Factor Screening Method

► Screening Method

Fastest-Path Computation

DONGHUI ZHANG
College of Computer & Information Science,
Northeastern University, Boston, MA, USA

Synonyms

Fastest route; Driving direction; Shortest paths; Dijktra's shortest path algorithm; A* algorithm

Definition

In the United states, only 9.3% of the households do not have cars. Driving is part of people's daily life. GIS systems like MapQuest and MapPoint are heavily relied on to provide driving directions. However, surprisingly enough, existing systems either ignore the driving speed on road networks or assume that the speed remains constant on the road segments. In both cases, the users' preferred leaving time does not affect the query result. For instance, MapQuest does not ask the users to input the day and time of driving. However, during rush hour, inbound highways to big cities allow much lower speeds than usual. Therefore, a fastest path computed during non-rush hours, which may consist of some inbound highway segments, may not remain the fastest path during rush hour.

Consider a road network modeled as a graph where each node is a road intersection and each edge is a road segment. Let each edge store a speed pattern, e. g., a piecewise-constant function. For instance, in a working day during rush hour (say from 7am to 9am), the speed is 0.3 miles per minute (mpm) and 1mpm at other times of the day.

The ***Time Interval All Fastest Paths (allFP) Query*** is defined as follows. Given a source node s, an end node e, and a leaving time interval at s, the *allFP* query asks to enumerate all fastest paths, each corresponding to a disjoint sub-interval of leaving time. The union of all sub-intervals should cover the entire query time interval.

An allFP query example is: *I may leave for work any time between 7am and 9am; please suggest all fastest paths, e. g., take route A if the leaving time is between 7 and 7:45 and take route B otherwise*.

It is also interesting to solve the allFP problem with an arrival time interval. For instance: *I need to drive from my home to New York International Airport. Please suggest all fastest paths if the anticipated arrival time is between 5pm and 6pm*.

There are two characteristics that distinguish the allFP problem from other shortest/fastest path problems.

- The query is associated with a leaving/arrival time INTERVAL, not a time instant. In fact, if the leaving time were fixed as a time instant, many existing algorithms could be applied, such as the Dijkstra's shortest path computation and the A* algorithm.
- Time is continuous. If time were distinct, e. g., one can only leave precisely at the top of the hour, one could run existing time-instant algorithms multiple times.

Historical Background

Most existing work on path computation has been focused on the shortest-path problem. Several extensions of the Dijkstra algorithm have been proposed, mainly focusing on the maintenance of the priority queue. The A* algorithm [8] finds a path from a given start node to a given end node by employing a heuristic estimate. Each node is ranked by an estimate of the best route that goes through that node. A* visits the nodes in order of this heuristic estimate. A survey on shortest-path computation appeared in [11].

Performance analysis and experimental results regarding the secondary-memory adaptation of shortest path algorithms can be found in [4,13]. The work in [3] contributes on finding the shortest path that satisfies some spatial constraints. A graph index that can be used to prune the search space was proposed in [15].

One promising idea to deal with large-scale networks is to partition a network into fragments. The boundary nodes, which are nodes having direct links to other fragments, construct the nodes of a high-level, smaller graph. This idea of hierarchical path-finding has been explored in the context of computer networks [6] and in the context of transportation systems [5]. In [12], the materialization trade-off in hierarchical shortest path algorithms is examined.

In terms of fastest-path computations, there exists work assuming the discrete model [1,9], the flow speed model [14], or theoretical models beyond road network [10]. The discrete model [1,9] assumes discrete time. The flow speed model work [14] addresses the fastest path query for a given leaving time instant, not a time interval. The theoretical model work [10] suggests operations on travel functions that are necessary without investigating how this operation can be supported.

To compute fastest paths on a road network with a leaving/arrival time interval and with continuous time, one can utilize a novel extension of the A* algorithm. More details are given below.

Scientific Fundamentals

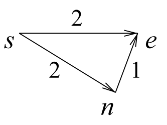
A simple extension to the A* algorithm can NOT be used to solve the allFP query. Let n_0 be the node to be expanded next and let n_0 have three neighbor nodes, n_1, n_2 and n_3. A* picks the neighbor node n_i ($i \in [1..3]$) to continue expanding if the travel time from s to n_i plus the estimated travel time from n_i to e is the smallest. The problem is that since the leaving time is not a fixed time instant, depending on the leaving time instant, different neighbors should be picked.

One possible solution is to expand all such neighbors simultaneously. However, expanding all picked neighbors may result in an exponential number of paths being expanded regardless of the size of the answer set.

Instead, [7] proposed a new algorithm called **IntAllFastestPaths**. The main idea of the algorithm is summarized below:

1. Maintain a priority queue of expanded paths, each of which starts with s. For each path $s \Rightarrow n_i$, maintain $T(l, s \Rightarrow n_i) + T_{est}(n_i \Rightarrow e)$ as a piecewise-linear function of $l \in$ leaving time interval I. Here, $T(l, s \Rightarrow n_i)$ is the travel time from s to n_i, measured as a function of leaving time l. $T_{est}(n_i \Rightarrow e)$ is a lower bound estimation function of the travel time from n_i to the end node e. A straightforward lower-bound estimator is $d_{euc}(n_i, e)/v_{max}$, which is the Euclidean distance between n_i and e, divided by the max speed in the network. A better estimator is called the *boundary node estimator*, which is described later.
2. Similar to the A* Algorithm, in each iteration pick a path from the priority queue to expand. Pick the path whose maintained function's minimum value during I is the minimum among all paths.
3. Maintain a special travel-time function called the *lower border function*. It is the lower border of travel time functions for all identified paths (i. e., paths already picked from the priority queue) that end to e. In other words, for any time instant $l \in I$, the lower border function has a value equal to the minimum value of all travel time functions of identified paths from s to e. This function consists of multiple travel time functions, each corresponding to some path from s to e and some subinterval of I during which this path is the fastest.
4. Stop either when there is no more path left in the priority queue or if the path picked to expand next has a minimum value no less than the maximum value of the lower border function. Report the lower border function as the answer to the allFP query.

Below is a running example. The example involves a simple road network given in Fig. 1. The goal is to find the fastest path from s to e at some time during $I = [6{:}50–7{:}05]$.



$s \to e$: [6-8):1/3
$s \to n$: [6-7):1/3, [7-8):1
$n \to e$: [6-7:08):1/3, [7:08-8):1/10

Fastest-Path Computation, Figure 1 A simple road network. Distances are given on the edges. Speed patterns (#mpm) are given at the right of the network

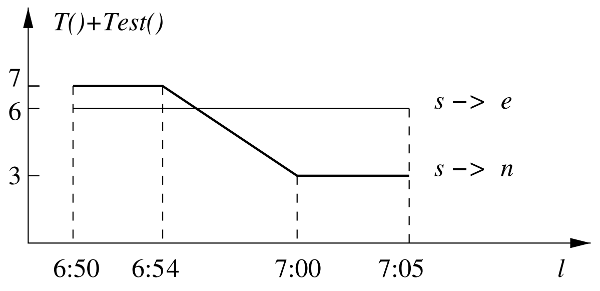
Initially, the priority queue contains only one entry, which corresponds to the unexpanded node s. It has two neighbors e and n. It can be derived that $T(l \in [6{:}50–7{:}05), s \to e) = 6$ min and $T(l \in [6{:}50–7{:}05), s \to n) =$

$$\begin{cases} 6, & l \in [6{:}50–6{:}54) \\ \frac{2}{3}(7{:}00 - l) + 2, & l \in [6{:}54–7{:}00) \\ 2, & l \in [7{:}00–7{:}05] \end{cases}$$

As expressed in step 1 of Algorithm IntAllFastestPaths, in the priority queue, the paths are ordered not by $T()$, but by $T() + T_{est}()$. The functions of the two paths are compared in Fig. 2. Here, $T_{est}(n \Rightarrow e) = 1$ min since $d_{euc}(n, e) = 1$ mile and $v_{max} = 1$ mpm.

