INTEGRATION OF MULTIPLE GEO-SPATIAL DATASETS

Eliyahu Safra
Yerach Doytsher
Mapping and Geo-Information Engineering
Technion – Israel Institute of Technology
Technion city, Haifa, 32000, Israel
safra@technion.ac.il
doytsher@technion.ac.il

ABSTRACT

When multiple geo-spatial datasets are integrated, the process should be done in several stages. In each stage two datasets are fused to one. Since different sequences may lead to different results, it is important to select a sequence that would lead to the required results. Two approaches, namely the sequential and hierarchical, for selecting the sequence are presented, and their results are compared. In the sequential approach, initially two datasets are integrated, then, in each stage the result of the previous stage is integrated with a fresh dataset. In the hierarchical approach, in each stage, every pair of datasets is integrated to one, until there is only one dataset. The results of extensive experimentations are presented. The tests show that for large overlap the second approach have better results in a cost of lower versatility.

INTRODUCTION

Spatial Data integration is one of the main challenges in front of the GIS community. Nowadays when lots of datasets may be used through the WEB, many queries that in the past required data collection can be solved using existing data. However, to solve queries that demand more than one dataset, integration algorithms are needed. So, we are looking for efficient methods that can suit for small and medium datasets.

Integration is more than just overlaying the datasets one on top of the other; integration is more like creating one comprehensive dataset using several partial datasets. When two or more datasets are integrated all the objects represent some entity (or for short corresponding objects) should be identified and fused to a single object. By doing so, each real-world entity is represented in the result dataset by one object. So the result dataset may be seen as a regular dataset. The focus of this work is in finding the sets of corresponding objects.

Integration Methodology

When several vector datasets are collected independently, we expect them to be different in both the schema and the data instances. These differences may occur even if the datasets subjects are the same. Typically, the integrator deals with these differences in the first stages of the integration process. First, an IS (Integrated Schema) is built using the two schemas, then the data instances are matched in order to find different data instances representing the same real-world entity. Finally, conflicts that arise during the process are dealt with. When the process is done a new dataset is created containing all the information in the input datasets.

Next, we will briefly describe each of the three steps:

1. **Building the IS –** The process of creating the result set schema consists of two parts:
   a. Finding correlations between attributes in the schema (inter-schema correspondence) ((Gal, 2003), (Madhavan, 2002), (Park, 2001), (Spaccapietra, 1994))
   b. Building the result-dataset schema. (Spaccapietra, 1992) Finding correspondences between schema elements is not enough. In order to build the IS we also needed to define integration rules and explicit instructions on how to deal with conflicts.
2. **Matching the data instances –** Finding correspondence between data-instances (i.e. objects describing the same real-world entity). Matching algorithms for non-spatial datasets using the well-known TF/IDF appear in (Gravano, 2001), (Gravano, 2003) and an approach for matching spatial objects appears in ((Lemarie,

TF/IDF – Term Frequency and Inverse Document Frequency

1996), (Sester, 1998)), however, these articles describe a general approach and not a detailed method. Matching spatial objects is also part of map conflation. However, in map conflation the focus is on geometrical fitting, thus only few objects are matched in a one to one manner.

3. **Removing conflicts –** There are several aspects of possible conflicts which appear. First, according to the assumption that each object should appear once in the result. If an object appears more than once, or does not appear at all, then it is a conflict. Another conflict may be when two matched objects have different values in the same attribute. When the conflict is in the location attribute, there are many algorithms for solving the problem (e.g. (Devogele, 2002), (Laurini, 1998), (Mikhail, 1976)).When the conflict is in alphanumeric attributes (e.g. phone number of a hotel) the problem is different since alphanumeric values are typically discrete and not continuous like location attributes, so the value is set according to the more trustable source (Papakonstantinou, 1996).

In the database discipline, most of the literature about integration deals with the integration Meta level, i.e. the system architecture and the schema matching, rather than the matching of data instances. In the geo-information discipline, the focus is on geometry alignment. Thus, the matching is between geometric features (e.g. lines) rather than between objects. This work focuses on ways to find corresponding objects, i.e. matching of data instances from several different geo-spatial sources.

Parts of this work were already published in (Safra, 2006); this paper includes additional parts as to the background and implementation aspects of the developed methods.

