objects from a certain collection. As a data structure, a "graph" is a set of vertices or "nodes" and a set of edges that connect pairs of vertices.

Historical Background

Graph data management has been studied for nearly two decades. A recent survey [1] states "Graph db-models are applied in areas where information about data interconnectivity or topology is more important, or as important, as the data itself. In these applications, the data and relations among the data, are usually at the same level.... It allows for a more natural modeling of data" and "Queries can refer directly to this graph structure. Associated with graphs are specific graph operations in the query language algebra, such as finding shortest paths, determining certain subgraphs, and so forth." One of the earliest applications of graph data models in science was the study of road networks [9]. While the development of medical terminologies started in the 1980s, the need to study the problem of medical terminology as a graph database came about in the early 1990s. Ellis [4] was one of the early attempts to model nomenclature such as SNOMED and the Unified Medical Language System (UMLS) as graphs.

Foundations

Graphs in Science

Many scientific disciplines deal with data that are structured like graphs. In chemistry, molecules can be considered as graphs whose nodes are atoms and edges are atomic bonds. In ecosystem studies, food webs are graphs where the nodes represent organisms and the edges are directed, representing predatory relationship. In infectious disease epidemiology, the contact network is an undirected graph, where the nodes are people affected by the disease, and an edge is created between pairs of people who came in contact with each other, and possibly caused the infection. In each application the properties of the graph are a little different. The nodes and edges may have additional attributes. The edges may have weights on them. The graph may be acyclic or have cycles. The graph may have a regular structure or may be more random in nature. Despite these difference, in all of these cases, the scientific pursuit involves some kind of network (i.e., graph-based) analysis. In the ecosystem example above, scientists analyze the structure of the graph (e.g., strongly connected components, weighted path lengths) as well as the values of node attributes (e.g., estimates of nutrients) [5].

Graph data management becomes important for science when (i) the problem domain has a very large graph and complex structural properties of this graph need to be extracted or computed by searching for structural patterns over the graph, or (ii) the problem domain has a large collection of graphs, each of which is not necessarily large. The fist situation occurs in the case of large protein networks in biology, where queries may correspond to extracting neighborhoods around nodes that satisfy properties given by the conditions of the query. The second situation occurs in the case of a library of chemical structures, from which one may want to find frequently occurring subgraphs from among a set of graphs that satisfy some query conditions. The data management needs and solutions for these two categories of problems are generally different.

Data Management for Large Graphs

Perhaps the most significant development in graph data management for the large graph case has been done in the context of biological networks, which are graphs where the nodes represent different kinds of biological molecules and edges represent observed or computed relationships between molecule pairs. In some cases, the graphs also include ontologies, which are graphs of standard terms and inter-term relationships. The simplest graph database represents a graph where nodes and edges have labels, and is implemented in a relational database system, and has a node table and an edge table over which graph operations are performed. There are three broad classes of operations that are performed in most common science applications.

1. *Relational Queries on the attributes of Nodes and Edges*: In general, the nodes and edges for the graphs are "thick," that is, they have attributes that are spread into one or more relational tables that can be joined with the basic node and edge tables. A typical query against the graph will have two kinds of conditions – those that are on the connectivity structure of the graph, and those that are standard relational queries on the attributes of the nodes and edges.
2. *Substructure Exploration, Extraction and Transformation*: These operations probe into the connectivity structure of the graph. A basic operation is to extract the k-neighborhood of a node, where the node is selected by some other query condition. Another basic operation is to find the shortest path(s) between two nodes. The query becomes much more complex if all paths between two nodes are desired – for a heavily interconnected graph (i.e., a graph with a small diameter) it may return the entire graph. Even if the graph is not heavily interconnected, the number of paths between two nodes is exponential in the number of edges. In practice, many scientific graph management systems do not allow an all-paths operation. A more complex operation seeks to extract a subgraph that satisfies certain conditions. The PQL system [10] allows queries of the form:

```
SELECT A[−2], A[−*]B
FROM A, B
WHERE  A[−<  5]B  AND  A  ISA
'enzyme' AND
B.name = 'Propane-1,2-diol'
```

   The query returns a 2-neighborhood of a graph around node A and all (denoted by −*) the paths between A and B, where A and B are nodes that satisfy the graph pattern specified by WHERE condition. The WHERE condition looks for all B's having a specific name, and all A's that are within five edges away from the B's, and at the same time have an edge labeled ISA with the term "enzyme." In the industry, the Life Sciences Division of IBM has created a graph extension of DB2, which allows similar operations, and further allows construction and merging of new graphs. Eckman and Brown [3] provides a detailed overview of the system as applied to the life science domain. The graph transformation operation is used in applications like the food web [5], where a graph like a binary food web graph is converted into an aggregate graph called the trophic food web, where nodes having similar properties are collapsed into aggregate nodes and edges between individual nodes are also aggregated into composite edges between the corresponding aggregate nodes.
3. *Computation of Structural Properties*: A different class of operations focuses on computing aggregate graph properties like the number of cliques or strong components in the graph, as well as node properties like betweenness centrality (see http://en.wikipedia.org/wiki/Centrality for a definition and other similar measures). Epidemiological studies like [2] first compute the largest connected component and then determine nodes having the highest centrality values. Applications like molecular structure analysis [6] identify operations like maximal clique operations over Cartesian products of graphs. While graph matching and searching techniques have started using structural properties of graphs, relatively little work has been done in using data management techniques to facilitate these computations.

Data Management for Large Collections of Graphs

In the case where a science application uses a large collection of graphs, most of the data management issues center around finding common substructures, structurally similar graphs and ways to index graph collections to improve retrieval efficiency. A specific class of techniques focuses on mining graph collections to find frequently occurring substructures for chemical data.

Some Data Management Techniques

- A class of techniques called *descriptor spaces* use the idea of keyword-based text retrieval and adapt it for graphs. If the graph is conceived as a document, a keyword is a characteristic substructure of the graph. Chains of a certain length, cycles up to a certain size and number, tree-like branching fragments etc. are commonly used as descriptors. Wale and Karypis [12] computes acyclic portions of the graph as descriptors for chemical compounds. Once these descriptors are collected, graphs are represented in a vector space where the dimensions of the space are formed by these extracted graph fragments. Given a query graph, a suitable similarity function is used to find a ranked list of similar graphs. A common similarity function is the radial basis function (see http://en.wikipedia.org/wiki/Radial_basis_function for a definition).
- A class of techniques has also been developed to rank matching graphs that are returned from any graph query. One basis of ranking graphs is by the frequency of their occurrence in a collection. A different way, often valued more in biological sciences, is by assigning a statistical significance to each returned graph, and ranking results on the significance value. In systems biology, such techniques have been developed for path ranking [8] and graph ranking [11]. However, the algorithms are not very efficient especially for large graphs. A more efficient graph ranking technique based on statistical significance has been developed for the frequent subgraph problem in [7].

Key Applications

Road networks, biological pathways, social networks, transportation and communication networks.

Cross-references

▶ Graph Management in the Life Sciences

Recommended Reading

1. Angles R. and Gutierrez C. Survey of graph database models. ACM Comput. Surv., 40:1–39, 2008.
2. De P., Singh A.E., Wong T., Yacoub W., and Jolly A.M. Sexual network analysis of a gonorrhoea outbreak. Sex. Transm. Infect., 80(4):280–285, 2004.
3. Eckman B.A. and Brown P.G. Graph data management for molecular and cell biology. IBM J. Res. Dev., 50(6):545–560, 2006.
4. Ellis G. Managing large databases of complex medical knowledge using conceptual graphs. In Proc. of the National Health Informatics Conference, 1993, pp. 4–13.
5. Gaedke U. A comparison of whole-community and ecosystem approaches (biomass size distributions, food web analysis, network analysis, simulation models) to study the structure, function and regulation of pelagic food webs. J. Plankton Res., 17(6):1273–1305, 1995.
6. Hattori M., Okuno Y., Goto S., and Kanehisa M. Development of a chemical structure comparison method for integrated analysis of chemical and genomic information in the metabolic pathways. J. Am. Chem. Soc., 125:11853–11865, 2003.
7. He H. and Singh A.K. GraphRank: statistical modeling and mining of significant subgraphs in the feature space. In Proc. 22nd Int. Conf. on Data Mining, 2006, pp. 885–890.
8. Kelley B.P, Yuan B., Lewitter F., Sharan R. Stockwell B.R., and Ideker T. PathBLAST: a tool for alignment of protein interaction networks. Nucleic Acids Res., 32(web server issue):W83–88, 2004.
9. Kunii H.S. DBMS with graph data model for knowledge handling. In Proc. Fall Joint Computer Conf. on Exploring Technology: Today and Tomorrow, 1987, pp. 138–142.
10. Leser U. A query language for biological networks. Bioinformatics, 21(Suppl 2):ii33–ii39, 2005.
11. Sharan R., Suthram S., Kelley R.M., Kuhn T., McCuine S., Uetz P., Sittler T., Karp R.M., and Ideker T. Conserved patterns of protein interaction in multiple species. In Proc. Natl. Acad. Sci. USA, 102:1974–1979, 2005.
12. Wale N. and Karypis G. Acyclic Subgraph-based Descriptor Spaces for Chemical Compound Retrieval and Classification. UMN CSE Technical Report #06–008, 2006.

Graph Data Structure

▶ Graph Data Management in Scientific Applications
▶ Information Integration Techniques for Scientific Data

Graph Database

PETER T. WOOD
Birkbeck, University of London, London, UK

Synonyms

Network database; Link database; Semi-structured database

Definition

A graph database is a database whose data model conforms to some form of graph (or network or link) structure. The graph data model usually consists of nodes (or vertices) and (directed) edges (or arcs or links), where the nodes represent concepts (or objects) and the edges represent relationships (or connections) between these concepts (objects). Therefore the nodes are typically labeled with the names of concepts or objects, while the edges are labeled with types of relationships. More elaborate labeling might involve sets of attribute-value pairs being associated with nodes and/or edges. In addition, more complex structures, such as nested graphs or hypergraphs, may also be permitted. On the other hand, the graph model may be restricted to allow only certain types of graph structures, for example, only acyclic graphs or those that have a distinguished root node.

Similar to relational databases, users can query the graph database with a high-level query language. However, such query languages are usually designed in such a way as to make it easy to ask about paths through the graph (or routes through the network) of unspecified length.

Historical Background

Connections between graph structures and databases existed from the earliest data models developed in the 1960s and 1970s. For example, the hierarchical data model viewed data as comprising tree structures, while the network data model viewed data as consisting of linked structures. The entity-relationship model also defines what is essentially a graph-based structure. However, none of these would really be considered as a graph database.

The notion of a graph database evolved from the recognition that, in a number of applications, a *single* relationship type connecting instances of a single concept may itself form a graph structure. Examples include the link structure in hypertext documents, road connections in geographical databases, and, more recently, various pathways in biological databases. In all of these applications, useful queries need to be able to ask about paths through the database, where the length of the required paths is not necessarily predetermined or known beforehand by the user. Such capabilities were not available in the data models mentioned above, nor in the relational model as originally defined.

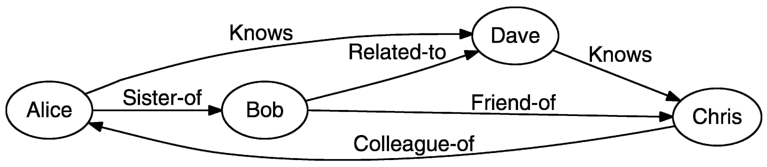
A further requirement of most data models is that data conforms strictly to the structural requirements of the model. Applications in heterogenous data integration and data on the web needed more flexibility, giving rise to the notion of semi-structured data [1] which also conformed to a graph model. Appropriate data models and query languages for graph databases and their applications were investigated mostly in the 1980s and 1990s, with interest having been renewed recently in the context of the semantic web and social networks.

Foundations

A graph database typically comprises a set of graphs. Each graph in the database consists of a set of nodes N and a set of edges E, with each edge in E connecting a pair of nodes in N. The edges may be directed or undirected. In the case of the former, an edge is an ordered pair of nodes; in the case of the latter, it is an unordered pair of nodes. Both nodes and edges may have labels associated with them. This is usually done by two functions, one mapping nodes to labels and the other mapping edges to labels. The labels may be simple values, tuples of values, sets of values, and so on.