According to step 2, the path $s \to n$ is to be expanded next since its minimum value, 3, is smaller than the minimum value, 6, of the path $s \Rightarrow e$.

In general, to expand a path $s \Rightarrow n$, first all the required information for n and its adjacent nodes needs to be retrieved. Then, for each neighbor n_j of n, the following steps need to be followed:



Fastest-Path Computation, Figure 2 Comparison of the functions $T() + T_{\mathrm{est}}()$ associated with paths $s \to e$ and $s \to n$

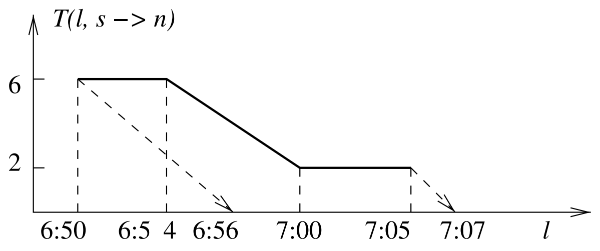
- Given the travel time function for the path $s \Rightarrow n$ and the leaving time interval I from s, determine the time interval during which the travel time function for the road segment $n \to n_j$ is needed.
- Determine the time instants $t_1, t_2, \ldots \in I$ at which the resulting function, i. e., the travel time function for the path $s \Rightarrow n_j$, $T(l \in I,\ s \Rightarrow n_j)$, changes from one linear function to another.
- For each time interval $[t_1, t_2), \ldots$, determine the corresponding linear function of the resulting function $T(l \in I, s \Rightarrow n_j)$.

In this example, the time interval for $n \to e$ is determined to be [6:56, 7:07], as shown in Fig. 3. At time 6:50 (start of I), the travel time along the path $s \to n$ is 6 minutes. Therefore, the start of the leaving time interval for $n \to e$, i. e., the start of arrival time interval to n, is $6{:}50 + 6\,\mathrm{min} = 6{:}56$. Similarly, the end of the leaving time interval is $7{:}05 + 2\,\mathrm{min} = 7{:}07$.

During the time interval [6:56–7:07], the travel time on $n \to e$, $T(l \in \texttt{[6:56 - 7:07]}, n \to e)$ is

$$\begin{cases} 3, & \texttt{if} \quad l \in \texttt{[6:56-7:05)} \\ 10 - \frac{7}{3}(\texttt{7:08} - l), & \texttt{if} \quad l \in \texttt{[7:05-7:07]} \end{cases}$$

There are two cases that trigger the resulting travel time function $T(l, s \Rightarrow n \to e)$ to change from one linear function to another. In the first simple case, the function $T(l, s \Rightarrow n)$ changes. The time instants at which the resulting function changes are the ones at which $T(l, s \Rightarrow n)$ changes.



Fastest-Path Computation, Figure 3 The time interval, [6:56–7:07], during which the speed on $n \to e$ is needed

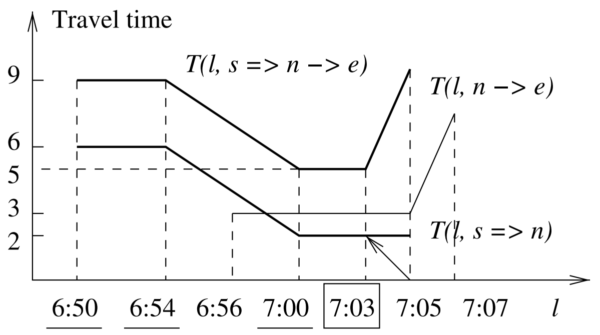
In Fig. 3, these correspond to time instants 6:50, 6:54 and 7:00. In the second trickier case, the changes of the resulting function are triggered by the changes of $T(l, n \to e)$, e. g., at time 7:05. In this example, one can determine that at time 7:03, $T(l, s \Rightarrow n \to e)$ changes. The reason for this is that if one leaves s at 7:03, since the travel time on $s \Rightarrow n$ is 2 minutes, one will arrive at n at 7:05. At that time the travel time function of $n \to e$ changes. To find the time instant 7:03, compute the intersection of the function $T(l, s \Rightarrow n)$ with a 135° line passing through the point (7:05, 0). The time instant 7:03 is the leaving time corresponding to that intersection point.

Now that all the four time instants 6:50, 6:54, 7:00, and 7:03 have been determined, the 4-piece function $T(l \in I, s \Rightarrow n \to e)$ can be derived by combining $T(l, s \Rightarrow n)$ and $T(n \to e)$.

For each l, $T(l, s \Rightarrow n \to e)$ is equal to $T(l, s \Rightarrow n)$ plus $T(l', n \to e)$, where l' is the time at which node n is reached. That is, $l' = l + T(l, s \Rightarrow n)$. The following algorithm should be used to expand a path for every identified time instant $t \in \{t_1, t_2, \ldots\}$ (e. g. 6:50):

- Retrieve the linear function of $T(l, s \Rightarrow n)$ at time t. Let it be $\alpha * l + \beta$.
- Retrieve the linear function of $T(l', n \to e)$ at time $t' = t + (\alpha * t + \beta)$. Let it be $\gamma * l' + \delta$.
- Compute a new linear function $(\alpha * l + \beta) + (\gamma * (l + \alpha * l + \beta) + \delta)$ which can be re-written as $(\alpha * \gamma + \alpha + \gamma) * l + (\beta * \gamma + \beta + \delta)$. This is the linear function as part of $T(l, s \Rightarrow n \to e)$ for the time interval from t to the next identified time instant.

For instance, the combined function $T(l \in \mathrm{I}, n \Rightarrow e)$, which is shown in Fig. 4, is computed as follows. At $t = 6{:}50$, the first linear function is a constant function 6. Hence, $t' = t + 6 = 6{:}56$. The second linear function starting with 6:56 is another constant function 3. Therefore, the combined function is 9, which is valid until the next identified time instant.



Fastest-Path Computation, Figure 4 The time instants at which $T(l, s \Rightarrow n \to e)$ changes to another linear function and the $T(l, s \Rightarrow n \to e)$ function

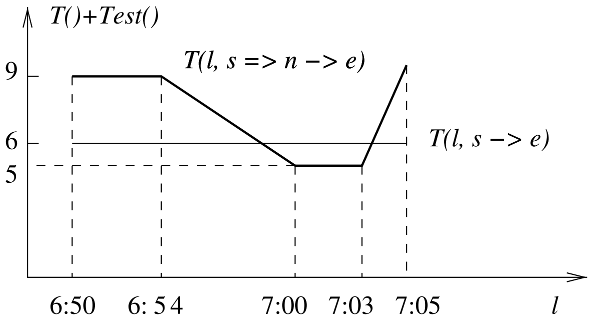
At $t = $ 6:54, the first linear function is $\frac{2}{3}(\texttt{7:00} - l) + 2$. Therefore, $t' = $ 6:54 + 6 = 7:00. The second linear function is 3. The combined function is $\frac{2}{3}(\texttt{7:00} - l) + 5$.

At $t = $ 7:00, the first function is constant 2. At $t' = $ 7:00 + 2 = 7:02, the second function is 3. Thus, the combined function is 5.