DEFINITIONS

Geographic Information System is a repository for geographic objects, or just objects for short. Each object contains information about a real world entity; a real world entity is described uniquely in the system by one object. In the system the objects are organized in datasets, where each dataset contains objects about a certain subject.

The input for fusion process is several datasets from different sources. The datasets should overlap on both the subject and the described area. In a preprocessing filtering stage, objects, that dose not belong to the intersection between the datasets, are eliminated. It is still assumed however, that the overlap (i.e. the number of world entities that appear in both sets) between the datasets, after the filtering stage, is not complete, i.e. there are world entities which appear only in one dataset.

When geographic datasets are integrated, the main task is to identify when two or more objects, from the different sources, represent the same entity and fuse those objects to a single object. Since each entity is represented by at most one object in a dataset, a fusion set may contain at most one object from each dataset. Thus, the matching between objects from two datasets is 1:1

Correct Fusion Set

A fusion algorithm operates over several input datasets and generates (non empty) fusion sets. Each fusion set is a collection of objects that are believed to represent the same world entity. A fusion set can contain at most one object from each dataset. A fusion set is complete if it contains all the objects that represent a given entity (but it may contain other objects as well). A fusion set is sound if all the objects in it represent the same entity (there may be, however, other objects which represent the same entity as well). A fusion set is correct if it is both complete and sound, i.e. it contains all the objects which represent an entity and nothing else.

When the number of input datasets is large, counting only correct sets do not distinguish between two sets, where one consists mostly of objects that represent the same entity while in the other set, each object represent a different entity. Intuitively, we would like to consider the first set as having a higher level of correctness. So we measure correct pairs in sets rather than correct sets (the complete definition of this measure can be found in (Beeri et al., 2005)).

Measuring the Quality of the Results

We measure the quality of the result in terms of recall and precision as in IR (Information Retrieval). Recall is the percentage of the correct pairs that were found from the total number of correct pairs in the perfect result. Precision is the percentage of the correct pairs in the result.

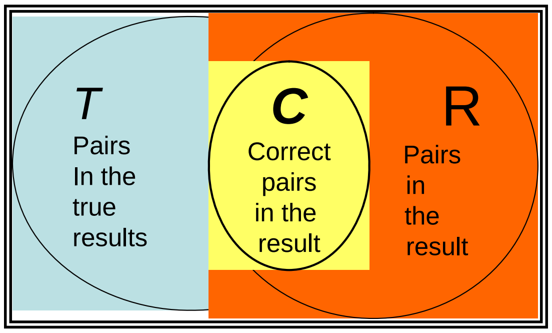

Figure 1. Type of pairs in the results.

$$Recall = \frac{\#Correct\_pairs\_in\_the\_results}{\#Correct\_pairs\_in\_the\_perfect\_answer} = \frac{C}{T}$$

$$Precision = \frac{\#Correct\_pairs\_in\_the\_results}{\#Pairs\_in\_the\_result} = \frac{C}{R}$$

Confidence

The confidence value is a number between zero and one attached to each fusion set. The confidence value is not only a measure for the certainty of a single fusion set; it can also be used as a filter for the result set. In many integration instances, the importance of the correctness of the objects in the result set (i.e. precision) is more important than its completeness (i.e. recall). The reason is that when two non-corresponding objects are fused the result may be confusing. Returning only objects with high confidence reduces the number of incorrect fusion sets, and increasing the purity of the result set.

METHODS

In previous work [Beeri et al., 2004; Beeri et al., 2005] we deal with integration of two or three sources. This work deals with integrating a large number of datasets. When the number of datasets is large, the integration should be done in several stages. Two approaches may be used in such process, namely, the *sequential* and the *hierarchical* approaches. In the *sequential* approach, first, two datasets are integrated, than, in each stage the result of the previous stage is integrated with a fresh dataset. In the *hierarchical* approach, in each stage every pair of datasets is integrated, until there is only one dataset (see Figure 2).

Note, that unlike the *holistic* approach [Beeri et al., 2005] that integrate all the input datasets in one step, both the *hierarchical* and *sequential* approaches separate the process to several stages. The reason is that integrating large number of dataset in one step is not practicable in means of computation time.