A very simple example of a directed graph is shown in Fig. 1. Here the node labels denote names of people and the edge labels denote relationships between people, so this graph could represent a tiny fragment of a simple social network. Note that the graph is *cyclic*; for example, Dave knows Chris who is a colleague of Alice who knows Dave.

Given a graph of data like that in Fig. 1, various queries about the data may be relevant. These might be conventional queries such as to find the pairs of people who know each other (Alice knows Dave, and Dave knows Chris), or to find the people who are sisters of friends of Chris (Alice). However, "schema-level"queries may also be needed, for example to find the relationships in which Alice is involved, or to find in what way(s) Alice and Chris are related. Users might also be interested in nodes in the graph that are connected by paths whose lengths are not known a-priori, such as people connected by any sequence of "knows" relationships or any sequence of either "knows" or "friend-of" relationships. In general, users may want to specify a pattern that should be matched by the sequence of edge labels on paths in which they are interested. One way of specifying such a pattern is by using *regular expressions* over the alphabet of edge labels [9]. So for example, the regular expression (know|friend-of)$^+$ specifies sequences of one or more occurrences of edges labeled with either "knows"



Graph Database. Figure 1. A simple example of a graph.

or "friend-of." The use of regular expressions for querying path information has been studied extensively in the area of semi-structured data [1]. XPath, a language for selecting nodes from XML documents, provides a limited form of path pattern specification for querying XML documents, which are essentially trees rather than graphs.

Depending of the application area for which a graph database is being used, there are many other forms of query that may be useful. Examples include shortest path queries (e.g., find the shortest or quickest route from one node to another), subgraph isomorphism queries (the query is viewed as a subgraph whose matches against a graph in the database are to be found), approximate graph matching (where the query may not match a subgraph exactly), finding complete subgraphs (where every node is connected to every other node), queries involving counting the indegree and outdegree of nodes (the number of incoming and outgoing edges, respectively, perhaps to see how "well-connected" a node is), finding the largest common subgraph among a number of graphs, or locating the least common ancestor of two or more nodes (usually in trees).

Unlike in a conventional database, the data in Fig. 1 does not have an explicit *schema* describing it. This lack of explicit *schema* is one of the characteristics of what has been termed semi-structured data [1]. In fact, some schema information is given by the edge labels, so the data are in some sense *self-describing*. However, there is no requirement that the data conform to a set of constraints as usually defined in a schema. Such constraints might assign types to nodes, state what edge labels are permitted, and limit the types of nodes that are permitted to participate in each relationship. Another graph-based model that does not require a schema is RDF (Resource Description Framework), one of the languages proposed for the semantic web. Nevertheless, researchers have defined various forms of schemas for graph databases [3], and RDF does have an associated schema language, although this is used for inference of derived information rather than for validation of RDF graphs.

As in conventional databases, one of the principal concerns in a graph database system is the efficient evaluation of queries. This gives rise to the study of query optimization [5,6] and appropriate index structures for graph data [10,15]. Both of these topics have been studied more extensively in the context of tree-structured XML data.

Some applications require graph structures that are more elaborate than the simple version of nodes and edges described above. For example, hypergraphs, where edges comprise sets of nodes rather than pairs of nodes, have been used to model hypertext [14]. Alternatively, the hypernode model [12] considers nodes that can themselves comprise graphs, thereby providing a useful abstraction mechanism. A different approach to abstraction is provided by so-called blobs that can appear in the hygraphs of the Hy^+ system [4]. These blobs comprise sets of nodes and share some similarities with the blobs defined in higraphs [8]. Graphs have also been used as models for object-oriented databases [7].

Key Applications

There are a large number of potential application areas for graph databases. These include hypertext [14] and the Web, discovery of semantic associations in national security (or criminal investigation) applications [13] (also called link analysis), chemical structure modeling, geographic information systems, bibliographic citation analysis, taxonomy and partonomy representation, data provenance graphs, and the semantic web, e.g., RDF [2]. In biology alone [11], there are numerous applications including metabolic pathways, signaling pathways, gene regulatory networks, gene clusterings, and protein interaction networks.

Cross-references

▶ Biological Networks
▶ Data Integration

▶ Graph Management in the Life Sciences
▶ Graph
▶ Graph Data Management in Scientific Applications
▶ Indexing Semi-Structured Data
▶ Information Integration
▶ Information Integration Techniques for Scientific Data
▶ Network Data Model
▶ Object Data Models
▶ Semi-Structured Data Model
▶ Social Networks
▶ Resource Description Framework
▶ Path Query
▶ Query Language

Recommended Reading

1. Abiteboul S., Buneman P., and Suciu D. Data on the Web: From Relations to Semi-structured Data and XML. Morgan Kaufmann, San Francisco, CA, 2000.
2. Angles R. and Gutierrez C. Querying RDF data from a graph database perspective. In Proc. 2nd European Semantic Web Conference, 2005, pp. 346–360.
3. Buneman P., Davidson S., Fernandez M., and Suciu D. Adding structure to unstructured data. In Proc. 6th Int. Conf. on Database Theory, 1997, pp. 336–350.
4. Consens M.P. and Mendelzon A.O. Hy$^+$: a hygraph-based query and visualization system. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1993, pp. 511–516.
5. Fernandez M. and Suciu D. Optimizing regular path expressions using graph schemas. In Proc. 14th Int. Conf. on Data Engineering, 1998, pp. 14–23.
6. Goldman R. and Widom J. DataGuides: enabling query formulation and optimization in semi-structured databases. In Proc. 23th Int. Conf. on Very Large Data Bases, 1997, pp. 436–445.
7. Gyssens M., Paredaens J., den Bussche J.V., and Gucht D.V. A graph-oriented object database model. IEEE Trans. Knowl. Data Eng., 6(4):572–586, August 1994.
8. Harel D. On visual formalisms. Commun. ACM, 31(5):514–530, May 1988.
9. Mendelzon A.O. and Wood P.T. Finding regular simple paths in graph databases. SIAM J. Comput., 24(6):1235–1258, December 1995.
10. Milo T. and Suciu D. Index structures for path expressions. In Proc. 7th Int. Conf. on Database Theory, 1999, pp. 277–295.
11. Olken F. Graph data management for molecular biology. OMICS: A Journal of Integrative Biology, 7(1):75–78, 2003.
12. Poulovassilis A. and Levene M. A nested-graph model for the representation and manipulation of complex objects. ACM Trans. Inf. Syst., 12(1):35–68, 1994.
13. Sheth A., Aleman-Meza1 B., Arpinar I.B., Bertram C., Warke Y., Ramakrishnan C., Halaschek C., Anyanwu K., Avant D., Arpinar F.S., and Kochut K. Semantic association identification and knowledge discovery for national security applications. J. Database Manag., 16(1):33–53, 2005.
14. Tompa F. W. A data model for flexible hypertext database systems. ACM Trans. Database Syst., 7(1):85–100, 1989.
15. Yan X., Yu P.S., and Han J. Graph indexing: a frequent structure-based approach. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2004, pp. 335–346.

Graph Database Mining

▶ Frequent Graph Patterns

Graph Drawing

▶ Visualizing Network Data

Graph Embedding

▶ Dimension Reduction Techniques for Clustering

Graph Layout

▶ Visualizing Network Data

Graph Management in the Life Sciences

Ulf Leser, Silke Trißl
Humboldt University of Berlin, Berlin, Germany

Definition

Graphs play an increasingly important role in many research areas in the life sciences. Especially in systems biology, graphs are used to model the complex temporal and spatial relationships between entities within an organism. For example, graphs are used to model signaling pathways, where nodes are proteins and edges represent the flow of information between proteins. The flow represents physical modifications of the participating proteins, such as the addition or removal of certain chemical groups. Since proteins are often involved in various signaling pathways, one can model the complete signaling management inside a cell as a graph consisting of tens of thousands of nodes and many more edges. However, graphs are also used in less obvious areas. Biological *ontologies* are cycle-free graphs of biological concepts connected by specialization relationships; they are called thesauri in information retrieval. Phylogenetic networks are formed by species and their evolutionary relationships.

Biological graphs are the subject of various types of analyses. *Graph querying* searches for all occurrences of a given (sub-) graph pattern inside a large graph, for instance to find common motifs of gene regulation. *Graph mining* aims at detecting significant patterns inside one or more graphs, for instance finding all graph patterns that are more frequent than expected by chance. *Graph clustering* tries to find dense subgraphs, which potentially can be associated with stable multi-protein complexes.

Formally, a graph G is composed of nodes V and edges E with $E \subseteq V \times V$. Nodes usually represent biological or chemical entities, and edges represent specific types of relationships between those entities. Depending on the application, edges may be directed or undirected, labeled or unlabeled, and weighted or unweighted.

Historical Background

For a long time, graphs have been used in the life sciences only for describing the two- or three-dimensional structure of complex molecules, especially of proteins. In such graphs, nodes represent atoms and edges represent bonds between atoms. However, with the recent massive increase in data production on all levels (including DNA sequences, protein structures, gene expression levels etc.), many more applications of graphs have been suggested. The most prominent types of graphs that emerged are: (i) networks of interacting proteins, (ii) networks of gene regulation, (iii) networks of signal transduction, and (iv) metabolic networks.

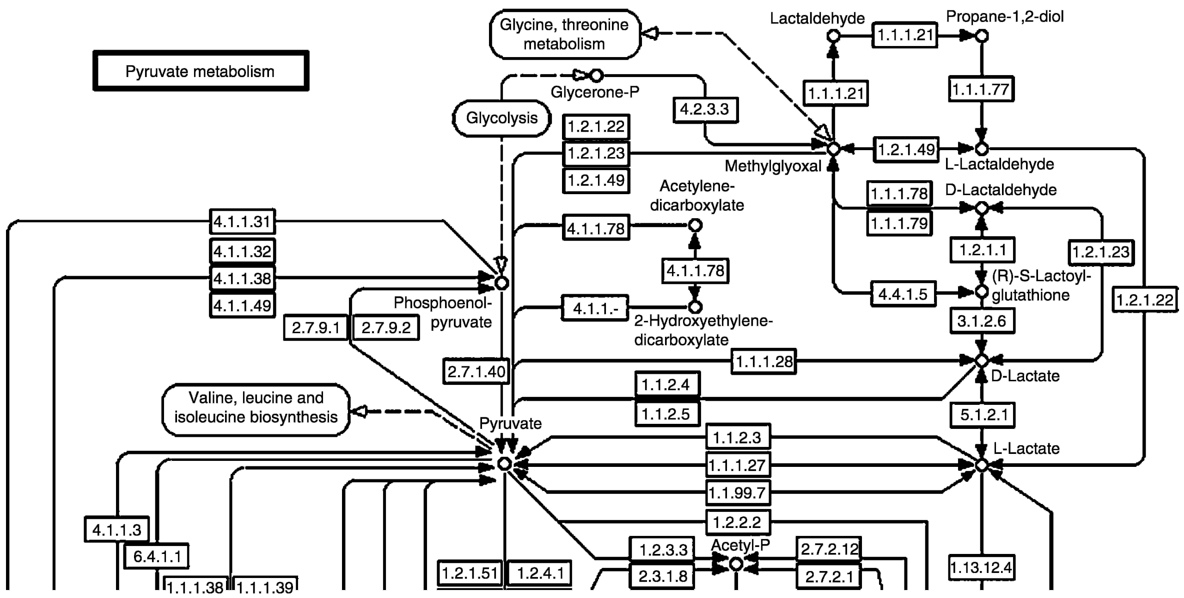
The study of such networks has flourished with the advent of high-throughput techniques to detect the different "types" of edges. The most important ones are the Yeast 2-Hybrid technique for detecting physical interactions between proteins and the co-immunoprecipitation technique for detecting regulation events. The study of the complex interplay of all components inside a cell and across cell boundaries is one of the main goals of modern systems biology. Today, the largest graphs available based on experimental evidence contain in the order of ~15,000 proteins and ~100,000 edges. When predicted information is taken into account, the resulting graphs are much larger, with up to one million nodes and five million edges.