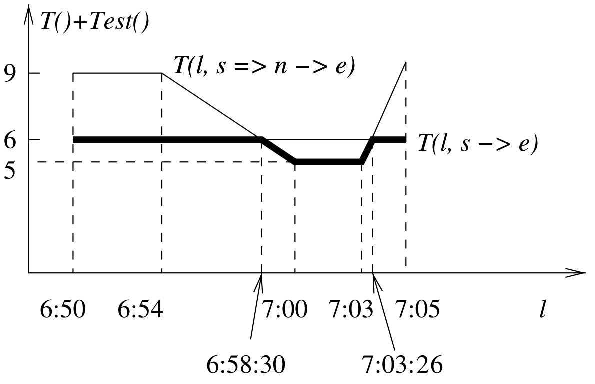
Finally, at $t = $ 7:03, the first function is 2, and at $t' = $ 7:03 + 2 = 7:05, the second function at $10 - \frac{7}{3}(\texttt{7:08} - l')$. Therefore, the combined function is $2 + (10 - \frac{7}{3}(\texttt{7:08} - (l + 2))) = 12 - \frac{7}{3}(\texttt{7:06} - l)$.

After the expansion, the priority queue contains two functions, as shown in Fig. 5. Note that in both functions, the lower bound estimation part is 0 since both paths already end with e.

The next step of Algorithm IntAllFastestPaths is to pick the path $s \Rightarrow n \rightarrow e$, as its minimum value (5 min) is globally the smallest in the queue. An important question that arises here is when to stop expanding, as expanding all paths to the end node is prohibitively expensive. The algorithm terminates when the next path has a minimum value no less than the maximum value of the maintained *lower border function*.



Fastest-Path Computation, Figure 5 The two functions in the priority queue



Fastest-Path Computation, Figure 6 The lower border and the result for Query 3

When there is only one identified path that ends with e, the lower border function is the function of this path. In Fig. 5, $T(l, s \Rightarrow n \rightarrow e)$ is the lower border function. As each new path ending with e is identified, its function is combined with the previous lower border function. For example, in Fig. 6, the new lower border function after the function $T(l, s \rightarrow e)$ is removed from the priority queue is shown as the thick polyline.

The algorithm can terminate if the next path to be expanded has a minimum value no less than the maximum value of the lower border function (in this case, 6). Since the maximum value of the lower border keeps decreasing while the minimum travel time of paths in the priority queue keeps increasing, the algorithm IntAllFastestPaths is expected to terminate very fast. In this example, the set of all fastest paths from s to e when $l \in$ [6:50–7:05] is:

$$\begin{cases} s \rightarrow e, & \texttt{if} \quad l \in \texttt{[6:50-6:58:30)} \\ s \rightarrow n \rightarrow e, & \texttt{if} \quad l \in \texttt{[6:58:30-7:03:26)} \\ s \rightarrow e, & \texttt{if} \quad l \in \texttt{[7:03:26-7:05]} \end{cases}$$

Finally, the boundary-node estimator, which is a lower-bound travel time from n_i to e and is used to improve the efficiency of the algorithm, is described below.

- Partition the space into non-overlapping cells [2]. A **boundary node** [6] of a cell is a node directly connected with some other node in a different cell. That is, any path linking a node in a cell C_1 with some node in a different cell C_2 must go through at least two boundary nodes, one in C_1 and one in C_2.
- For each pair of cells, (C_1, C_2), pre-compute the fastest travel time (function) from each boundary node in C_1 to each boundary node in C_2.
- For each node inside a cell, pre-compute the fastest travel time from and to each boundary node.
- At query time, n_i and e are given. Since any path from n_i to e must go through some boundary node in the cell of n_i and through some boundary node in the cell of e, a lower-bound estimator can be derived as the summation of three parts: (1) the fastest time from n_i to its nearest boundary node, (2) the fastest time from some boundary node in n_i's cell to some boundary node in e's cell, and (3) the fastest time from e's nearest boundary node to e.

Key Applications

The key application of fastest-path computation is road navigation systems. Some examples include Mapquest.com, Local.live.com, and MapPoint. Such systems can produce better driving directions if integrated with traffic patterns and fastest-path computation techniques.

Future Directions

To speed up the calculation, the road network should be partitioned. At the index level, each partition is treated as a single network node and the details within each partition are omitted. Pre-computation is performed to calculate the travel time from each input edge to each output edge. The partitioning can be performed hierarchically. Another direction is that, to simplify the computed fastest paths, the algorithm should be extended to allow the users to provide a maximum number of changes in road names.

Cross References

► Dynamic Travel Time Maps
► Routing Vehicles, Algorithms
► Trip Planning Queries in Road Network Databases

Recommended Reading

1. Chabini, I.: Discrete Dynamic Shortest Path Problems in Transportation Applications. Trans Res Rec **1645**, 170–175 (1998)
2. Chakka, V.P., Everspaugh, A., Patel, J.M.: Indexing Large Trajectory Data Sets With SETI. In: Biennial Conf. on Innovative Data Systems Research (CIDR), Asilomar, CA, USA 2003
3. Huang, Y., Jing, N., Rundensteiner, E.: Spatial Joins Using R-trees: Breadth-First Traversal with Global Optimizations. In: VLDB, Athens, Greece, pp. 396–405 (1997)
4. Jiang, B.: I/O-Efficiency of Shortest Path Algorithms: An Analysis. In: ICDE, pp. 12–19, Tempe, AZ, USA, 1992. IEEE Computer Society (1992)
5. Jing, N., Huang, Y.W., Rundensteiner, E.A.: Hierarchical Encoded Path Views for Path Query Processing: An Optimal Model and Its Performance Evaluation. TKDE **10**(3):409–432 (1998)
6. Kamoun, F., Kleinrock, L.: Hierarchical Routing for Large Networks: Performance Evaluation and Optimization. Comput Netw **1**:155–174 (1977)
7. Kanoulas, E., Du, Y., Xia, T., Zhang, D.: Finding Fastest Paths on A Road Network with Speed Patterns. ICDE, Atlanta, GA, USA, 2006. IEEE Computer Society (2006)
8. Kung, R.M., Hanson, E.N., Ioannidis, Y.E., Sellis, T.K., Shapiro, L.D., Stonebraker, M.: Heuristic Search in Data Base Systems. In: Expert Database Systems Workshop (EDS), pp. 537–548 (1984)
9. Nachtigall, K.: Time depending shortest-path problems with applications to railway networks. Eur J Oper Res **83**:154–166 (1995)
10. Orda, A., Rom, R.: Minimum Weight Paths in Time-Dependent Networks. In: Networks: An International Journal **21** (1991)
11. Pallottino, S., Scutellà, M.G.: Shortest Path Algorithms in Transportation Models: Classical and Innovative Aspects. In: Marcotte, P., Nguyen, S. (ed.) Equilibrium and Advanced Transportation Modelling, pp. 245–281. Kluwer Academic Publishers, New York (1998)
12. Shekhar, S., Fetterer, A., Goyal, B.: Materialization Trade-Offs in Hierarchical Shortest Path Algorithms. In: SSTD, pp. 94–111. Springer, Berlin (1997)
13. Shekhar, S., Kohli, A., Coyle, M.: Path Computation Algorithms for Advanced Traveller Information System (ATIS). In: ICDE, Vienna, Austria, 1993. IEEE Computer Society, pp. 31–39 (1993)
14. Sung, K., Bell, M.G.H., Seong, M., Park, S.: Shortest paths in a network with time-dependent flow speeds. Eur J Oper Res **121**(1), 32–39 (2000)
15. Zhao, J.L., Zaki, A.: Spatial Data Traversal in Road Map Databases: A Graph Indexing Approach. In: Proc. of Int. Conf. on Information and Knowledge Management (CIKM), Gaithersburg, MD, USA, 1994. ACM, pp. 355–362 (1994)

Fastest Route

► Fastest-Path Computation

FCC 94-102

► Indoor Localization

Feature Catalogue

JEAN BRODEUR[1], THIERRY BADARD[2]
[1] Center for Topographic Information, Natural Resources Canada, Sherbrooke, QC, Canada
[2] Center for Research in Geomatics (CRG), Department of Geomatic Science, Université Laval, Quebec, Canada

Synonyms

Content metadata; Abstract representation of geographic data; Machine readable geographic data

Definition

In the context of geographic information and ISO/TC 211 vocabulary, a feature catalogue refers to a description of an abstraction of reality that may be used to depict one or more geographic datasets. Consequently, it constitutes a classification of phenomena [3] and may be used along with an application schema.