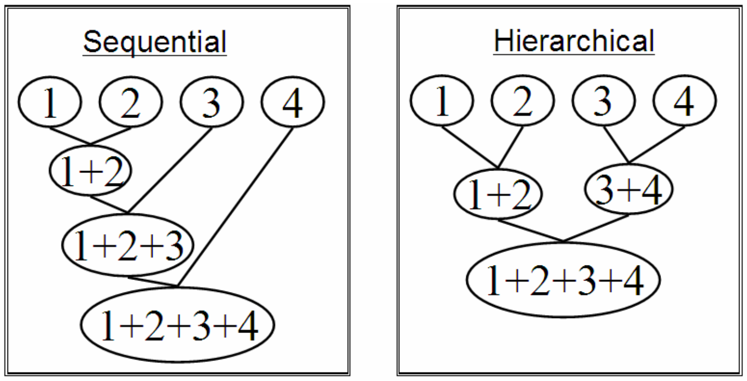

Figure 2. *Sequential* versus *hierarchical.*

Comparing Between the Approaches

The difference between the *sequential* and *hierarchical* approaches is not only in the results, but also in the process itself. This paragraph summarizes the main differences between the approaches.

Space. The *sequential* approach uses less space, since in each stage only one intermediate dataset should be saved, while in the *hierarchical* approach several intermediate datasets should be stored.

Time. *T*he number of integration operations is identical in both approaches (i.e. D-1, where D is the number of input datasets). However, the *hierarchical* approach can be implemented using p*arallel computing*. When the process can be divided between several processors the *hierarchical* procedure can be done in $\log_2(D)$ steps, while the *sequential* procedure necessarily takes (*D-1*) steps.

Versatility. In the *sequential* approach there is only one intermediate dataset in each stage, which contains all the information from the datasets that were integrated so far. When the process is stopped from some reason, this intermediate dataset may be used as an incomplete result set. In the *hierarchical* approach, we have several intermediate datasets during the integration process, the only time we have one dataset is when the process is complete.

Integration Methods

Each approach was tested with three different methods. The *Nearest-Neighbor* method, which is commonly used in commercial applications (Minami, 2000), and two other methods that we have developed, namely the *Mutually-Nearest* method, and the *Normalized-Weights* method (Beeri et al., 2004). Next we will shortly describe each of the three methods:

1. The ***Nearest-Neighbor*** join is a tool to integrate different sources that is used in commercial applications (e.g. the GIS application of ESRI). The operation (denoted as NN) takes two datasets *A* and *B*, and to each objects of *A* matches the closest object in *B*.

2. The ***Mutually-Nearest*** operation matches two objects *iff* each one of the objects is the closest to the other. Other objects, that do not have a mutual nearest neighbor object generate singleton fusion sets i.e. fusion sets with only one object.

3. In the ***Normalized-Weights*** method, the likelihood of each potential fusion set is estimated. This method creates a table of size $m+1 \times n+1$, where *m* is the size of the first dataset *(A)*, and *n* is the size of the second dataset *(B)*. Each cell*(i,j)* in the table, where $1 \le i \le m$, $1 \le j \le n$,contains the likelihood of the fusion set $\{a_i, b_j \mid a \in A, b \in B\}$; in the last column and row (i.e. $i = m+1$ or $j = n+1$) the values are the likelihoods of objects from *A* or *B* to be in singleton fusion set. Then, fusion set $\{a_i, b_j\}$ is created *iff* the value in the cell*(i,j)* exceeds a predefined threshold. Fusion set $\{a_i\}$ or $\{b_j\}$ is created *iff* the value in the cell $(i, n+1)$ or $(m+1, j)$ exceed the same predefined threshold. The value in the cell*(i,j)* is also the

iff – If and only if

confidence value of the fusion set $\{a_i, b_j\}$, and the same $(i, n+1)$ and $(m+1, j)$ for $\{a_i\}$ and $\{b_j\}$. The method consists of three main stages. First, a probability function is used to assigns values for the matching of each possible set, the probability that a_i would be matched with b_j, is inversely proportional to the distance between them. Then, a normalization process is applied to capture mutual influences between matches. Finally, a threshold is used to capture the mutual influences between matches. A full description of the algorithms appear in (Beeri, 2004) and in (Beeri, 2005).