Systems for storing biological networks and their experimental evidence were first developed around 2000, with the notable exception of the KEGG database (first published in 1997) and the EcoCyc system (first published in 1994). With more and more networks being available, various approaches emerged to analyze the information contained in those graphs, for instance to predict further edges based on graphs from evolutionary closely related species, or to predict the function of a protein from its neighbors in a protein-protein interaction network. Only very recently, there have been proposals for developing specific data management solutions that take into account the intrinsic nature of biological graphs.

Foundations

Types and Models of Graphs

The most prominent types of graphs in the life sciences are (i) Networks of physically interacting proteins: Here, nodes represent proteins and edges represent a physical interaction between two proteins. Edges are directed or undirected and may be labeled with the type of the interaction and the evidence for the biological truth of the interaction; (ii) Networks of gene regulation: Nodes represent genes or transcription factors (TF), and edges represent the influence of transcription factors on gene expression. Since transcription factors are also produced by gene expression, TFs also influence the regulation of other TFs. Edges are directed and labeled to reflect a positive or a negative influence; (iii) Signal transduction: Nodes represent proteins or other molecules and edges model the flow of information. Edges are directed and may be labeled; (iv) Metabolic networks (see Fig. 1): Nodes represent arbitrary molecules and edges represent a chemical reaction between the connected molecules catalyzed by enzymes. Edges are labeled and directed



Graph Management in the Life Sciences. Figure 1. Fraction of the Pyruvate Metabolism as represented in the KEGG Database.

or, as many reactions are reversible, bidirectional. In any real system, nodes usually have a wealth of information attached, such name(s), sequences, structures, functions, or experimental values, e.g., values from microarray experiments.

Properties of Biological Networks

A human cell is estimated to contain approximately 20,000 genes that are translated in up to 150,000 different proteins. The same number can also be estimated for other mammals, such as mouse, rat, or chimp. Metabolic networks currently contain up to 20,000 nodes and 40,000 edges, depending on the organism (for bacteria, the networks are much smaller). In contrast, cross-species protein-protein interaction networks currently contain up to 1,000,000 nodes and, depending on the method used to infer the interactions, many more edges.

Biological networks exhibit a specific structure, which is shared by other complex systems, such as the Internet or social networks. Biological networks are scale-free, i.e., the degree distribution follows a power-law. In *scale-free graphs* many nodes have only few edges. These nodes are connected by some nodes that have many edges, called hubs. Scale-free networks can be grown in computer simulations using the so-called preferential attachment, which reflects the evolution of existing biological networks. Preferential attachment means that graphs are grown incrementally. One starts with a small number of nodes and incrementally adds new nodes, which are attached to existing ones, preferably to nodes that are already well connected [1].

Graph Databases

Graph databases in the life sciences can be divided into different categories. First, there are databases about metabolic pathways. The best known sources are KEGG, aMAZE, Reactome, and BioCyc, which contain information added manually by biological experts. For example the reference pathway in KEGG currently (as of 2007) contains 20,000 nodes and 40,000 edges. Secondly, there exist various protein-protein interaction databases, e.g., DIP, MINT, STRING, or PubGene. While the first two databases only accept entries for which strong evidence has been published, STRING also accept entries from less reliable high-throughput experiments. STRING and PubGene also automatically mine the scientific literature for interactions and include those into the database. Therefore the size of the data sets as well as the data quality varies to a great degree.

To exchange data between different databases several standards were established. The "Systems Biology Markup Language" (SBML) and the "BioPAX Data Exchange format" were both designed for the exchange of data in systems biology and especially for biological networks. The "Proteomics Standards Initiative Molecular Interaction XML format" (PSI MI) targets only protein-protein interaction data. For a comparison of the different format standards see [11].

Graph Query Languages

Query languages for graphs are based on the description of properties of subgraphs. Evaluating a graph query on a graph means to find all subgraphs having the specified properties. Numerous different properties may be used, which require the existence of nodes with certain attributes, edges with specific labels, or the existence of paths of various shapes connecting nodes.

The specific difficulty in developing graph query languages is the required support for path queries, i.e., queries over sequences of connected nodes. An exemplary query posed on a biological network might be "Give me all enzymes that affect a compound called glucose in less than five steps and which are also associated with the enzyme 1.6.3.1 in an arbitrary number of steps." Such queries can be solved by either traversing the graph at query time, or by using special graph indexing structures, such as the 2-Hop-Cover. In general, indexing graphs is much harder than indexing trees. Even when graph queries contain no transitive predicates their evaluation is equivalent to solving the subgraph isomorphism problem, which is NP-complete. However, as nodes in biological graphs usually have unique labels, evaluation of queries in real applications is often much less complex.

Examples of graph query languages in biology are the pathway query language [5] and the language described in [2]. Both approaches are implemented on top of a relational database management system.

Graph Mining

The need to analyze graph data has lead to the development of various tools and algorithms that solve specific problems on graphs, of which three are particularly important. (i) Finding cliques or densely connected clusters (quasi-cliques) in a graph of protein-protein interactions is important to identify molecular complexes, which are groups of proteins that together perform a biological function. Jeong and colleagues present different algorithms, for instance based on supermagnetic clustering or a Monto Carlo optimization algorithm [10]; (ii) Subgraphs that are present in the PPI networks of various species (conserved subgraphs) are important evidence towards the evolutionary importance of subnetworks. Algorithms either find perfectly conserved subgraphs or allow insertions and deletions of nodes or edges. One approach based on path alignment is presented in [8]; (iii) Frequent subgraphs in biological networks are believed to hint "building blocks" of the molecular machinery of a cell. A subgraph of a graph is called frequent when it appears more often than expected. An efficient algorithm based on random sampling of isomorphic subgraphs is presented in [14].

Key Applications

Graph-Based Function Prediction

The computer-driven prediction of the biological function of a given protein is a long-standing goal of bioinformatics research, which, until recently, was mostly based on finding similarities between the sequences of different genes or proteins. The possibility to analyze the proteins with which a protein interacts in a cell has added a powerful new source of evidence for function prediction. It has, for instance, been shown that proteins in densely connected regions of a protein-protein-interaction network very often share the same biological function. Furthermore, the fact that two proteins interact with the same other proteins increases the likelihood that these two proteins share a biological function. This interaction can happen within a species, or can be transferred from orthologous (i.e., highly similar) proteins in other species. A survey on those methods has appeared in [8].

Managing Biological Ontologies

Biological ontologies are structured sets of concepts. Usually, concepts have a name, a textual description, and are related to each other through ISA (generalization/specialization) or PART-OF relationships. Prominent examples are the Gene Ontology (for describing the function and location of genes) and the Mammalian Phenotype Ontology (for describing phenotypes of mammalians). Naturally, an ontology may be seen as a graph, where the concepts are nodes and the relationships are edges. Ontologies form trees or directed acyclic graphs. Therefore, many approaches to the management, analysis, and visualization of ontologies use a graph-based framework. For instance, the semantic similarity between two concepts is often computed by finding the least common ancestor of the two concepts in the ontology [6]. In the same spirit, the semantic similarity between two genes based on all their assigned concepts may be computed by comparing the two subgraphs formed by the concepts of the genes and their ancestors in the ontology. Such techniques are often used to score the functional coherence of groups of genes, for instance, whose gene expression react in a similar way under external stimulus.

Management of Phylogenetic Trees

Phylogenetic trees are used to describe the relationship between different species. The trees may be rooted or not and usually are unweighted. Leaf nodes represent different species and inner nodes stand for common, often unknown ancestors. Phylogenetic trees are constructed using one or more common features from the species, usually a gene whose genetic sequence varies slightly in different species. Since there are various algorithms for constructing phylogenetic trees using different features, very often contradicting trees emerge. Recently, tree banks have been proposed as special applications for managing and searching phylogenetic trees. A particular problem is to find for a given query tree those trees in a database of phylogenetic trees that are most similar to the query. This requires a tree distance measure and a search procedure (see for instance, [13]).

It is more and more acknowledged that trees are inappropriate for modeling certain effects in evolution, such as plant hybridization or horizontal gene transfer. This results in efforts to develop algorithms for the inference of phylogenetic networks, i.e., graphs where one species can have two or more ancestors. An interesting method based on a Maximum Parsymony approach to network reconstruction is presented in [3].

Visualization of Biological Networks

Visualization tools for biological networks must address several issues. First of all, they must be able to display the graph using suitable layout algorithms. In addition, these tools must provide visualization of attached information, e.g., name of a node, its cellular location, or results of high-throughput experiments. Several tools were specifically developed for this purpose. A well known tool is Cytoscape, a Java-based open source tool that can display large networks and can be extended using plug-ins. Further tools are Pathway Tools, Pathway Studio, or Ospey, which both also allow various kinds of network analysis. For a review on visualization tools for biological networks see [12].

Future Directions

Systems biology will be a major research area in the life sciences in the coming years. Since it focuses on the interplay of chemical entities in cells, it builds on graphs in a very fundamental manner. Therefore, the importance of graphs in the life sciences is expected to grow further in the near future. This will also bring a steep increase in the size of graphs under study, as more and more experimental data become available. Since, for many applications also the inclusion of predicted edges perfectly makes sense, already now graphs may contain up to several hundreds of thousands of edges. It is therefore foreseeable that studies that address interaction data from many species will have to cope with millions of objects in the near future.

Technically, relational databases nowadays build the backbone of most graph management systems. However, a graph may also be modeled using the *Resource Description Framework* (RDF), an underlying technology of the *Semantic Web*. Therefore, query languages for RDF might be a natural choice for also analyzing biological graphs. However, the current proposals lack the advanced graph search features required in the life sciences. The development of new graph query languages, possibly including certain types of simple graph analysis predicates, will remain an important topic.

URL to Code

A few pathway databases

- "Pathguide: the pathway resource list": http://www.pathguide.org/
- "KEGG: Kyoto Encyclopedia of Genes and Genomes": http://www.genome.ad.jp/kegg/
- "Reactome": http://www.reactome.org/
- "BioCyc Home": http://biocyc.org/
- "DIP:Home": http://dip.doe-mbi.ucla.edu/
- "MINT database": http://mint.bio.uniroma2.it/mint/Welcome.do
- "STRING; functional protein association networks": http://string.embl.de/

A few visualization tools

- "Cytoscape: Analyzing and Visualizing Network Data": http://www.cytoscape.org/
- "Ariadne Genomics: Pathway Studio": http://www.ariadnegenomics.com/products/pathway-studio/
- "OSPREY: Network Visualization System": http://biodata.mshri.on.ca/osprey/servlet/Index

A few pathway management systems

- "PathCase: Metabolic Pathways Database System": http://nashua.cwru.edu/pathways
- "Pathway Tools Information Site": http://brg.ai.sri.com/ptools/
- "PatikaWeb": http://www.cs.bilkent.edu.tr/~patikaweb/

Cross-references

▶ Biomedical Data/content Acquisition, Curation
▶ Biological Networks
▶ Frequent Graph patterns
▶ Grpah Database
▶ Graph Data Management in Scientific Applications
▶ Ontologies and Life Science Data Management
▶ Query Languages for the life Sciences
▶ Tree-based Indexing

Recommended Reading

1. Barabàsi A.-L. and Oltvai Z.N. Network biology: understanding the cell's functional organization. Nat. Rev. Genet., 5:101–113, 2004.
2. Eckman B.A. and Brown P.G. Graph data management for molecular and cell biology. IBM J. Res and Dev., 50:545–560, 2006.
3. Jin G., Nakhleh L., Snir S. and Tuller T. Efficient parsimony-based methods for phylogenetic network reconstruction. Bioinformatics, 23:123–2128, 2006.
4. Karp P.D., Paley S., and Romero P. The pathway tools software. Bioinformatics, 18(Suppl. 1):S225–S232, 2002.
5. Leser U. A query language for biological networks. Bioinformatics, 21(Suppl. 2):ii33–ii39, 2005.
6. Lord P.W., Stevens R.D., Brass A., and Goble C.A. Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation. Bioinformatics, 19(10):1275–1283, 2003.
7. Schaefer C.F. Pathway databases. Ann NY Acad Sci., 1020:77 –91, 2004.
8. Sharan R. and Ideker T. Modelling cellular machinery through biological network comparison. Nat. Biotechnol. 24(4):427–433, 2006.
9. Sharan R., Suthram S., Kelley R.M., Kuhn T., McCuine S., Uetz P., Sittler T., Karp R.M., and Ideker T. Conserved patterns of protein interaction in multiple species. In Proc. Natl. Acad. Sci. USA., 102(6):1974–1979, 2005.
10. Spirin V. and Mirny L.A. Protein complexes and functional modules in molecular networks. In Proc. Natl. Acad. Sci. USA., 100(21):12123–12128, 2003.
11. Strömbäck L. and Lambrix P. Representations of molecular pathways: an evaluation of SBML, PSI MI and BioPAX. Bioinformatics, 21(24):4401–4407, 2005.
12. Suderman M. and Hallett M. Tools for visually exploring biological networks. Bioinformatics, 23(20):2651–2659, 2007.
13. Wang J.T.L., Shan H., Shasha D., and Piel W.H. Fast structural search in phylogenetic databases. Evol. Bioinform. Online, 1:37–46, 2005.
14. Wernicke S. Efficient detection of network motifs. IEEE/ACM Trans. Comput. Biol. Bioinformatics., 3(4):347–359, 2006.