Main Text

A feature catalogue consists of a collection of metadata that provides the semantics and the structure of the objects stored in a geographic database. A feature catalogue includes (1) the names and definitions of feature types, (2) their properties' name and definition including feature attributes, geometry (shapes and specifications, datum, map projection, etc.), temporality (dimensions and specifications, datum, units, resolutions, etc.), operations, and roles, (3) descriptions of attribute values and domains, relationships, constraints, and so on. A feature catalogue may be presented in various forms such a text document, a database, a spreadsheet, etc. Typically, a feature catalogue is available in electronic form to support interoperability of geographic information. Although a feature catalogue addresses the same content of an application schema, they are both complementary in the manner they represent it. Levels of details regarding catalogues (data dictionaries) and schemata (models) are described in the cross-references.

Cross References

► Modeling with ISO 191xx Standards
► Modeling with Pictogrammic Languages

Recommended Reading

1. Brodeur, J., Bédard, Y., Proulx, M.J.: Modelling Geospatial Application Databases using UML-based Repositories Aligned with International Standards in Geomatics. 8th ACM Symposium on Advances in Geographic Information Systems (ACMGIS) (2000)
2. ISO/TC211 ISO19109:2005 Geographic information – Rules for application schema. ISO (2005)
3. ISO/TC211 ISO19110:2005 Geographic Information – Methodology for feature cataloguing. ISO (2005)

Feature Extraction

► Image Mining, Spatial

Feature Extraction, Abstract

Ashok Samal, Sharad Seth
Department of Computer Science and Engineering,
University of Nebraska-Lincoln, Lincoln, NE, USA

Synonyms

Automated map generalization; High-level features; Functional description; Abstract features

Definition

Extraction of features from remotely sensed imagery is an important problem in many application domains. Current technologies for feature extraction for urban scenes rely on physical attributes as the basis for classification. Roads, railroads, rivers, buildings, center pivots, lakes, reservoirs, airports, and canals are typical examples of such features. Although this approach yields important information from imagery, low-level techniques to information extraction, classification and management limit both the analysis model and the quality and quantity of information extracted from the image. They do not yield to a higher-level analysis, which allows for a true understanding of the role that individual and groups of features play in processes akin to human cognition. The reason for their failure is the gap between the representation used for features and the models used for cognitive analyses. The current models use individual physical entities as the basic unit for analysis, but the processes that shape significant events on the earth's surface are affected by many competing and coordinating entities that form the basic building blocks for analysis. These entities, therefore, can be viewed as abstract features and must be used to obtain a deeper understanding of geographic events in both space and time. A high-level feature is defined by the characteristic distribution of its constituent features and also by its relationship with other high and low level features – its geographic context. Figure 1 shows an example of such abstract features.

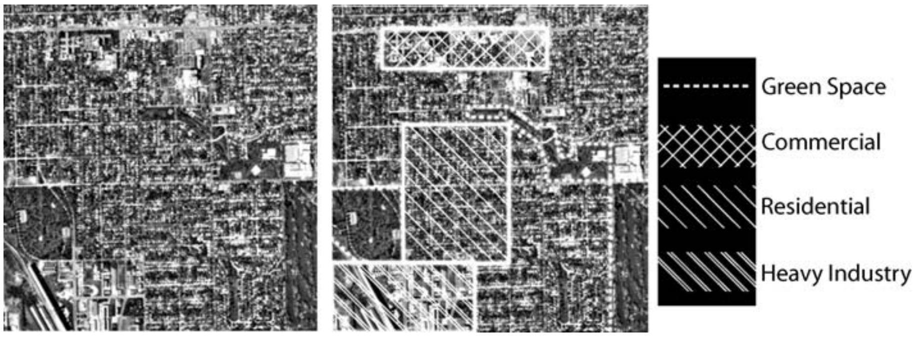
There is a need to extend the classification model into one that more closely parallels the human cognitive system. By organizing features at a higher level than simple physical location, shape and size at a single point in time and space, one can derive a functional description of the geographic space.

Historical Background

Satellite imagery and aerial photograph interpretation has been widely used over the past three decades. It was initially used in military applications for reconnaissance purposes. With the increasing availability and the decreasing cost of such sources of data, it has migrated to many civilian applications, e. g., generations of digital maps, map planning, and surveys of land use and land cover. These remotely sensed images, ranging from the 1-m high-resolution panchromatic images provided by IKONOS and QuickBird to 2.5-m medium-resolution panchromatic images provided by SPOT 5 and the 15-m low-resolution panchromatic images provided by Landsat 7 have become one of the most pervasive sources of spatial data used in the GIS community.

Before their incorporation in GIS applications, images should be delineated and classified to predefined patterns and objects that can be used to build GIS topology useful for analysis, modeling, map making or 3D terrain visualizations. Often, interpretation is manually performed by experienced photo interpreters. Due to the inherent inefficiency in manual interpretation and the limited availability of qualified operators, many images remain unprocessed. This problem will only exacerbate as more high-resolution satellites are launched in the coming years and as other remotely sensed images become available and affordable.

In order to improve the efficiency of interpretation, many fully or partially automated processes have been developed



Feature Extraction, Abstract, Figure 1 USGS Digital Orthophoto Quarterquadrangle (1:12000) Lincoln, Nebraska and some possible abstract features

with their focus dependent on image resolution. For high-resolution images of sub-meter resolution, the emphasis is on extracting individual buildings or the road network in a standard two-step process [2,5,6,8,9,10].

1. Line segments deemed as parts of roads or buildings are extracted by tracking the extreme value of local gradient or the energy of the image intensity.
2. Extracted segments are conceptually grouped according to a road or building model.

For low-resolution images, usually greater than ten meters, the focus has been on methods to extract the boundary of regions that have specific spectral properties or texture patterns. The image segmentation for land cover and land use could be divided into edge-based and region growing approaches. The most common application of such analysis is in drawing the land use or land cover thematic map. There is relatively little research with focus on an integrated approach that seamlessly bridges the gap between the processing of high and low resolution images. In other words, there is no framework to interpret the internal structure of a land cover. For example, in an urban area, one not only would want to find the coverage in terms of residential or commercial areas, but may further want to know whether the covered area is a single family residence or a multi-family residence, or a general office. Better yet, one may want to find the geographical composition of a region, i. e., by building the spatial relation of the blocks within the region. Such a representation of the image has many useful applications including understanding the urban sprawl process, planning the utility locations, managing traffic, and evaluating the loss due to a hazard.

One trend in GIS is to develop an ontology-driven GIS [3], which means the elements of the GIS are not the usual layers of points, lines and polygons, but objects that denote location information, spatial relations, associated events and interfaces between objects. Ontology-driven GIS will facilitate recording or simulating the evolution of an area in a period of time. This proposal for image interpretation fits ithis scheme by providing the spatial description of the ontology.

