Editing and Versioning Dynamic Network Models

Petko Bakalov, Erik Hoel, Wee-Liang Heng,
Environmental Systems Research Institute
Redlands, CA 92373, USA
{pbakalov,ehoel,wheng}@esri.com

Vassilis J. Tsotras
University of California
Riverside, CA 92521, USA
tsotras@cs.ucr.edu

ABSTRACT

Network data models are widely used to describe the connectivity between spatial features in GIS architectures. Recent applications demand that such models are editable in multiuser environments. The preferred method to resolve conflicts in GIS systems is the use of multiple versions of the data to encapsulate the modifications generated by the end users. In this demo we present a flexible versioning scheme for network models. Our solution is based on marking "dirty areas" (regions which contain conflicts between multiple versions) and subsequent cleaning of these dirty areas. We have implemented a prototype of this versioning scheme and have used it extensively to support various ESRI applications which utilize network models.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Spatial databases and GIS; E.1 [**Data Structures**]: Graphs and networks

1. INTRODUCTION

Network data models provide an efficient way to describe the connectivity information among spatial features in GIS [3]. Connectivity for such features is represented by network elements (junctions and edges) in a graph termed "logical network". The type of the network element used to describe a spatial feature in the model depends on the geometry of the feature. Points are represented as junctions while lines are noted as edges. Different versions of the network model have been implemented in existing operational systems like Arc/Info [5] and TransCAD [2]

A typical requirement for network models, is that they must provide support for many users creating and updating large amounts of geographic information. Moreover, this editing environment must support edit sessions that typically span a number of days or weeks, the facility to undo or redo changes made to the data and the ability to develop new models and alternative application designs without affecting the published database.

The first problem to solve is how to make the network model editable and keep the connectivity information persistent with the modifications. We thus first demonstrate an efficient solution which *incrementally* maintains the connectivity between the spatial features while updates occur [1]. We introduce the notion of a "dirty area" inside a network data model which tracks feature edits. This concept has been used previously in topologies [4].

A dirty area is a region inside the network spatial extend where the network features have been modified but the correctness of their connectivity information has not been verified. The network data model is assumed correct only when it is dirty area free. A dirty area is incrementally reduced by a process called *rebuilding*. The end user specifies which portions of the dirty area should be cleaned and the rebuild process analyzes and re-establishes the connectivity information there. Allowing users the ability to rebuild only portions of the dirty area is a practical requirement in scenarios involving very big seamless networks. The user may clean only these portions of the network extend which are of interest to a given application or query, thus avoiding a costly total rebuild.

The second step is to make the network model editable in a multiuser environment. However, traditional DBMS solutions that implement concurrency control using transactions and locking would not be efficient for a GIS multiuser environment. Since edit tasks over the same data may take more than a few minutes to complete, the short transaction row-locking mechanisms adopted by many DBMSs would be prohibitively restrictive for other database users. Instead, versioning approaches have been proposed (and widely used in the ArcGIS system). We will thus demonstrate a new versioning scheme for network models that utilizes further the dirty areas concept introduced by the connectivity rebuild algorithm. When versioning in the system is allowed, users need the ability to combine (merge) versions. Before such merging is applied, the application needs to discover areas/objects between the two versions that have conflicts. Conflicts are then covered by facilitating dirty areas. The incremental rebuild approach will first clean such areas and then the related versions can be merged. Thus the incremental property allows us to focus on the dirty areas and objects than on reconstructing the network from scratch.

2. DISCOVERING VERSION CONFLICTS

In ArcGIS users are allowed to obtain their own alternative, independent, persistent view of the database that does not involve creating a copy of the data. These views are called *versions*. A new version is derived from an existing version (called the *parent* version). The new version is referred as the *child* version and originally contains the same view over the geodata as the parent version. If users want to merge versions, all conflict regions within these versions need first to be identified. We thus introduce four basic rules that are *complete* and *sufficient* in finding all such conflicts.

- Any dirty areas or dirty objects present in the parent or child version that did not exist before the creation of the child ver-

ACM GIS '08 , November 5-7, 2008. Irvine, CA, USA
(c) 2008 ACM ISBN 978-1-60558-323-5/08/11 ...$5.00.

sion will remain dirty as a result of the reconcile process.

- Any dirty area or object in the parent version which has been cleaned in the child version will become dirty as a result for the reconcile process.
- Any dirty area or dirty object introduced and validated in the parent version, whether or not it was present in the child version, will remain validated as a result of the reconcile process.
- Any dirty area or object in the child version will produce a dirty area or object as the result of reconcile, whatever it is validated in the child version.

Those rules depict the fact that during the reconciliation we utilize only one of the associated logical networks - the one on the parent side. This is done in order to simplify the process of version merging.