Graph Mining on Streams

Andrew McGregor
Microsoft Research, Silicon Valley, Mountain View, CA, USA

Synonyms

Graph streams; Semi-streaming model

Definition

Consider a data stream $A = \langle a_1, a_2, \ldots, a_m \rangle$ where each data item $a_k \in [n] \times [n]$. Such a stream naturally defines an undirected, unweighted graph $G = (V, E)$ where

$$V = \{v_1, \ldots, v_n\} and$$
$$E = \{(v_i, v_j) : a_k = (i, j) \text{ for some } k \in [m]\}.$$

Graph mining on streams is concerned with estimating properties of G, or finding patterns within G, given the usual constraints of the data-stream model, i.e., sequential access to A and limited memory. However, there are the following common variants.

Multi-Pass Models: It is common in graph mining to consider algorithms that may take more than one pass over the stream. There has also been work in the *W-Stream* model in which the algorithm is allowed to write to the stream during each pass [9]. These *annotations* can then be utilized by the algorithm during successive passes and it can be shown that this gives sufficient power to the model for PRAM algorithms to be simulated [8]. The *Stream-Sort* model goes one step further and allows *sorting passes* in which the data stream is sorted according to a key encoded by the annotations [1].

Weighted, Dynamic, or Directed Graphs: For many problems it is implicitly assumed that the elements a_k are distinct. When the data items are not distinct, the stream naturally defines a multi-graph, i.e., an edge (v_i, v_j) has multiplicity equal to $|\{k : a_k = (i, j)\}|$. In such a model it would be natural to consider the deletion of edges but this has not been explored thoroughly. Finally, the definition can be generalized to define weighted graphs where a third component of the data item, $a_k \in [n] \times [n] \times \mathbb{R}^+$, would indicate a weight $w_{(u,v)}$ associated with the edge (u,v), or directed graphs.

Adjacency and Incidence Orderings: In some applications it is reasonable to assume that all edges incident on the same node arrive consecutively. This assumption defines the *incidence model*. In the *adjacency model* no such assumption is made.

Historical Background

Graph problems were considered by Henzinger et al. [15] in one of the earliest papers on the data-stream model. Their work considered a variety of problems including pointer jumping problems based on the degrees of various nodes in a directed layered graph. Unfortunately, many of the results showed that a large amount of space is required for these types of problems. Subsequent early work considered counting the number of triangles in a graph [2] and estimating common neighborhoods [3]. Again, a large component of these results were negative. It seemed that more "complicated" computation was not possible in this model using only a small amount of space.

It seems that most graph algorithms need to access the data in a very adaptive fashion. Since the entire graph can not be stored, emulating a traditional algorithm may necessitate an excessive number of passes over the data. This has motivated various specific stream models tailored to processing graphs, including the *Semi-Streaming*, *W-Stream*, and *Sort-Stream* models. The semi-streaming model is characterized by an $O(n$ polylog $n)$ space restriction, i.e., space proportional to the number of nodes rather than the number of edges. For dense graphs this represents considerably less space than that required to store the entire graph. This restriction was identified as an apparent "sweet-spot" for graph streaming in a survey article by Muthukrishnan [17] and was first explored by Feigenbaum et al. [13]. The W-Stream and Stream-Sort models, described earlier, were introduced by Demetrescu et al. [9] and Aggarwal et al. [1] respectively.

Foundations

This section discusses the main upper and lower bounds known for solving graph problems in the data-stream model. The focus is on results for the standard data-stream model (including the Semi-Streaming space restriction) but note that some of the below problems, including testing connectivity, computing shortest paths, and constructing minimum spanning trees, have been considered in the W-Stream and Stream-Sort models [1,9].

Connectivity and Minimum Spanning Trees: Determining whether a graph is connected is a fundamental problem. It can be solved in the semi-streaming model and it can be shown that any one-pass algorithm requires $\Omega(n)$ space [15]. More generally, connectivity is a *balanced property* where a graph property $\mathcal{P}$ is *balanced* if there exists a constant $c > 0$ such that for all sufficiently large n, there exists a graph $G = (V, E)$ with $|V| = n$ and $u \in V$ such that:

$$\min\{|\{v : (V, E \cup \{(u, v)\}) \text{ has } \mathcal{P}\}|,\\ |\{v : (V, E \cup \{(u, v)\}) \text{ has} \neg\mathcal{P}\}|\} \geq cn.$$

It was shown that all balanced properties require $\Omega(n)$ space [12]. This was part of the justification for the semi-streaming space restriction. However, many problems become feasible with $O(n$ polylog $n)$ space such as testing planarity and determining whether a graph is bipartite. Testing higher degrees of vertex-connectivity has also been considered in the semi-streaming model. The below table summarizes the start of the art results for determining if a graph is k-connected, i.e., whether k vertices need to be removed in order to disconnect the graph:

k	Passes	Time (per-edge)	Ref.
1,2,3	1	$O(\alpha(n))$	[12]
4	1	$O(\log n)$	[12]
k	1	$O(k^2 n)$	[18]
k	$k+1$	$O(k + \alpha(n))$	[18]

where $\alpha(n)$ is the inverse Ackermann function. Lower bounds have also been investigated for k-edge and k-vertex connectivity [15]. Other related results include a semi-streaming, one-pass algorithm for constructing the minimum spanning tree with $O(\log n)$ processing-time per edge.

Distances and Spanners: An undirected graph $G = (V, E)$ naturally defines a distance function $d_G : V \times V \to \mathbb{R}$ where $d_G(u, v)$ is the length of the shortest path in G between u and v. The diameter of G is the length of the longest shortest path, i.e.,

$$\mathrm{Diam}(G) = \max_{u,v\in V} d_G(u,v),$$

and the girth of G is the length of the shortest cycle in G, i.e.,

$$\mathrm{Girth}(G) = \min_{(u,v)\in E}[w_{(u,v)} + d_{G\backslash(u,v)}(u,v)].$$

To date, most of the algorithms for approximating quantities related to distance are based on constructing sparse *spanners*, where a subgraph $H = (V, E')$ is an (α, β)-spanner of $G = (V, E)$ if, for any vertices $x, y \in V$,

$$d_G(x,y) \le d_H(x,y) \le \alpha \cdot d_G(x,y) + \beta.$$

Note that constructing an $(\alpha, 0)$-spanner H for an unweighted graph also gives an indication of the girth of the original graph. In particular, if $H \neq G$ then $\mathrm{Girth}(G) \le \alpha + 1$ because there exists $(u, v) \in E(G)\backslash E(H)$ and this must satisfy $d_{G\backslash(u,v)}(u,v) \le \alpha$ if H is an $(\alpha, 0)$-spanner.

The first spanner constructions in the data-stream model were presented in [12,13]. The state of the art construction is due to Elkin [10] who presented a randomized, single-pass algorithm that constructs a $(2t - 1,0)$-spanner for an unweighted graph in $O((t \log n)^{1-1/t} n^{1+1/t})$ space (with probability $1 - 1/n^{\Omega(1)}$). The algorithm processes each edge (u, v) in $O(1)$ expected time and $O(\log d_{u,v}/\log\log d_{u,v})$ worst-case time where $d_{u,v}$ is the sum of the degrees of u and v. The algorithm can be generalized to weighted graphs by rounding edge weights to powers of $(1 + \varepsilon)$, constructing spanners for each edge set with the same weight, and taking the union of these spanners. This adds a factor of $O(\varepsilon^{-1}\log \omega)$ (where ω is the ratio between the biggest and smallest weights) in the space and time complexity and adds a factor of $(1 + \varepsilon)$ in the approximation factor of a distance. Elkin and Zhang [11] present various algorithms for constructing (α, β)-spanners.

If only a specific distance, $d_G(u, v)$, needs to be approximated then it may appear that constructing a spanner for the entire graph is excessive. However, this is not the case. In particular, it can be shown that any single-pass algorithm that approximates the (weighted) graph distance between two given nodes up to a factor t with probability at least 3/4 requires $\Omega(n^{1+1/t})$ space [13]. Furthermore, this bound also applies even with the promise $d_G(u,v) = \mathrm{Diam}(G)$. Consequently approximation via spanners is at most a factor 2 from optimal.

Counting Triangles: As was noted earlier, approximating the number of triangles in an undirected graph was one of the earliest problems considered in the data-stream model [2]. The number of triangles is related to the clustering and transitivity coefficients of the graph. The state of the art [4] are one-pass, randomized algorithms that estimate the number of triangles up to a multiplicative factor or $(1 + \varepsilon)$ with probability at least $1 - \delta$ using space:

Model	Space
Adjacency	$O(\varepsilon^{-2}(1 + T_1/T_3 + T_2/T_3)\log\delta^{-1})$
Incidence	$O(\varepsilon^{-2}(1 + T_2/T_3)\log\delta^{-1}\log n)$

where T_i is the number of node triples upon which the induced sub-graph has exactly i edges. While both of these space bounds can be large (e.g., for dense graphs with few triangles) this compares favorably to the

$$O(\epsilon^{-2}(1 + T_0/T_3 + T_1/T_3 + T_2/T_3)\log\delta^{-1})$$

space required by the naive algorithm that randomly samples node-triples and computes the fraction that are triangles. It can also be shown that any p-pass algorithm determining if the number of triangles is non-zero requires $\Omega(p^{-1}n^2)$ space [2]. There has also been work on estimating the count of larger cycles or cliques. However, it can be shown using results in extremal graph theory and a reduction from the set-disjointness communication problem that any p-pass algorithm that determines if $\mathrm{Girth}(G) \ge g$ requires $\Omega(p^{-1}(n/g)^{1+4/(3g-4)})$ space [13].

A related problem is that of estimating *path aggregates*. Define P_k to be the number of pairs of vertices that are joined by a simple path of length k. For a graph with r connected components, it is possible to approximate P_2 in one pass and $O(\varepsilon^{-2}m(m - r)^{-1/4}\log\delta^{-1})$ space even if edges may be deleted [14]. A space lower bound of $\Omega(\sqrt{m})$ is also known.

BFS trees: A common subroutine in many traditional graph algorithms is that of constructing a breadth-first-search (BFS) tree. Unfortunately it can be shown that computing the first l layers of a breadth-first-tree from a prescribed node requires either $\lceil(l - 1)/2\rceil$ passes or $\Omega(n^{1+1/l})$ of space [13].