The focus of the paper is limited to the urban landscape for two reasons. First, urban areas include a large variety of regions occurring in complex configurations that are challenging to interpret. Second, being the result of human activities, urban areas have a certain degree of regularity in the arrangement of spatial objects that can be helpful in geographic model building and automated feature interpretation.

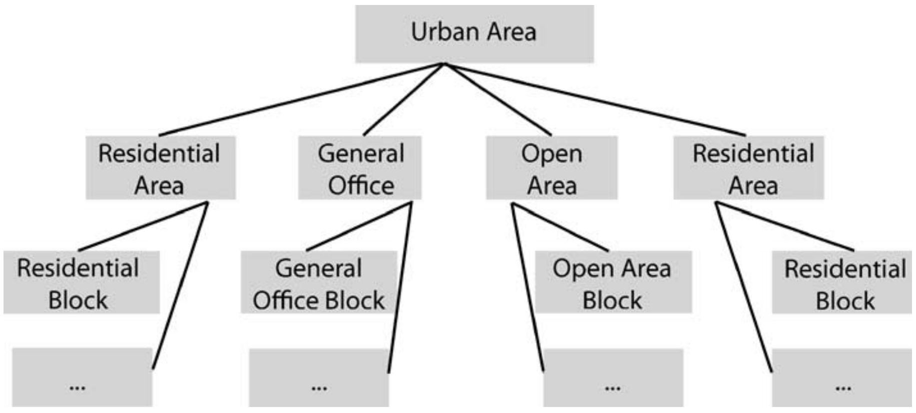
Further, because extensive work on the extraction of roads and buildings already exists, the assumption is that this level of image interpretation is available in vectorized form. The focus is, instead, on analyzing the structure of an urban area.

Human interpreters deduce land use by means of various clues, such as color, texture, shape, size, and the spatial relationship between adjacent objects visible in the image. All these clues of a local region will be denoted as its *context*. While context has been used in traditional interpretation methods to recognize individual buildings, presently it is used for a higher level of image interpretation, e. g., to find a cluster of buildings that have a specific function.

Scientific Fundamentals

Abstract features are rich, diverse, spatially disjoint, temporally correlated and inherently fuzzy. These properties impart tremendous power and pose great challenges to abstract feature extraction. The properties of abstract features are briefly described below.

- The difference between physical and abstract features is not just a matter of scale. In general, abstract features will be defined at broader scales in both time and space. For example, an abstract feature may contain disjoint geographic areas that may display particular characteristics.
- Abstract features are domain dependent. Since the abstract features are tied to concepts, they are defined in the context of an application domain.
- The constituents of an abstract feature may not be spatially contiguous. The power of abstract features is



Feature Extraction, Abstract, Figure 2 The hierarchical composition tree of an urban area

derived from the ability to connect spatially disjoint, but conceptually similar physical features.

- Abstract features are dependent on the temporal dimension. Time is a critical factor in the definition and hence, abstract features can not be obtained by static analysis.
- Abstract features are fuzzy. Processes affecting events on the surface of the Earth can not be characterized with parameters that are certain or hard. Thus, a faithful representation of abstract features must incorporate some uncertainty.

Attributes of abstract features are described in a variety of sources and conflation is an integral part of their extraction. In addition to remotely sensed images, a variety of ancillary data may be used to increase the efficiency and accuracy of the feature extraction process. By drawing upon multiple sources, a more complete description of features may be generated.

The focus of this article is extraction of abstract features for urban areas. The basic element in this analysis is the block, defined as the smallest region that cannot be divided by any road segments. A block includes both its surrounding roads and the buildings inside. In this approach, the category of a block is first determined by checking its global context. If it is a mixed block, it is broken into sub blocks based on the consistency of the global context. Conversely, if adjacent blocks have been determined to be of the same category, they are merged. At the end of the interpretation process, a hierarchical composition tree of an urban area similar to the one shown in Fig. 2 is derived.

Interpretation of the image of the urban area is done in three steps in this approach:

1. Define the classification system for the land cover and land use in the urban area
2. Determine the global context for each category in the classification system
3. Design an algorithm for recognizing each category.

The classification system in remote sensing is organized in a 4-level hierarchy [1]. Levels I and II are generally of interest to users who desire data on a nationwide, interstate or statewide basis. Levels III and IV are used by those concerned about information at the regional, county, or municipal level. USGS has standardized the classification of Levels I and II [1].

Levels III and IV classifications consider the remotely sensed image in more detail and require a substantial amount of supplemental information in addition to the image data. This is one reason for the lack of standardization at these levels. Some examples of Levels III and IV classifications used by local government have been described in the literature [4]. However, these classifications emphasize more on functions of the region that are not available from the remotely sensed imagery. Since the focus here is to only extract the information from the imagery, the classification system has been revised by merging classes that have the same visual appearance but have different functions. For example, commercial office building and public service building categories have been combined into a general office building category. It should be noted that use of ancillary data can only improve this analysis.

To transfer the function directed classification system to an appearance directed system, the boundary of Level II classification is broken and the Level III classification is reorganized. For example, Class 1, described below, corresponds to Category 14 in Level II, Classes 2, 3, and 4 belong to Category 11, and Class 5 matches with Category 12, but Class 9 is a combination of some subclasses of Categories 13 and 15. The classes in the scheme are described below.

1. *Aviation Area (AA)*: The area which is used or is intended to be used, primarily for the take-off and landing of aircraft and any appurtenant areas which are used, or intended to be used, for airport buildings or facilities, including open spaces, taxiways and tie-down areas, hangers, and other accessory buildings. The indication of Aviation Area is the run-

way, and the vast area of grassland around the runway.

2. *Single Family Residence (SFR)*: The area which is composed of lots where each one has one building designed for and contains not more than two separate units with facilities for living, sleeping, cooking, and eating therein. The buildings in this category are of equal size and have a regular layout.
3. *Multi-Family Residence (MFR)*: The area that is a cluster of dwelling units that supports multi-families in a residential setting. Buildings in this area are larger than those in *SFR* and are surrounded by a sufficient parking place. *Multi-Family Residenc*es are always close to *Single Family Residence*s.
4. *Mobile Home Residence (MHR)*: The area that is a cluster of transportable structures built on a chassis and is designed, when connected to the required utilities, to be used as a year-round dwelling unit with or without a permanent foundation. Buildings in *Mobile Home Residence* are smaller than those in *SFR* and the layout is more compact.
5. *Commercial Business District (CBD)*: The area that is the concentration of financial and professional services and major retail outlets, the focus of transport lines, where land use is the most dense and land values are at their highest. Most of this area is covered by man-made objects and it always appears in the center of the city.
6. *Shopping Complex (SC)*: The area used for mercantile establishment consisting of a carefully landscaped complex of shops representing leading merchandisers. *Shopping Complex* is an agglomeration of several large buildings surrounded by a large parking lot.
7. *Common Commercial Service and Office (CCSO)*: The area used for commercial or industrial activities that typically have operating characteristics or traffic service requirements generally incompatible with residential environments, such as equipment sales, custom manufacturing, vehicle storage, or construction services. Buildings in this category are not as big as those in the *Shopping Complex* category, but larger than those in residential and spread along the main streets. The layout of entities in this category is not as compact as those in *Commercial Business District*.
8. *Open Area (OA)*: The area that has very few man-made objects mostly consists of bare land, land covered by vegetation, or bodies of water.
9. *Industrial District (ID)*: The area that is intended for commercial services, research and development, administrative facilities, warehousing and distributions, and manufacturing uses. Buildings in the *Industrial District* are big, but have little space for parking, and this category always appears at the outskirt of the city.