3. DEMONSTRATION DESCRIPTION

The objective of this demonstration is twofold. First, we want to convey a general introduction to our network model architecture, its components, and their interaction. Second, we want to demonstrate the incremental rebuild functionality as well as the proposed versioning scheme for network models.

3.1 Network model basics

The first part of our demonstration presents the network model, its analysis components (*solvers* and their ability to perform analysis in three environments: (i) Multi-modal transportation systems, (ii) Hierarchical systems and (iii) Systems with complex constraints.

A multi-modal transportation network [6] is a network which includes more than one mode of transportation (for example, freeways and railroads). In such transportation systems, the users can change their transportation mode (drive a car to the train station and then take a train) but such changes can occur only in a set of connectivity points shared by both transportation modes. We are going to demonstrate how the different modes of transportation are captured in our model and how the user can specify the connectivity points between the two modes.

A hierarchical systems is described by a hierarchy of levels; this is something which occurs naturally in the road network where the freeways are typically associated with the highest hierarchy level followed by the highways and the city streets. We are going to demonstrate how this natural hierarchy can be reflected in our model and how the solvers take it into consideration during the analysis.

Finally, we will examine systems with complex constraints like turns, one way moving restrictions, height/weight restrictions [7]. The presence of such turns and one way restrictions in the network can have great impact on the movement inside the network. Nevertheless, our model can easily represent such constraints.

3.2 Editing network models

This part of the demonstration will depict the effects of the feature edits in the network model. We will first demonstrate how simple modifications can make the model inconsistent. Then we will show how changes are tracked in the feature space through the dirty area mechanism. We will demonstrate the concept of dirty objects - an extension to the dirty area mechanism for network features without geometrical properties like the turns (typically turns are defined as a relation between two or more line features and do not have shape). Finally we will demonstrate the efficiency of our incremental rebuild algorithm as well as the UI (see Figure 1) which allows the users to specify portions of the dirty areas for a rebuild.

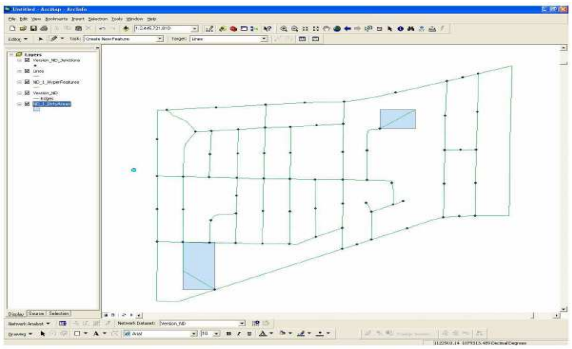

Figure 1: A capture of the rebuild algorithm UI

3.3 Versioning scheme for network models

In the third part of our demonstration we will show how network model modifications are handled in a multiuser environment. We will demonstrate the process of a creation of a new version and how the changes in given version are isolated form the other users. We will also demonstrate how version reconciliation is performed when two versions are merged. During this process the conflict areas are discovered and marked as dirty. Then the cleaning process takes over using the incremental rebuild algorithm leading to a consistent network model that can be used for analysis.

4. CONCLUSIONS

Network models represent an efficient way to describe complex connectivity information among spatial features. Recent applications require the ability to edit the same geographic data simultaneously by many users. Traditional DBMS has not been tailored to meet the special needs that are required in handling large network datasets. In this demonstration we first describe an efficient solution that incrementally maintains network connectivity in the presence of user updates. Our approach facilitates the notions of dirty areas and/or objects, that are then incrementally "cleaned" to re-establish network consistency. On top of the model, we then present a flexible versioning scheme that utilizes dirty areas/objects to reconcile conflicts between versions before merging them. The ideas presented in this demo have been prototyped in the well established ArcGIS development framework and have been shown to provide efficient editing and versioning for large network models.

5. REFERENCES

[1] Petko Bakalov, Erik G. Hoel, Wee-Liang Heng, and Vassilis J. Tsotras. Maintaining connectivity in dynamic multimodal network models. In *ICDE*, 2008.

[2] Caliper Corporation. *TransCAD Transportation GIS Software Ref. Manual*. Caliper Corporation, 1996.

[3] E. Hoel, W.L. Heng, and D. Honeycutt. High performance multimodal networks. In *SSTD*, 2005.

[4] E. Hoel, S. Menon, and S. Morehouse. Building a robust relational implementation of topology. In *SSTD*, 2003.

[5] S. Morehouse. Arc/info: a geo-relational model for spatial information. *Proc. of AUTOCARTO 8. ASPRS*, 1985.

[6] F. Southworth and B. Peterson. Intermodal and international freight network modeling. *Geographic Information Systems in Transportation Research*, 8:147–166, 2000.

[7] S. Winter. Modeling costs of turns in route planning. *Geoinformatica*, 6(4):345–361, 2002.