Matchings: Given a graph $G = (V, E)$, the *Maximum Cardinality Matching* (MCM) problem is to find the largest set of edges such that no two adjacent edges are selected. More generally, for an edge-weighted graph, the *Maximum Weighted Matching* (MWM) problem is to find the set of edges whose total weight is maximized subject to the condition that no two adjacent edges are selected. The following semi-streaming algorithms are known for these problems:

Weighted	Passes	Approximation Ratio	Ref.
No	1	0.5	[13]
No	$O_\varepsilon(1)$	$1 - \varepsilon$	[16]
Yes	1	0.179	[19]
Yes	$O_\varepsilon(1)$	$0.5 - \varepsilon$	[16]

The first of the above algorithms is based on a simple greedy approach, i.e., the algorithm always maintains a matching and adds the latest edge if it is not adjacent to a currently stored edge. The third and fourth algorithms are variants on this basic idea. The second algorithm is based on finding augmenting paths for a currently stored matching, i.e., odd length paths with every second edge in the currently stored matching. For each augmenting path found it is possible to increase the size of the currently stored matching by one. It can be shown that if there are only a relatively small number of $O(\varepsilon^{-1})$-length augmenting paths then the current matching is already a good approximation. Alternatively, if there are many short augmenting paths then it is possible to find a constant fraction using a randomized approach.

Degree Distributions and Random Walks: Given a stream of edges with possible duplicates, a natural question that arises is to estimate properties of the underlying graph. Work has been done on estimating the frequency moments, heavy hitter, and range sums of the degrees of this underlying graph [6]. For example, if d_v is the degree of v in the underlying graph then it is possible to approximate $M_2 := \sum_v d_v^2$ in one pass and $O(\epsilon^{-4}\sqrt{n}\log n)$ space. Note that many of these problem are solvable using standard techniques if there are no duplicates in the stream of edges. A related problem is to estimate the entropy of a random walk on an undirected, unweighted graph. Here the graph stream is an observation of a random walk whose states are nodes of the graph. There exists a single-pass $O(\varepsilon^{-4} \log^2 n \log^2 \delta^{-1})$ space algorithm that estimates the entropy of the walk [5]. These algorithms use a combination of algorithms for counting distinct items and the AMS sampling technique. The problem of actually constructing random walks has also been considered [7]. This has applications to estimating the page-rank vector, mixing time and conductance of graphs.

Key Applications

Massive graphs arise naturally in many real world scenarios. Two examples are the *call-graph* and the *web-graph*. In the call-graph, nodes represent telephone numbers and edges correspond to calls placed during some time interval. In the web-graph, nodes represent web pages, and the edges correspond to hyper-links between pages. When processing these graphs it is often appropriate to use the data-stream model. For example, the graph may be revealed by a web-crawler or the graph may be stored on external memory devices and being able to process the edges in an arbitrary order improves I/O efficiency. One of the major drawbacks of traditional graph algorithms, when applied to massive graphs such as the web, is their need to have random access to the edge set. Massive graphs also arise in structured data mining, where the relationships among the data items in the data set are represented as graphs, and social networks.

Future Directions

There are numerous specific open problems that arise from the existing work on graph streams. For example, one could attempt to improve the approximation factors of the known approximate matching algorithms or the space required to approximate M_2. Another idea is to explore distance approximation in multiple passes. While computing exact distance may require many passes this does not preclude the possibility of better approximation with a few additional passes.

Other more general ideas include investigating graph problems in the *probabilistic data-stream model* or the *random-order data-stream model*. In the probabilistic data-stream model, a probability p_e would be assigned to each edge e. The goal of an algorithm would be to estimate the probability that the random graph generated has a certain property given that each edge is present independently with probability p_e. In the random-order data-stream model the assumption, as the name suggests, is that the edges arrive in random order. The goal is to design algorithms that estimate some property with high probability where the probability is defined over both the coin flips of the algorithm and the ordering of the stream.

Cross-references

▶ One-Pass Algorithm
▶ Stream Models
▶ Synopsis Structure

Recommended Reading

1. Aggarwal G., Datar M., Rajagopalan S., and Ruhl M. On the streaming model augmented with a sorting primitive. IEEE Symposium on Foundations of Computer Science, 2004, pp. 540–549.
2. Bar-Yossef Z., Kumar R., and Sivakumar D. Reductions in streaming algorithms, with an application to counting triangles in graphs. In ACM-SIAM Symp. on Discrete Algorithms, 2002, pp. 623–632.
3. Buchsbaum A.L., Giancarlo R., and Westbrook J. On finding common neighborhoods in massive graphs. Theor. Comput. Sci., 1–3(299):707–718, 2003.
4. Buriol L.S., Frahling G., Leonardi S., Marchetti-Spaccamela A., and Sohler C. Counting triangles in data streams. In Proc. 25th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2006, pp. 253–262.
5. Chakrabarti A., Cormode G., and McGregor A. A near-optimal algorithm for computing the entropy of a stream. In ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 328–335.
6. Cormode G. and Muthukrishnan S. Space efficient mining of multigraph streams. In Proc. 24th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2005, pp. 271–282.
7. Das Sarma A., Gollapudi S., and Panigrahy R. Estimating PageRank on graph streams. In Proc. 27th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2008, pp. 69–78.
8. Demetrescu C., Escoffier B., Moruz G., and Ribichini A. Adapting parallel algorithms to the w-stream model, with applications to graph problems. In Mathematical Foundations of Computer Science, 2007, pp. 194–205.
9. Demetrescu C., Finocchi I., and Ribichini A. Trading off space for passes in graph streaming problems. In ACM-SIAM Symposium on Discrete Algorithms, 2006, pp. 714–723.
10. Elkin M. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. In Int. Colloquium on Automata, Languages and Programming, 2007, pp. 716–727.
11. Elkin M. and Zhang J. Efficient algorithms for constructing $(1 + \varepsilon, \beta)$-spanners in the distributed and streaming models. Distrib. Comput., 18(5):375–385, 2006.
12. Feigenbaum J., Kannan S., McGregor A., Suri S., and Zhang J. Graph distances in the data-stream model. SIAM J. Comput., 38(5):1708–1727, 2008.
13. Feigenbaum J., Kannan S., McGregor A., Suri S., and Zhang J. On graph problems in a semi-streaming model. Theor. Comput. Sci., 348(2–3):207–216, 2005.
14. Ganguly S. and Saha B. On estimating path aggregates over streaming graphs. In Int. Symp. on Algorithms and Computation, 2006, pp. 163–172.
15. Henzinger M.R., Raghavan P., and Rajagopalan S. Computing on data streams. External memory algorithms, 1999, pp. 107–118.
16. McGregor A. Finding graph matchings in data streams. In APPROX-RANDOM, 2005, pp. 170–181.
17. Muthukrishnan S. Data Streams: Algorithms and Applications. Foundations and Trends in Theoretical Computer Science, 1(2), 2005.
18. Zelke M. *k*-connectivity in the semi-streaming model. CoRR, cs/0608066, 2006.
19. Zelke M. Weighted matching in the semi-streaming model. In Proc. Symp. on Theoretical Aspects of Computer Science, 2008.

Graph Streams

▶ Graph Mining on Streams

Graph Theory

▶ Graph Data Management in Scientific Applications
▶ Information Integration Techniques for Scientific Data

Graph-based Clustering

▶ Spectral Clustering

Graphic

▶ Diagram

Graphic Design

▶ Visual Interaction

Graphic Representation of Data

▶ Data Visualization

Graphical Displays of Many Variables

▶ Multivariate Visualization Methods

Graphical Interaction

▶ Direct Manipulation

Graphical Representation

▶ Visual Representation

Graphical User Interfaces

▶ Visual Interfaces

Graphics

▶ Image

Graphics for Continuous Data

▶ Visualizing Quantitative Data

Graphics for Discrete Data

▶ Visualizing Categorical Data

Grid and Workflows

JINJUN CHEN, YUN YANG
Swinburne University of Technology, Melbourne, VIC, Australia

Synonyms

Workflow on grid

Definition

Built on Internet and World Wide Web, the Grid is a new class of infrastructure which supports coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [6,7]. In a grid architecture, a grid workflow management system is a type of user-level grid middleware. It aims to support large-scale sophisticated scientific and business processes in a variety of complex e-science and e-business applications [7,10,12,15]. Such sophisticated processes are modeled or redesigned as grid workflow specifications at build-time stage by some modeling languages such as Grid Workflow Execution Language (GWEL), Abstract Grid Workflow Language (AGWL), and Martlet [4,5,9]. The specifications normally contain a large number of computation, data and/or transaction intensive activities [1,2,12]. Then, at run-time instantiation stage, grid workflow instances are created [4]. Finally, at run-time execution stage, grid workflow instances are executed by facilitating the super computing and data sharing capability of underlying grid infrastructure to complete the computation, data and/or transaction intensive activities [8,11].

Historical Background

Workflow research can be traced back to mid 1980s. In 1993, the WfMC (Workflow Management Coalition) was founded to promote the research and development of workflow technologies such as workflow reference model [13]. WfMC mainly focuses on business workflows, i.e., workflows in business applications. According to WfMC, workflow is defined as the computerized facilitation or automation of a business process, in whole or part [13].

Grid research can be traced back to mid 1990s. It aims to support the sharing of large-scale distributed and heterogeneous resources in a full and negotiable fashion. Resource providers and consumers can dynamically negotiate the sharing mode and extent. For example, they can share resources in client/server mode at one time, but then change to peer-to-peer fashion at another time [7].

With intensive research poured into grid, around the year 2000 the term grid workflow came into picture. Its basic idea is to execute large-scale complex workflows on grid by facilitating the grid's powerful computing and resource sharing capability. The main motivation for grid workflow is that many grid applications such as climate modeling and disaster recovery simulation often require the creation of a collaborative workflow management system as part of their sophisticated problem solving processes so that scientists and business people who lack the low-level expertise can still utilize the current generation of grid toolkits, such as GT4 (Globus Toolkits), Gridbus, and SwinGrid [7,14,15].

Foundations

Traditionally in the business workflow domain, a business workflow is more about logic between activities rather than computation, data or transaction intensive. As a result, there is a very modest level data transfer between activities. However, a grid workflow is often computation, data or transaction intensive. Hence, the key scientific fundamentals in grid workflow area are how to accommodate huge amount of data transfer, computation and transaction processing, and how to schedule and map to the computation capable resources in the underlying grid infrastructure [4,14]. Research and development are ongoing to tackle the key fundamentals [10,11,14,15].

Key Applications

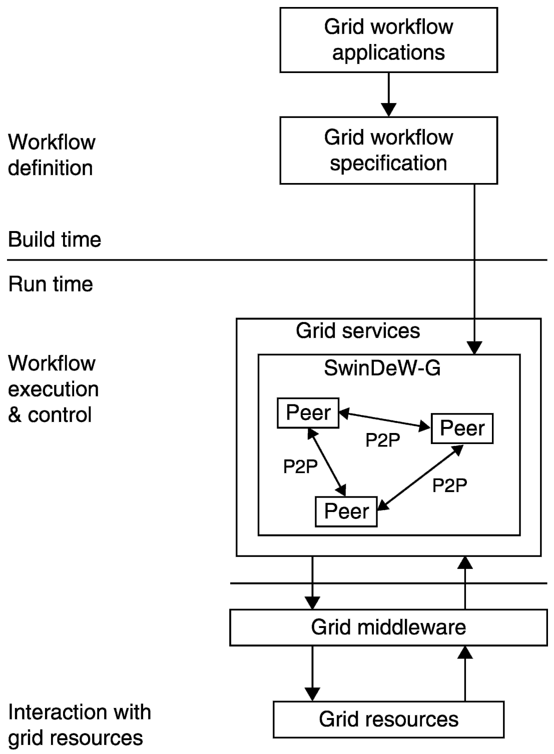
Grid workflow aims to support large-scale sophisticated scientific and business processes in a variety of complex e-science and e-business applications. Typically, such e-science applications are data and computation intensive such as climate modeling, disaster recovery simulation, earth observation data processing, and high energy physics. Such e-business applications are often transaction intensive such as bank, stock modeling, and insurance processing [7,10,12,14,15].