In the definition of each class, some of the spatial characteristics have been described. However, these descriptions of geometric, spatial and other properties should be formalized so that they can be used in an automated system. These properties for a specific scene are called its context or sometimes its signature. The context defines the rules for combining adjacent buildings into a community unit with a specific social function. Context is used as the basis for recognition.

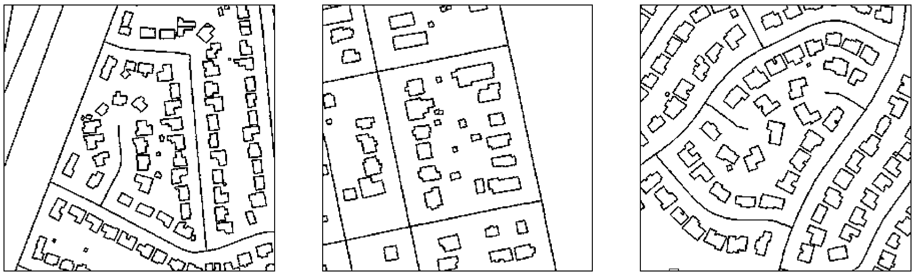
The focus here is on the *Single Family Residence* (*SFR*). Figure 3 presents several examples of the *SFR*, from which many important properties can be extracted.

By analyzing the structure of the blocks in the *SFR*, several constraints that can be used as a global context or signature for the *SFR* are derived. Two kinds of constraints are considered: *Scalar* Constraints and *Spatial* Constraints. Scalar constraints describe properties without geometric information like dimension and size. Spatial constraints characterize direction and location relationships between the adjacent objects and their layout of the full block. Examples of each type of constraint follow.

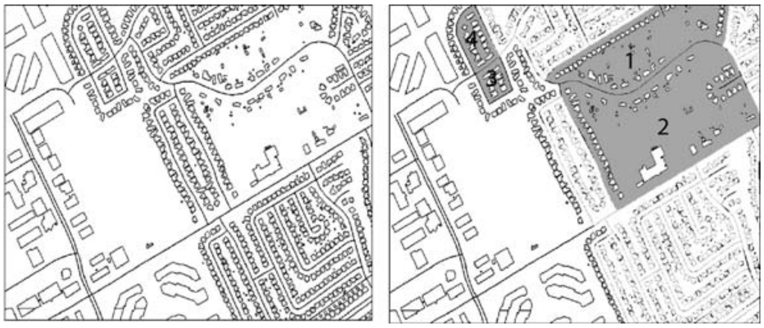
1. *Spatial Constraints*
   a. The *SFR* buildings are approximately at the same distance to their nearest road segments.
   b. The buildings typically are linearly arranged and the distances between buildings are approximately the same.
   c. One of the dimensions of a building is aligned with the nearest road segment.
2. *Scalar Constraints*
   a. The area and dimensions of buildings in a *SFR* block are similar.
   b. The residential buildings in a *SFR* block typically cover only a small part of the total area of the block.

The following algorithm describes the approach to determining if a block is a *SFR* block.

1. Calculate the area of all the buildings in the block.
2. If *(buildings area)/(block area)* > *th1* (a predefined but adaptable threshold), then the block is rejected as an *SFR* and exit.
3. Enumerate all the buildings whose area falls in a predefined range [*t1*, *t2*]. (*t1* and *t2* are derived from statistical analysis of given *SFR* examples).
4. Find the bounding rectangles for all buildings that may be single family housing units. Link the center of these bounding rectangles by following along the road boundary.
5. Find all single family building chains by analyzing the distance to the nearby road network, the distance to adjacent buildings and the direction of the build-



Feature Extraction, Abstract, Figure 3 Example composed primarily of a Single Family Residential Area (SFR)



Feature Extraction, Abstract, Figure 4 An urban scene (*left*) and its interpretation using only scalar constraints (*right*)

ings. A building that satisfies three spatial constraints declared in the previous paragraph will be seen as a candidate for a single family house unit. Start a chain by choosing a candidate randomly and track along the boundary road. If the next building is a candidate, the chain continues, otherwise the chain stops and a new chain starts when the next candidate is found in the track.

6. Project the centers of the two end buildings to the boundary and the boundary between these two projected points is defined as SF-Chain boundary. Calculate the length of the SF-Chain boundary.
7. If Σ (*length of SF-Chain boundary*)/(*length of the whole boundary*) > *th2* (another predetermined threshold), then this block is classified as an *SFR*.
8. Otherwise, this block is classified as a MIXED block.
9. Merge all adjacent *SFR* blocks.

Another approach for building generalization for urban scenes using morphology and Gestalt theory has been proposed by Li, et al. [7].

A Case Study

To show the feasibility and the accuracy of the context directed interpretation, it has been tested with the GIS data of Austin, TX. The population of this city is approximately 660,000 and it covers about 260 square miles. It has 277,000 housing units and is the 16th largest city in the US.

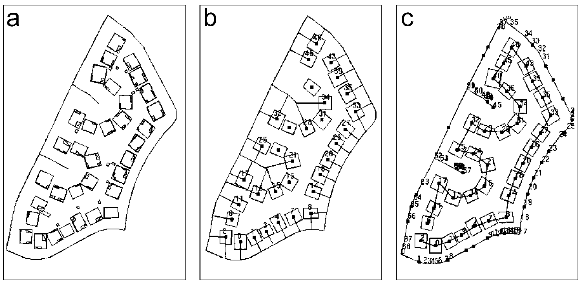
Only some sections of the city are used for testing and reported here. Figure 4 (left) shows part of the dataset used for testing. The long linear structures represent the road network and the small polygons are the footprints of the buildings. The dataset consists of a *Single Family Residence* adjacent to a *General Office* or a *Multi-Family Residence*. Figure 4 (right) shows the results of the interpretation using only two scalar constraints:

1. The building coverage of the block is between 0.15 and 0.35
2. The average size of the building is between 1200 and 3000 square units.

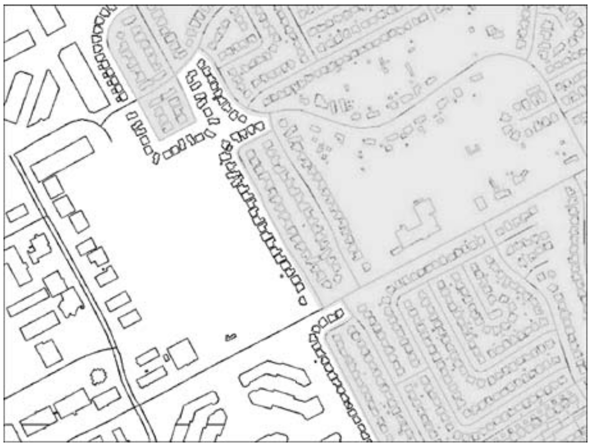
The thresholds are determined manually by studying the properties of several blocks. They can also be extracted by unsupervised methods.

Of all the 33 blocks in Fig. 4, only four SFR blocks (1, 2, 3 and 4) are misclassified. The shaded regions represent the correctly recognized Single Family Residence blocks. The reason for misclassification of Blocks 1 and 2 is that the building density is too low and the misclassification of Block 3 and 4 is due to the high building density. This is a disadvantage of using scalar constraints, because universal thresholds are difficult to accurately determine.

Figure 5 shows several steps during the interpretation using both scalar and spatial constraints. Figure 5a shows the result of finding the bounding box of the building footprints. Figure 5b shows the principal direction of each building; Buildings less than 1200 square units are filtered out. Figure 5c shows the chains of buildings around the boundary of the block. Figure 6 shows the final interpretation. It can be seen that the Blocks (1, 2, 3 and 4), misclassified by the previous algorithm, are interpreted correctly when spatial constraints are used.