TESTS AND THEIR RESULTS

We tested the two approaches extensively with synthetic datasets (tests with real-world data would be included in future work). The datasets were generated using environment that was implemented to imitate the real world. The datasets store location for each object, and the matching is done using only the location attribute. The reason we used points and not lines or polygons is that point is the simplest way to store location. Therefore replacing lines by points (e.g. by taking their junctions), or converting polygons to points (e.g. by taking their centroids) is simple, while transforming points to lines or polygons is much harder.

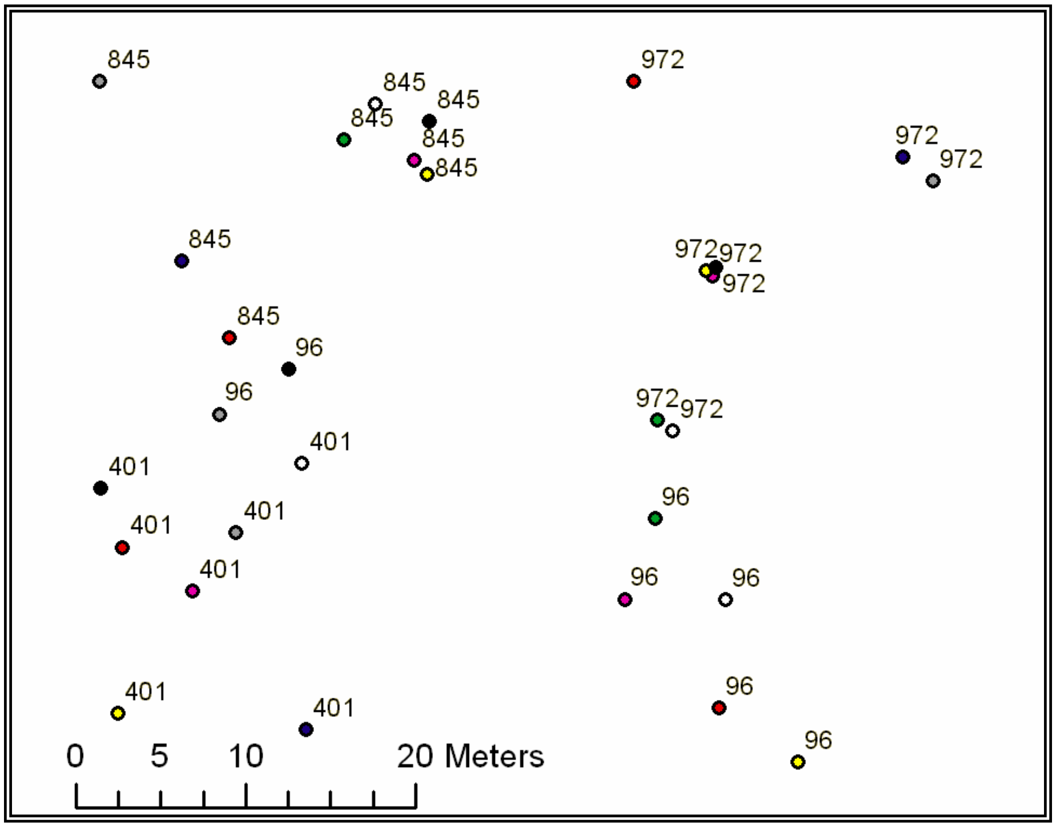

Figure 3. A partial visual view of the created datasets, each dataset with 1000 objects.

A main component of this environment is that each object in the generated datasets has an ID attribute to point to the entity it represents; therefore we can check the correctness of each fusion set, and measure the recall and precision.

The Generation Process

The generation process consists of two main stages. The first stage is to generate the "world", i.e. the entities. The parameters needed for this stage are coordinates of a square area, number of entities in the specified area and the minimal distance between any two entities.

The entity creation is iterative. In each one of the iterations, the generator chooses a random pair of numbers within the square area, for the coordinates of the new entity, and checks whether there is already a generated entity too close to it; if there is not, it generates a new entity. The entities generation process ends when the specified number of entities is reached.

The second stage is to create the datasets with the objects. The parameters needed to this stage are the number of objects in each dataset and the Error-interval i.e. the maximal distance between an object and the entity it represents.