Future Directions

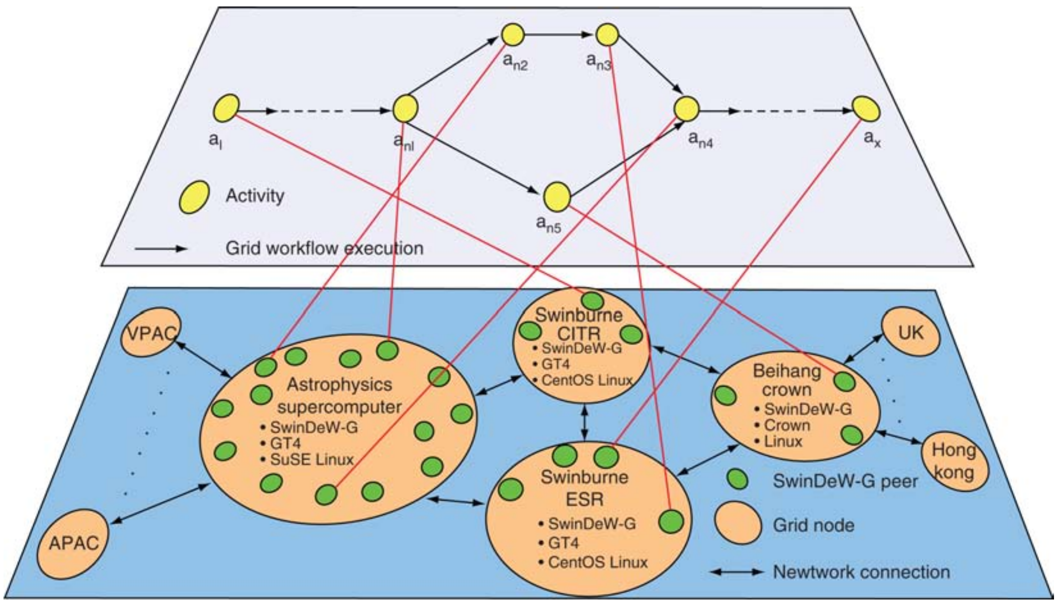
Since in a business workflow, there is a very modest level data transfer between activities, the interaction between activities is normally performed via a centralized workflow engine. However, a grid workflow is normally computation, data and/or transaction intensive. There is often huge amount of data transfer and processing between activities. Therefore, it is inefficient or even congests the centralized workflow engine if the interaction between activities is performed via the centralized engine. As such, the interaction with data transfer should be decentralized. Peer-to-peer processing becomes an ideal alternative. At the same time, the centralized monitoring and management are still necessary. Thus, in overall terms, a promising future direction can be peer-to-peer based interaction with huge amount of data transfer between activities plus centralized monitoring and management with modest information interaction between activities and the centralized engine [4,14]. In this direction, many issues need to be investigated such as global monitoring of distributed peers, workflow scheduling and modeling in support of data, computation and/or transaction intensity.

Experimental Results

The overall architecture of a grid workflow management system can be depicted in Fig. 1. Currently, many grid workflow management systems such as SwinDeW-G, Kepler, Taverna, Triana, Gridbus Workflow, and Pegasus have been developed to experimentally perform the execution of grid workflows [10,11,14,15]. Among them, SwinDeW-G (Swinburne Decentralised Workflow for Grid) as depicted in Fig. 2 is a peer-to-peer based grid workflow management system which naturally matches to the feature of computation, data and/or transaction intensity [3,14]. In SwinDeW-G,


Grid workflow applications
Grid workflow specification
Workflow definition
Build time
Run time
Grid services
SwinDeW-G
Workflow execution & control
Peer
Peer
Peer
P2P
P2P
P2P
Grid middleware
Interaction with grid resources
Grid resources

Grid and Workflows. Figure 1. Overall architecture of a grid workflow management system.



Grid and Workflows. Figure 2. SwinDeW-G system.

a grid workflow is executed by different peers that are distributed in the underlying grid infrastructure. Different peers communicate with each other directly in a peer-to-peer fashion. Kepler supports fault tolerance and p2p or centralised data transfer [10]. Taverna uses centralised approach to support data transfer and retry for fault tolerance [11]. Triana uses p2p for data transfer [15]. Gridbus Workflow also uses a centralised approach for data transfer. Pegasus supports the mapping from abstract workflow specification to specific grid resources for workflow execution [15]. There are more grid workflow management systems which can be referred to [15]. In general, many experimental practices and implementations of grid workflow management are being widely conducted.

Cross-references

▶ Activities
▶ Adaptive Workflow/Process Management
▶ Grid File
▶ Grid Storage
▶ Scheduler
▶ Scientific Workflows
▶ Workflow Management and Workflow Management System
▶ Workflow Management Coalition
▶ Workflow Model
▶ Workflow Schema

Recommended Reading

1. Abramson D., Kommineni J., McGregor J.L., and Katzfey J. An atmospheric sciences workflow and its implementation with web services. Future Generation Comput. Syst., 21(1):69–78, 2005.
2. Aloisio G., Cafaro M., Carteni G., Epicoco I., Quarta G., and Raolil A. GridFlow for earth observation data processing. In Proc. 2005 Int. Conf. on Grid Computing and Applications, 2005, pp. 168–176.
3. Chen J. and Yang Y. Adaptive selection of necessary and sufficient checkpoints for dynamic verification of temporal constraints in grid workflow systems. ACM Trans. Auton. Adapt. Syst., 2(2):Article 6, 2007.
4. Cybok D. A Grid Workflow Infrastructure. Concurrency comput. Pract. Experience (Special Issue on Workflow in Grid Systems), 18(10):1243–1254, 2006.
5. Fahringer T., Pllana S., and Villazon A. A-GWL: Abstract grid workflow language. In Proc. 4th Int. Conf. on Computational Science, LNCS 3038, 2004, pp. 42–49.
6. Foster I., Kesselman C., and Tuecke S. The anatomy of the grid: enabling scalable virtual organizations. Int. J. Supercomput. Appl., 15(3):200–222, 2002.
7. Foster I., Kesselman C., Nick J., and Tuecke S. The physiology of the grid: an open grid services architecture for distributed systems integration. In Proc. 5th Global Grid Forum Workshop, 2002.
8. Fox G.C. and Gannon D. Concurrency Comput. Pract. Exp. (Special Issue: Workflow in Grid Systems). 18(10):1009–1019, 2006.
9. Goodman D. Introduction and evaluation of Martlet, a scientific workflow language for abstracted parallelisation. In Proc. 16th Int. World Wide Web Conference, 2007, pp. 983–992.
10. Ludäscher B., Altintas I., Berkley C., Higgins D., Jaeger-Frank E., Jones M., Lee E., Tao J., and Zhao Y. Scientific workflow

management and the Kepler system. Concurrency Comput. Pract. Exp., 18(10):1039–1065, 2006.
11. Oinn T., Greenwood M., Addis M., Alpdemir M.N., Ferris J., Glover K., Goble C., Goderis A., Hull D., Marvin D., Li P., Lord P., Pocock M.R., Senger M., Stevens R., Wipat A., and Wroe C. Taverna: Lessons in creating a workflow environment for the life sciences. Concurrency Comput. Pract. Exp., 18(10):1067–1100, 2006.
12. Simpson D.R., Kelly N., Jithesh P.V., Donachy P., Harmer T.J., Perrott R.H., Johnston J., Kerr P., McCurley M., and McKee S. GeneGrid: a practical workflow implementation for a grid based virtual bioinformatics laboratory. In Proc. of the UK e-cience All Hands Meeting, 2004, pp. 547–554.
13. Workflow Management Coalition: The Workflow Reference Model. TC00–1003, 1995, http://www.wfmc.org.
14. Yang Y., Liu K., Chen J., Lignier J., and Jin H. Peer-to-peer based grid workflow runtime environment of SwinDeW-G, In Proc. 3rd IEEE Int. Conf. on e-Science and Grid Computing, 2007, pp. 51–58.
15. Yu J. and Buyya R. A taxonomy of workflow management systems for grid computing. J. Grid Comput., 3(3):171–200, 2005.

Grid File

► Hash-based Indexing

Grid File (and Family)

Apostolos N. Papadopoulos[1],
Yannis Manolopoulos[1], Yannis Theodoridis[2],
Vassilis Tsotras[3]
[1]Aristotle University of Thessaloniki, Thessaloniki, Greece
[2]University of Piraeus, Piraeus, Greece
[3]University of California at Riverside, Riverside, CA, USA

Definition

The Grid File is a *multidimensional indexing* scheme capable to efficiently index database records in a symmetrical manner, i.e., by avoiding the distinction between primary and secondary keys. The structure is dynamic and adapts gracefully to its contents under insertions and deletions. A single record retrieval costs two disk accesses at most (upper bound), whereas range queries and partial match queries are also executed efficiently. The Grid File can be thought of as a generalization of dynamic hashing (e.g., *extendible hashing*) in multiple dimensions.

Historical Background

Until the 1980s many file structures for the processing of single attribute queries have been proposed, i.e., queries on the primary key or any secondary key for which a corresponding index has been built. Multi-attribute queries are the ones where the user seeks objects that satisfy constraints (such as equality or range) on several attributes. Such queries can be executed by accessing all the corresponding indices (if they exist) and combine the partial results, or resort to sequential scanning.

To speed up the processing of multiple attribute queries, a better solution is to create an index that leads the search directly to the objects of interest. Such an index can be designed if a data record with k attributes is envisioned as a point in a k-dimensional space. A multi-attribute range query would then be a hyper-rectangle in this k-dimensional space and the answer to it would be all points inside this rectangle. Access methods that can handle multidimensional points are called *Point Access Methods* (PAMs). In 1975, Bentley proposed such a basic PAM, which is called the k-dimensional tree or k-d tree [2]. The Grid File is yet another structure designed to handle similar cases, proposed by Nievergelt, Hinterberger, and Sevcik in 1984 [10]. Since then, several variations have been proposed in the literature in an effort to optimize its space and time performance behavior.

Foundations

The Grid File can be viewed as an access method comprising of two separate parts: (i) the *directory*, and (ii) the *linear scales*. To illustrate this, assume that an index is to be constructed on the *Employee* file using two attributes, say *salary* and *dept* (extension to more dimensions is straightforward). The Grid File imposes a grid on the two-dimensional attribute space. Each cell in this grid corresponds to one data page. The data points that "fall" inside a given cell are stored in the cell's corresponding page. Each cell must thus store a pointer to its corresponding page. This information is stored in the Grid File's *directory*. However, cells that are empty do not use a page. Rather, two or more cells can share a page (i.e., point to the same page). The grid adapts to the data density by introducing more divisions in areas where there are more points.

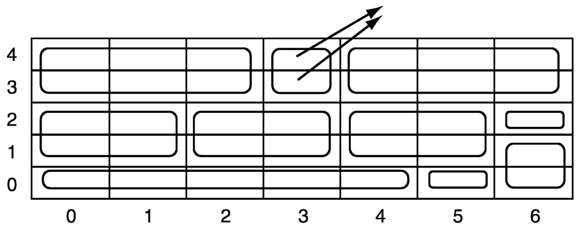
The information of how each dimension is divided (and thus how data values are assigned to cells) is kept through *linear scales*. There is one linear scale per dimension (indexed attribute). Each linear scale is a one-dimensional array that divides the values on a particular dimension in such a way that records (points) are uniformly distributed across cells. An example of a Grid File on the "Dept" and "Salary" attributes appears in Fig. 1.

Searching for a record with given attribute values involves two operations: (i) the Grid File's directory is searched to locate the cell that the record is hosted in, (ii) the cell's pointer is followed to access the corresponding data page (say *A*), and (iii) the record is searched only in data page *A*. If the record is found in *A* then the search terminates successfully, otherwise the search for the specific record is unsuccessful (i.e., the record does not exist). The Grid File can also address multi-dimensional *range queries* by selecting the appropriate cells from each dimension's linear scale. For example, such a query may ask for all employee records with the *salary* attribute ranging between 10K and 40K and the *dept* attribute ranging between 2 and 6. Again, the first step examines the directory and determines the cells that are intersected by the query range in both attributes, then the corresponding pointers to data pages are collected and finally the data pages are examined for relevant records. The accessed cells may also contain some records outside the query range. These records are eliminated from further consideration and they are not returned as part of the query answer.