Feature Extraction, Abstract, Figure 5 Steps of interpretation using the spatial constraints: **a** computing bounding rectangles, **b** principal directions, **c** finding building chains



Feature Extraction, Abstract, Figure 6 Result of interpretation using both scalar and spatial context constraints

Key Applications

Tactical Intelligence

In automated target recognition it is often critical to identify high value targets while reducing the risk of collateral damage. Depending on the mission, the goal is to minimize areas which are likely to have high population density. In such cases, it is important to identify the housing areas (single and multi-family) and shopping complexes during the high traffic times. Automated identification of such high-level features from information derived from remotely sensed imagery will likely be critical.

Urban Planning

Most of the urban centers in the world continue to see significant growth in population. Even smaller towns in rural areas near large metropolitan centers have seen growth in recent years. The growth in many cases is not coordinated with a master plan. Use of the approach described here, will allow the monitoring of the growth of these areas which is critical for planning and inventory.

Advanced Geo-Search Engines

Traditional text-based search engines are increasingly being augmented with geographic information. Satellite imagery is now available online for any part of the world, albeit with varying resolution. Extraction of abstract feature would allow the geo-searches to look for specific geographic features in satellite images. For example, an avid golfer may search for an abstract feature that consists of a golf course adjacent to an airport using the approach described here.

Crime Mapping

Law enforcement data in urban areas often show that some types of criminal activities are prevalent in a certain configuration of spatial features, such as, large downtown blocks with large parking lots and with easy access to a highway. One can define an abstract feature to capture this spatial configuration and identify it for better crime monitoring and alert.

Future Directions

As the amount of satellite–based and other remotely sensed imagery becomes even more available and methods to extract lower level features such as buildings and roads become more robust, there will be an increase in the interest to extract higher level features such as those described here. The focus is likely to change from deriving lower level to higher level feature extraction and from a structural description to a functional description of the scenes. Techniques that use ancillary data (e. g., census data, other GIS layers) to improve the efficiency and accuracy of feature extraction will become more feasible and critical.

Cross References

- ► Abstraction of GeoDatabases
- ► Crime Mapping and Analysis
- ► Geospatial Semantic Integration
- ► Patterns, Complex

Recommended Reading

1. Anderson, J.R., Hardy, E.E., Roach, J.T., Witmer, R.E.: A Land Use and Land Cover Classification System For Use With Remote Sensor Data. USGS Professional Paper **964** (1976)
2. Barzoharand, M., Cooper, D.B.: Automatic Finding of Main Roads in Aerial Images by Using Geometric Stochastic Models and Estimation. IEEE Transaction on Pattern. Anal. Mach. Intell. **18**(7), 707–721 (1996)
3. Fonseca, F., Egenhofer, M.: Ontology-Driven Geographic Information Systems. In: 7th ACM Symposium on Advances in Geographic Information Systems, Kansas City, MO, pp. 14–19. (1999)

4. Garry, S.: Ohio Land Use Land Cover Classification System. Remote Sensing Program Miscellaneous Report No. 17, Fifth Edition (1988)
5. Gruen, A., Li, H.: Semi-automatic Linear Feature Extraction by Dynamic Programming and LSB-Snake. Photogrammetric Eng. Remote Sensing **63**(8), 985–995 (1997)
6. Huertas, A., Lin, C., C., Nevatia, R.: Detection of Building form Monocular View of Aerial Scenes using Perceptual Grouping and Shadows. In: Proceeding DARPA Image Understanding Workshop, Washington, DC, pp. 253–260 (1993)
7. Li, Z., Yan, H., Ai, T., Chen, J.: Automated building generalization based on urban morphology and Gestalt theory. Int. J. Geogr. Inf. Sci. **18**(5), 513–534 (2004)
8. Lin, C., Nevatia, R.: Building Detection and Description from Monocular Aerial Images, Proceeding DARPA Image Understanding Workshop, Palm Springs, CA, pp. 461–468 (1996)
9. McKeown, D.M., Pane, J.F.: Alignment and Connection of Fragmented Linear Feature in Aerial Imagery. In: Proceedings of 1985 IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, pp. 55–61 (1985)
10. Wang, J., Zhang, Q.: Applicability of a Gradient Profile Algorithm for Road Network Extraction – Sensor Resolution and Background Consideration. Canadian J. Remote Sensing **26**(5), 428–439 (2000)

Feature Matching

► Conflation of Features

Features

► Geography Markup Language (GML)

Features, Linguistic

► Geospatial Semantic Integration

Features, Physical

► Geospatial Semantic Integration

FGDC

► Spatial Data Transfer Standard (SDTS)

Field Data

► Oracle Spatial, Raster Data

Filter and Refine Strategy

JORDAN WOOD
Department of Computer Science,
University of Minnesota, Minneapolis, MN, USA

Synonyms

Filter-Refine Paradigm

Definition

The filter and refine strategy is a general method for completing a computationally intensive task as quickly as possible. In some tasks, such as a spatial join, we need to do a computationally intensive step on only a small subset of the original data. In order to find this subset, we first apply a filter step which removes data that are not involved in our task. We then do the refining step on the remaining data.

Main Text

When we are using the filter and refine strategy with spatial data, we generally filter the data by running the algorithm in question on simplified representations of the spatial objects. Minimum bounding rectangles (MBRs) are commonly used for this purpose. The point of the filter step is not to reduce the data to only that data needed to find the answer to the problem, but rather to remove a significant part of the uninvolved data in a computationally efficient manner. For example, when trying to detect overlap present in a set of polygons, we might remove more data in the filter step if, for every polygon, we approximate using the smallest quadrilateral possible which contains the entire original polygon. However, if we impose the restriction that the quadrilateral must be a rectangle with sides which are parallel to the x and y axes, we can simplify both the algorithm which calculates the rectangles and the algorithm which performs the filter step.We therefore trade some computation in the refine step for potentially greater computation time savings in the filter step.

Cross References

► Indexing Spatial Constraint Databases
► Minimum Bounding Rectangle

Recommended Reading

1. Shekhar, S., Chawla, S.: Spatial Databases, A Tour, Pearson Education, Inc., New Jersey, USA (2003)
2. Orenstein, J.A., Spatial query processing in an object-oriented database system, Proceedings of the 1986 ACM SIGMOD international conference on Management data, p. 326–336, May 28–30, Washington, D.C., United States (1986)

Filtering

► Hierarchies and Level of Detail

Filter-Refine Paradigm

► Filter and Refine Strategy

FIPS 173

► Spatial Data Transfer Standard (SDTS)

FIPS PUB 173

► Spatial Data Transfer Standard (SDTS)

First Law of Geography

► Crime Mapping and Analysis

First-Order Logic with Constraints Queries

► Linear Versus Polynomial Constraint Databases

Fleet Management

► Routing Vehicles, Algorithms

Floating Car Data

DIETER PFOSER
RA Computer Technology Institute, Athens, Greece

Synonyms

Probe vehicle data; PVD; Vehicle tracking data

Definition

Floating car data (FCD) refers to using data generated by one vehicle as a sample to assess the overall traffic condition ("cork swimming in the river"). Typically this data comprises basic vehicle telemetry such as speed, direction and, most importantly, the position of the vehicle.

Main Text

The over-time-collected positional data component of FCD is referred to as *vehicle tracking data*. FCD can be obtained by tracking using either GPS devices or mobile phones. The latter type results in low-accuracy data. Depending on the collection method and data accuracy, more or less sophisticated map-matching algorithms have to be used to relate tracking data to a road network.