The object creation for a dataset is also iterative; the generator randomly selects an entity *e*. The generator then checks if *e* is already represented in the dataset; if it is not, the generator creates an object *o* with location in a circle that is centered in the location of *e* and which has a radius that is equal to the error-interval parameter. The location of *o* is calculated using two random numbers. One is the angle $a\ (0 \le a \le 360)$ in a uniform distribution, and one is the distance $d\ (0 \le d \le Error - interval)$ in a Gaussian distribution. The location of *o* is in distance *d* from *e*, and in angle *a* from *e* with respect to the north.

The objects generation process ends when the number of objects is reached. Note that the overlap between the sources is not set directly, but is influenced by the world and sources size.

Test Parameters

In the tests shown in this paper, the dataset generator randomly placed 1000 entities in area of 1350*1350m, and then 5 different groups of 8 datasets have been created. The sizes of the datasets in each group are the same (the sizes are 200, 400, 600, 800 and 1000 objects). For each dataset, the represented entities were selected randomly. The location of an object relatively to the entity it represents was randomly computed using the rules that were described earlier. Note, that since the number of entities is fixed, the overlaps between the datasets are determined by setting the sizes of the datasets.

Figure 3 is a visual view of one of these tests. In this test there were 1000 objects in each dataset (i.e. complete overlap). Objects from each dataset are depicted with different color. The number attached to each object is the ID of the entity it represents.

Test Results

The tests results are shown in Figure 4. The results of the three methods are calculated for each size of datasets. For each method its bar shows the harmonic mean of recall and precision.

Our tests show that for small overlap both approaches have similar results, for medium overlap there is no better approach which consistently give better results, whereas for the case of large overlap the results of the hierarchical approach are slightly better in all the methods. All the results, including datasets with different sizes and errors, would be published in the future.

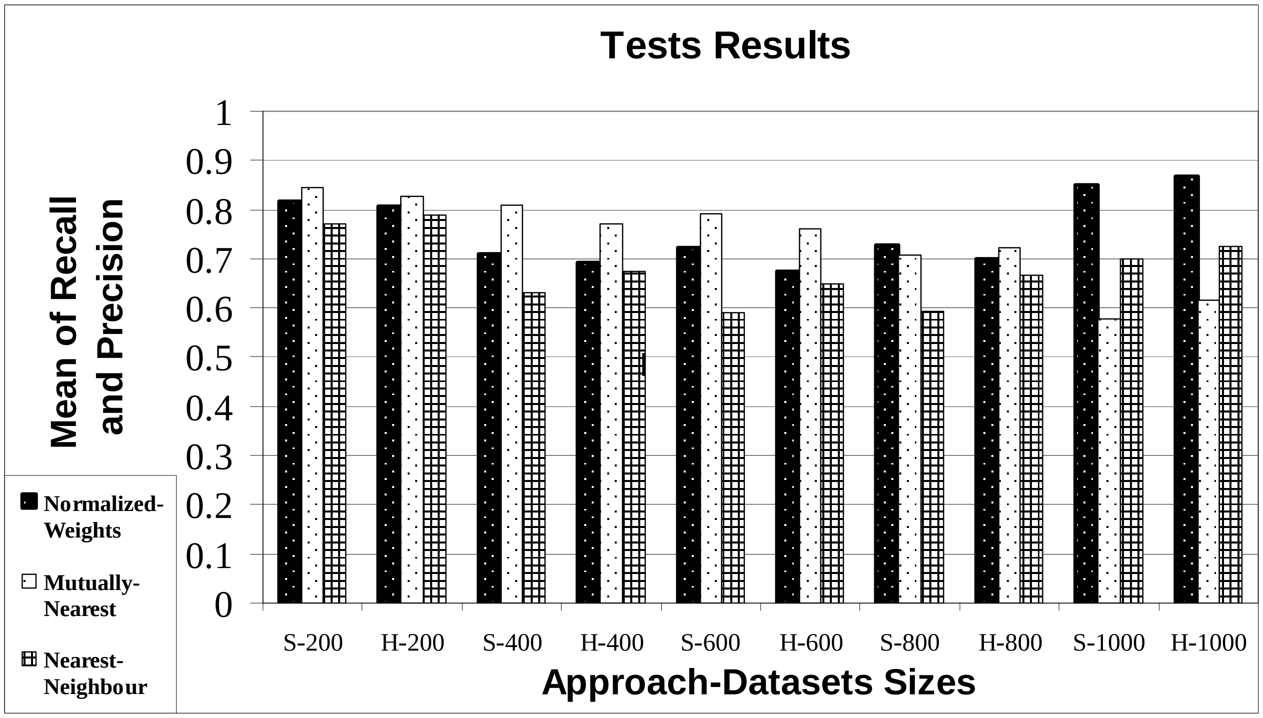

Figure 4. Harmonic mean of recall and precision (S-Sequential, H-Hierarchical).