Inserting a new record in this method is straightforward. First, the two linear scales are searched so as to map the record's *salary* and *dept* attribute values in each dimension. This mapping provides a cell in the directory. This cell is then accessed and using its pointer, the appropriate page, say *A*, for the new record is found. If this page has enough space to accommodate the new record, the insertion process is complete. Otherwise, a new page *B* is allocated. If page *A* was pointed to by more than one cell, the pointers of these cells are rearranged such that some will point to page *A* and some to page *B* (and the records of page *A* are redistributed accordingly between *A* and *B*). If page *A* was pointed by a single cell and overflows, a reorganization of the Grid File is needed. This reorganization will expand the directory and the scales by introducing a new column (or row) of cells.

In the sequel, the insertion process is illustrated by an example given in Fig. 2 White dots correspond to existing records, whereas black dots are used to indicate new records being inserted to the Grid File. Assume that each data page can host at most three records. Practically, this number is larger in real applications and depends on the size of the data page and the number of attributes. Assume that initially the Grid File is empty (does not contain any records). The first three records can be easily accommodated in the single data page *A*pointed by the single cell of the directory (corresponding to the whole data space), as illustrated in Fig. 2(a). The next inserted record is *d*. However, the new record cannot be hosted by data page *A* because its capacity is exceeded. Therefore, another data page *B* is allocated and records are distributed to the two data pages as it is shown in Fig. 2 (b). The next two insertions for records *e* and *f* do not cause any reorganization since the new records can be easily accommodated in the corresponding data pages pointed by the cells, as illustrated in Fig. 2(c). Finally, the insertion of record *g* causes an overflow in data page *A*. The corresponding cell is split again using the other attribute and one more data page is allocated and records are distributed accordingly. The final shape of the Grid File is given in Fig. 2(d).

4	12
3	9-11
2	5-8
1	4
0	1-3



< 10K	11-30K	31-50K	51-60K	61-90K	91-100K	>101K
0	1	2	3	4	5	6

Grid File (and Family). Figure 1. A Grid File.

Deletions are also supported, but they are handled differently. Initially, the deleted record is located using the directory and the corresponding data page is determined. If the record is found, it is deleted from the data page. Instead of overflowing, data page deletions may cause the underutilization effect, which means that several data pages may contain too few records. Therefore, appropriate merging operations are required to maintain the storage utilization of the Grid File at an acceptable level. For a detailed description of the methods used for merging as well as for splitting, the reader is directed to [10].

The Grid File has a set of nice properties: (i) it is based on simple mechanisms for insertion, deletion, and search; (ii) it guarantees only two disk accesses for exact match queries (one for the directory and one for the data page); and, (iii) it treats all indexed attributes symmetrically which leads to simple directory management policies. However, it has a set of serious disadvantages such as: (i) it introduces a space overhead for the directory, which can be large for high-dimensional spaces, (ii) it has an extra update overhead, since reorganization affects many cells and not only the cell with the overflowing page; and, (iii) it suffers from performance degradation if the attributes are correlated, since the uniform scheme for performing splits is not adequate to guarantee performance efficiency.

Toward improving the behavior of the Grid File several research efforts have been performed. Some of these efforts are highlighted in the following lines. One of the first variations, the BANG File, has been proposed by Freeston [4]. The BANG File is based on a self-balanced tree-based directory, which better reflects changes of the data distribution. To achieve better storage utilization the Twin Grid File has been proposed by Hulflesz, Six, and Widmayer in [5]. The new scheme is as efficient as the original Grid File during range query processing but shows significant improvements regarding storage utilization. Blanken et al. proposed the Generalized Grid File [3], which offers fast access for single attribute queries. The Multilevel Grid File [12] is another research effort to improve the performance of the original structure for exact-match, partial-match, and range queries. This new scheme uses multiple grid levels and succeeds in better directory management and more efficient query processing than the original structure.

In addition to the variations proposed in the literature, there are efforts to use the Grid File in a parallel environment, toward more efficient data management. In [13] the authors study the problem of partitioning a Grid File to multiple disk devices toward more efficient search. When a data page split is performed, the new data page is carefully allocated to a disk. Since disks can be accessed in parallel, several data pages can be read simultaneously during range query processing, offering significant performance improvements in comparison to a single-disk system. More complex queries on Grid Files, like *spatial joins*, have been also parallelized [6] toward reduced query response times. A different approach has been followed by [7]. The authors have proposed a method to load a Grid File in parallel. The data file is initially partitioned to the available processors using dynamic programing and sampling, and then each processor builds its own part of the Grid File.

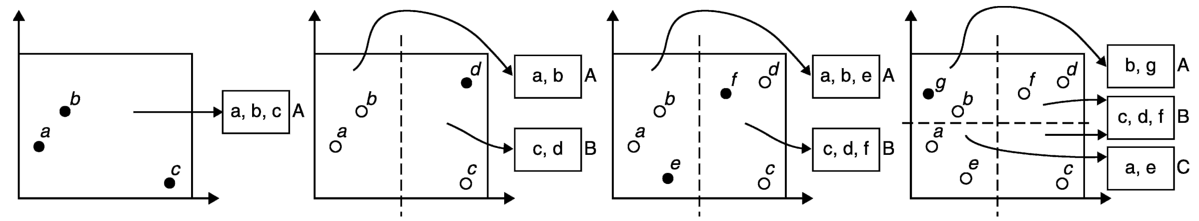
Key Applications

Spatial Databases

In Spatial Databases it is commonly required to join spatial data sets or perform nearest neighbor searches. Several algorithms have been proposed for such operations by adopting the Grid File as the underlying access method [1].

Data Mining

The Grid File can also be used for clustering data sets to identify correlation characteristics of the underlying value space. This stems from its ability to group



Grid File (and Family). Figure 2. Insertions in the Grid File.

patterns into blocks and cluster them with respect to the blocks by a topological neighbor search algorithm [11].

Data Warehouses

The Grid File can be used for efficient data cube storage in warehouses [9].

Future Directions

The Grid File has eventually emerged as a popular theoretical access method. However, although it has been widely honored in theory, in practice it has not been used by the database industry.

Experimental Results

A detailed performance evaluation of the Grid File can be found in [10], where the authors offer a detailed experimental section studying the properties of the structure regarding capacity of data pages, directory size, and evaluation of splitting and merging policies. Moreover, interesting experimental results can be found in [5,3], which compare the original Grid File with the corresponding variation proposed in each work.

Cross-references

- ▶ Extendible Hashing
- ▶ K-D trees
- ▶ Multidimensional Indexing
- ▶ Range Query
- ▶ Spatial Join

Recommended Reading

1. Becker L., Hinrichs K., and Finke U. A new algorithm for computing joins with grid files. In Proc. 9th Int. Conf. on Data Engineering, 1993, pp. 190–197.
2. Bentley J.L. Multidimensional binary search trees used for associative searching. Commun. ACM, 18(9):509–517, 1975.
3. Blanken H.M., Ijbema A., Meek P., and van den Akker B. The generalized grid file: description and performance aspects. In Proc. 14th Int. Conf. on Data Engineering, 1990, pp. 380–388.
4. Freeston M. The BANG file: a new kind of grid file. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1987, pp. 260–269.
5. Hutflesz A., Six H.-W., and Widmayer P. Twin grid files: space optimizing access schemes. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1988, pp. 183–190.
6. Kim J.-D. and Hong B.-H. Parallel spatial join algorithms using grid files. In Proc. 6th Int. Conf. on Database Systems for Advanced Applications, 1999, pp. 226–234.
7. Li J., Rotem D., and Srivastava J. Algorithms for loading parallel grid files. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1993, pp. 347–356.
8. Lim Y. and Kim M. A Bitmap Index for Multidimensional Data Cubes. In Proc. 15th Int. Conf. Database and Expert Syst. Appl., 2004, pp. 349–358.
9. Luo C., Hou W.C., Wang C.F., Wang H., and Yu X. Grid File for Efficient Data Cube Storage. Computers and their Applications, conference paper CATA, pp. 424–429, 2006.
10. Nievergelt J., Hinterberger H., and Sevcik K.K. The grid file: an adaptable, symmetric multikey file structure. ACM Trans. Database Syst., 9(1):38–71, 1984.
11. Schikuta E. and Erhart M. The BANG-clustering system: grid-based data analysis, In Adv. in Intelligent Data Analysis, 2nd Int. Symp., 1997, pp. 513–524.
12. Whang K.-Y. and Krishnamurthy R. The multilevel grid file – a dynamic hierarchical multidimensional file structure. In Proc. 2nd Int. Conf. on Database Systems for Advanced Applications, 1991, pp. 449–459.
13. Zhou Y., Shekhar S., and Coyle M. Disk allocation methods for parallelizing grid files. In Proc. 10th Int. Conf. on Data Engineering, 1994, pp 243–252.

Grid Workflow

▶ Scientific Workflows

Group Difference

▶ Emerging Patterns

Grouping

▶ Abstraction

GUID

▶ Resource Identifier

GUIs for Web Data Extraction

Cai-Nicolas Ziegler
Siemens AG, Munich, Germany

Synonyms

Visual web data extraction; Wrapper generator GUIs; Visual web information extraction

Definition

While content management systems (CMS) are geared towards *adding* presentational information to relational and structured data from database systems, thus dynamically generating HTML documents, the goal of GUIs for Web data extraction is diametrically opposed: The commonly semi-automatic Web data extraction tools intend to *remove* all presentational information from Web pages, so that only pure structured content remains. The extraction process itself does not address single documents, but template types, such as the product page of an online retailer or the news page template of an online journal. That is, for each template type, one set of extraction rules is generated. These extraction rules are defined in a graphical manner, by selecting the pieces of information that are relevant and by assigning labels to them. To this end, GUIs are used that largely resemble Web browsers, extended by user actions for highlighting and assigning appropriate labels. For instance, for a book seller's product page, such labels were "author," "publisher," "price," or "number of pages."

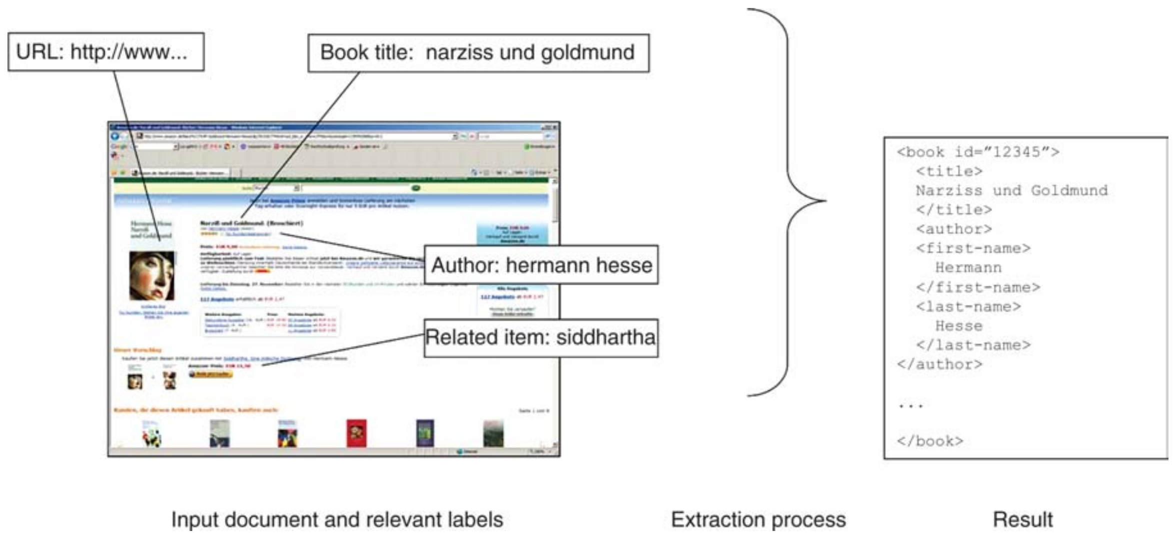
The learned extraction rules for one document template, called "wrapper" [5], are then applied to those documents that have been generated based on the template type at hand. Referring to the book seller's example, one could now extract structured relational information for each book published on the online retailer's website (see Fig. 1).

Historical Background

GUIs for Web data extraction are advanced variants of wrapper induction systems [5]. Being graphics-oriented, these follow-ups add to the state of art by making the extraction process more user-friendly and easier to manage, saving time and making Web data extraction accessible to a broad range of non-experts as well.