In database terms, the tracking data can be modeled in terms of a *trajectory*, which is obtained by interpolating the position samples. Typically, linear interpolation is used as opposed to other methods, such as polynomial splines. The sampled positions then become the endpoints of line segments and the movement of an object is represented by an entire polyline in 3D space.

FCD is a powerful means to assess traffic conditions in urban areas given that a large number of vehicles collect such data. Typical vehicle fleets comprise taxis, public transport vehicles, utility vehicles, but also private vehicles.

Cross References

► Dynamic Travel Time Maps
► Map-Matching

Recommended Reading

1. Schaefer, R.-P., Thiessenhusen, K.-U., Wagner, P.: A traffic information system by means of real-time floating-car data. In: Proc. ITS World Congress, Chicago (2002)

Flocking

► Movement Patterns in Spatio-temporal Data

Folksonomy

► Uncertainty, Semantic

Footprint

► Retrieval Algorithms, Spatial

Format

► Geography Markup Language (GML)

Fourier Series

► Constraint Databases and Data Interpolation

Four-Intersection Model

► Dimensionally Extended Nine-Intersection Model (DE-9IM)

Frame of Discernment

► Computing Fitness of Use of Geospatial Datasets

Free GIS

► MapWindow GIS

Frequent Itemset Discovery

MARKO SALMENKIVI
Department of Computer Science,
University of Helsinki, Helsinki, Finland

Synonyms

Market-basket analysis

Definition

Consider the set of all products sold by a supermarket. Assume that the owner of the supermarker is interested in finding out subsets of products that are often purchased together. Each customer transaction is stored in a transaction database, indicating the products that the customer purchased together. The database can be described as a table, whose columns are the products (items), and the rows are the transactions. The value of a specific entry, that is, (row, column)-pair, in the table is 1 if the corresponding product was purchased in the transaction, and 0 otherwise. The task is to find itemsets such that the items frequently occur in the same row (products purchased together). The most important interestingness measure in frequent itemset mining is *support* of an itemset. It is defined as the fraction of rows of the database that contain all the items $x \in X$. An itemset is *frequent* if its support exceeds a user-specified threshold value.

Association rules are a closely related pattern class. Let R be a set of products, r a transaction database, and $X, Y \subseteq R$ itemsets. Then $X \to Y$ is an association rule over r. The interestingness of an association rule $X \to Y$ is usually measured by its support defined as $support(X \to Y) = support(X \cup Y)$, and confidence: $conf(X \to Y) = \frac{support(X \cup Y, r)}{support(X, r)}$. Thus, the confidence is the conditional probability that a randomly chosen row from r that contain the items in X also contain the items in Y. Given thresholds for support and confidence, the association rule mining task in a given database is to find all the rules, whose supports and confidences exceed the thresholds.

Main Text

One of the first significant contributions of data mining research was the notion of association rule. All interesting association rules from a relational database can be found by solving a subtask known as frequent itemset discovery, often called market-basket analysis.

The name market-basket analysis originates with the domain of the analysis of customer behavior in supermarkets, which was the starting point of the methodology in the late 1980's. The most well-known algorithm for finding all frequent itemsets is Apriori introduced in 1994 [1]. Many algorithms that are more efficient than Apriori have been introduced later (e. g., [4]). It was soon realized that frequent itemset discovery often resulted in very large number of frequent itemsets and confident association rules. Recent research has concentrated on developing condensed representations (closed sets, non-derivable sets, non-derivable association rules) for the set of interesting patterns. The basic idea is that usually in practice a large part of the interesting patterns are redundant in the sense that they can be deduced from a subset of all frequent patterns [2,3]. This leads to a new interestingness measure: a pattern is interesting only if its support (and/or confidence) cannot be derived from the supports (and/or confidences) of its subpatterns.

Cross References

► Co-location Pattern
► Co-location Pattern Discovery
► Frequent Pattern

Recommended Reading

1. Agrawal, R., Ramakrishnan, S.: Fast algorithms for mining association rules in large databases. Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499, September 12–15 (1994)
2. Goethals, B., Muhonen, J., Toivonen, H.: Mining non-derivable association rules. SIAM International Conference on Data Mining, 239–249, Newport Beach, California (2005)

3. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Proceedings of the International Conference of Database Theory – ICDT '99. Lecture Notes in Computer Science vol. 1540, pp. 398–416 (1999)
4. Zaki, M.: Scalable algorithms for association mining. IEEE Trans Knowl Data Eng **12**(3), 372–390 (2000)

Frequent Itemset Mining

► Patterns, Complex

Frequent Pattern

Marko Salmenkivi
Department of Computer Science,
University of Helsinki, Helsinki, Finland

Synonyms

Interesting pattern; Selected pattern

Definition

Given a set of patterns, i. e., a class of expressions about databases, and a predicate to evaluate whether a database satisfies a pattern, the frequent pattern mining task is to determine which patterns are satisfied by a given database. Formally, assume that $\mathcal{P}$ is a set and q is a predicate $p : \mathcal{P} \times \{\mathbf{r} \mid \mathbf{r} \text{ is a database}\} \to \{\text{true, false}\}$. Elements of $\mathcal{P}$ are called patterns, and q is a selection criterion over $\mathcal{P}$. Given a pattern φ in $\mathcal{P}$ and a database $\mathbf{r}$, φ is selected if $q(\varphi, \mathbf{r})$ is true. Given a database $\mathbf{r}$, the theory $\mathcal{T}/\mathcal{P}, \mathbf{r}, q)$ of $\mathbf{r}$ with respect to $\mathcal{P}$ and q is $\mathcal{T}/\mathcal{P}, \mathbf{r}, q) = \{\phi \in \mathcal{P} | q(\phi, \mathbf{r}) \text{is true}\}$.

Main Text

It is important to note that the interpretation of "$q(\varphi,\mathbf{r})$ is true" depends on the application domain. For some applications it could mean that φ occurs often enough in $\mathbf{r}$, that φ is true or almost true in $\mathbf{r}$, or that φ defines, in some way, an interesting property of subgroup of $\mathbf{r}$.

For computational purposes, a very important property of the selection criterion (i. e., interestingness measure) is *monotonicity*. Let $\preceq$ be a partial order on the patterns in $\mathcal{P}$. If for all databases $\mathbf{r}$ and patterns $\varphi, \theta \in \mathcal{P}$ it holds that $q(\varphi, r)$ and $\theta \preceq \varphi$ imply $q(\theta, \mathbf{r})$, then $\preceq$ is a *specialization relation* on $\mathcal{P}$ with respect to q. Then q is monotonous with respect to the pattern class and the specialization relation. If the monotonicity holds, levelwise algorithms (such as Apriori-algorithm [1]) can be used to find all interesting patterns.

For instance, in market-basket analysis (frequent itemset discovery) a natural way to define the specialization relation is the size of the itemset. Here the number of items in itemset φ is denoted by $|\varphi|$, and $\theta \preceq \varphi$ if and only if $|\theta| \geq |\varphi|$. Clearly, the support of φ is monotonous with respect to the size of the pattern. Similarly, in the case of the co-location patterns, the definition of specialization relation can be based on the number of features in the patterns. Participation ratio and prevalence, for instance, are monotonous with respect to the specialization relation.

Cross References

► Co-location Pattern
► Co-location Pattern Discovery
► Co-location Patterns, Interestingness Measures
► Frequent Itemset Discovery

Recommended Reading

1. Agrawal, R., Ramakrishnan, S.: Fast algorithms for mining association rules in large databases. Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499, September 12–15 (1994)
2. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Mining Knowl Disc **1**(3), 241–258 (1997)

Functional Description

► Feature Extraction, Abstract

Fundamental Matrix

► Photogrammetric Methods

Fuzzy Sets

► Objects with Broad Boundaries