CONCLUSIONS AND FUTURE WORK

In this work, we have investigated location-based join of multiple geo-spatial datasets. Two join approaches, namely, the *sequential* and the *hierarchical* are presented and compared. The novelty of our work is in investigating the advantages and drawbacks of each approach under different situations such as sizes of dataset and different overlaps between the datasets.

The tests show that in high overlap the hierarchical approach should be used, where in small and medium overlaps the approach should be selected according to other needs such as whether the process might be stopped in the middle or whether parallel computing is accessible.

In this work in each test, the features of the dataset are identical. Future work will deal with cases of non identical accuracies and non identical datasets sizes.

REFERENCES

Beeri, C., Kanza, Y., Safra, E., and Sagiv, Y. (2004) Object fusion in geographic information systems. In *Proc. of the 13th International Conference on Very Large Data Bases*, Toronto (Ontario, Canada).

Beeri, C., Doytsher, Y., Kanza, Y., Safra, E., and Sagiv, Y. (2005) Finding corresponding objects when integrating several geo-spatial datasets. In *Proceedings of the 13th ACM International Workshop on Geographic Information Systems, ACM-GIS*, Bremen (Germany).

Devogele, T. (2002). A New Merging Process for Data Integration Based on the Discrete Frechet Distance. *IAPRS*, vol. 34.

Gal, A., A. Trombetta, A. Anaby-Tavor, and D. Montesi. (2003). A Model for Schema Integration in Heterogeneous Databases. *IDEAS*. 2-11.

Gravano, L., P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. (2001).

Approximate String Joins in a Database (Almost) for Free. *Proceedings of the 27th International Conference on Very Large Data Bases*.

Gravano, L., P. G. Ipeirotis, N. Koudas, and D. Srivastava. (2003). Text joins in an RDBMS for web data integration. In *Proceedings of the 12th international conference on World Wide Web*.

Laurini, R. (1998). Spatial Multi-Database Topological Continuity and Indexing: A Step Towards Seamless GIS Data Interoperability. *International Journal of Geographical Information Science* 12 (4): 373-402.

Lemarie, C., and L. Raynal. (1996). Geographic data matching: first investigations for a generic tool. *In Proceedings of GIS/LIS*. 405-420.

Mikhail, E. M. (1976). *Observations and Least Squares*. University Press of America.

Madhavan, J., P. A. Bernstein, P. Domingos, and A. Y. Halevy. (2002). Representing and Reasoning about Mappings between Domain Models. *AAAI/IAAI*. 80-86.

Minami, M. (2000) Using ArcMap. *Environmental Systems Research Institute, Inc.*

Sester, M., K. H. Anders, and V. Walter. (1998). Linking Objects of Different Spatial Data Sets by Integration and Aggregation. *GeoInformatica* 2 (4): 335-358.

Papakonstantinou, Y., S. Abiteboul, and H. Garcia-Molina. (1996). Object Fusion in Mediator Systems. *Proceedings of the 22nd International Conference on Very Large Databases*.

Park, J. (2001). Schema Integration Methodology and Toolkit for Heterogeneous and Distributed Geographic Databases. *Journal of the Korea Industrial Information Systems Society*, pp. 51-64.

Safra, E., and Y. Doytsher. (2006). Integrating a Sequence of Geo-Spatial Datasets. *GISRUK 2006.*

Spaccapietra, S., and C. Parent. (1994). View Integration: A Step Forward in Solving Structural Conflicts. *IEEE Trans. Knowl. Data Eng.* 6 (2): 258-274.

Spaccapietra, S., C. Parent, and Y. Dupont. (1992). Model independent assertions for integration of heterogeneous schemas. *The VLDB Journal* 1 (1): 81-126.