Early Web data extraction systems have been implemented by writing small programs in scripting languages like Perl. These programs, termed screen scrapers, contained manually defined extraction rules that focused on structural communalities of information types to be extracted from the documents, e.g., price, title, and publisher. The huge disadvantage was that people had to look at the very HTML code of documents, comparing several instances of the document template used and trying to figure out which sequences of HTML documents and/or textual content remained static and which appeared dynamic.

The tedious nature of screen scraping favored the naissance of wrapper induction systems. These applications did not require the human to manually write extraction rules, but to label a number of documents and have machines inductively learn the extraction rules. Early seminal systems include Kushmerick's wrapper generator [5], Stalker [6,7], and RoadRunner [4]. The latter system takes HTML documents that have been generated using the same template as input



```
<book id="12345">
  <title>
  Narziss und Goldmund
  </title>
  <author>
  <first-name>
    Hermann
  </first-name>
  <last-name>
    Hesse
  </last-name>
</author>

...

</book>
```

GUIs for Web Data Extraction. Figure 1. Web data wrappers extract relevant information from Web documents and store the labeled results into data structures, such as XML documents.

and infers a most specific grammar that subsumes the variety of all of them.

At the same time, early GUI-based extraction systems have begun to evolve. NoDoSe [1] extracts information from plain string sources and provides a user interface for instance labeling. Kushmerick proposed Wien [1], a visual support tool, as an extension to his early wrapper system. Wien receives a set of training pages where the user can label relevant information and learn a wrapper.

The shift from research towards commercial tools became manifest with the advent of Lixto [2], which started off as an academic project and transformed into a commercial product soon after. (See *http://www.lixto.com* for details). The early Lixto system allowed the user to mark relevant, information-bearing passages in a browser-like window, assign labels to these passages, e.g., "price," "manufacturer," etc., test the wrappers on unseen data, and modify the learned extraction rules. Lixto incorporated a system for inductive extraction rule learning based on the ELOG [3] language, which bears traits of both PROLOG and XPATH (see Fig. 2). The extracted and labeled result was written to an XML document.

Complex enterprise-scale commercial solutions have evolved since then, boasting graphical user interfaces that allow for recording HTML form inputs and interaction events in a macro-like fashion, handling dynamic and AJAX-enhanced documents, and creating complex processing and wrapping workflows.

Foundations

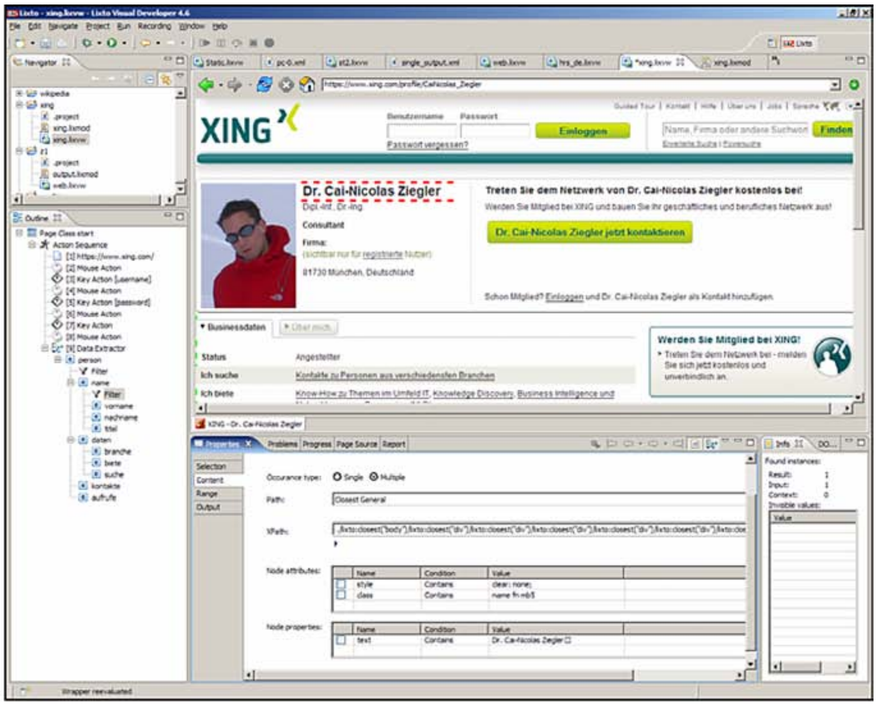
With the advent of graphical user interfaces for Web data extraction, the focus of supervised wrapper generation has greatly changed from one relying on inductive machine learning techniques to one that intends to offer the non-expert user a broad range of easy-to-use actions that allow him to specify what he intends to extract in a comfortable and 100% WYSIWYG fashion. The goal is not so much the use of intelligent techniques for the generalization and conjunction of extraction patterns as much more to cover the full content extraction and publication process, by means of an end-to-end lifecycle approach. This includes the easy interfacing to databases for the parameterization of wrapper input data and output storage, the availability of APIs and Web service interfaces in order to embed the extraction process as yet another component or service of a larger Web application infrastructure, as well as the ability to design wrappers that can be reused as subordinate components in other wrappers.

At the time of this writing, the major vendors of commercial state-of-the-art GUIs for Web data extraction are Kapow (*http://www.kapow.com*), Lixto (*http://www.lixto.com*), Denodo (*http://www.denodo.com*), and Twinsoft (*http://www.twinsoft.fr*). All products offered exhibit a similar look and feel, with an HTML browser window being the central component that all other dialogues and tool frames are clustered around (see Fig. 3).

Typically, the user can indicate regions that contain relevant information by highlighting the respective region with his mouse. Moreover, most GUIs offer the possibility to navigate, in another window, the DOM tree of the HTML document at hand and select the nodes that match the demanded criteria. Next, the user can assign labels to these selected information regions and thus generate extraction rules based on path expressions. Some tools, such as Lixto Visual Developer, attempt to generalize the matching extraction paths in an intelligent fashion, assuming wildcards for appropriate location steps so as to avoid over-fitting [3]. In order to handle Dynamic HTML and cascading stylesheets (CSS), most graphical

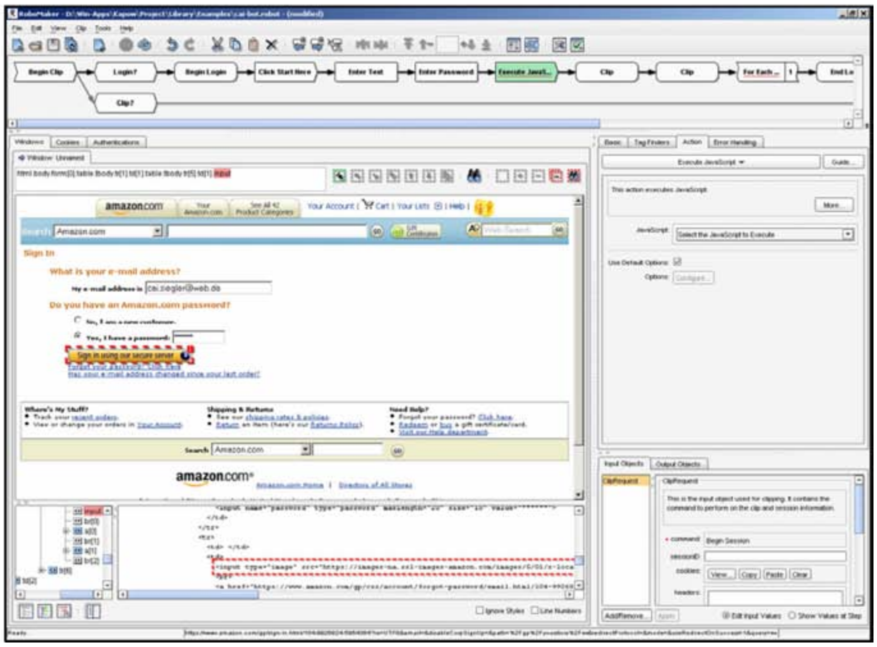
```
 tablesq(S,X)  ← document("www.ebay.com/",S),subsq(S,(.body,[]),(.table,[]),(.table,[]),X),
                 before(S,X,(table,[elementtext,item,substr]),0,0,-,-,-),after(S,X,.hr,0,0,-,-)
  record(S,X)  ← tableseq(-,S),subelem(S,.table,X)
 itemnum(S,X)  ← record(-,S),subelem(S,*.td,X),notbefore(S,X,.td,100)
 itemdes(S,X)  ← record(-,S),subelem(S,(*.td.*.content,[(a,,substr)],X)
   price(S,X)  ← record(-,S),subelem(S,(*.td,[(elementtext,\var[Y].*,regvar)],X),isCurrency(Y)
    bids(S,X)  ← record(-,S),subelem(S,*.td,X),before(S,X,.td,0,30,Y,-)price,(-,Y)
currency(S,X)  ← price(-,S),subtext(S,\var[Y],X), is Currency(Y)
 pricewc(S,X)  ← price(-,S),subtext(S,[0-9]+\.[0-9]+,X)
```

GUIs for Web Data Extraction. Figure 2. Generated ELOG rules for extracting relevant information, such as item number, description, price, and bids, from ebay (*http://www.ebay.com*).



Lixto - xing.lixvw - Lixto Visual Developer 4.6
XING
Dr. Cai-Nicolas Ziegler
Treten Sie dem Netzwerk von Dr. Cai-Nicolas Ziegler kostenlos bei!
Dr. Cai-Nicolas Ziegler jetzt kontaktieren
Werden Sie Mitglied bei XING!
Navigator
Outline
Properties
Wrapper reevaluated

a



b

GUIs for Web Data Extraction. Figure 3. Screenshots of two state-of-the-art commercial GUIs for Web data extraction, (a) Lixto Visual Developer and (b) Kapow Mashup Server.

extraction systems have a built-in JavaScript interpretation engine. Debugging capabilities are also state-of-the-art, allowing the user to step through the entire extraction process, from the wrapper's entering of the website to the eventual extraction and storage.

Commercial products for Web data extraction have matured over the years and have become user-friendly and extremely powerful tools. From a scientific perspective, though, there is little new technology to discover.

Key Applications

The number of applications for wrapper generation GUIs is rife and extends to all areas where information extraction on the Web is put to use. Integration of Web data is surely the most important application scenario and heavily advertised by solution vendors. However, instead of referring to "integration," the synonymous term "mashup" is used, which hints at the deliberate placement of such tools into the Web 2.0 universe. For instance, the GUIs could be used to create extraction wrappers for news articles from various sources, which are then presented in one page. Another mashup application scenario is the creation of value-added services that exploit heterogeneous information sources: For instance, the implementation of a travel service that acts as meta search engine for other travel search engines, integrates their results into a single list, and enriches each entry by resorting to another wrapper which obtains information for the destination country at hand from encyclopedic websites such as Wikipedia (*http://www.wikipedia.com*).

Cross-references

▶ Fully-Automatic Web Data Extraction
▶ Information Extraction
▶ Information Filtering
▶ Wrapper Induction

Recommended Reading

1. Adelberg B. NoDoSE: A tool for semi-automatically extracting structured and semi-structured data from text documents. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 283–294.
2. Baumgartner R., Flesca S., and Gottlob G. Visual web information extraction with lixto. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 119–128.
3. Baumgartner R., Flesca S., and Gottlob G. The ELOG web extraction language. In Proc. Artificial Intelligence on Logic for Programming, 2001, pp. 548–560.
4. Crescenzi V., Mecca G., and Merialdo P. RoadRunner: towards automatic data extraction from large web sites. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 109–118.
5. Kushmerick N., Weld D., and Doorenbos R. Wrapper induction for information extraction. In Proc. 15th International Joint Conference on Artificial Intelligence, 1997, pp. 119–128.
6. Muslea I., Minton S., and Knoblock C. Stalker: learning extraction rules for semistructured, web-based information sources. In Proc. of the AAAI Workshop on AI and Information Integration, 1998.
7. Muslea I., Minton S., and Knoblock C. Hierarchical wrapper induction for semistructured information sources. Auton. Agent. Multi Agent Syst., 4(1–2):93–114, 2001.