Object Fusion in Geographic Information Systems

Catriel Beeri Yaron Kanza Eliyahu Safra Yehoshua Sagiv

The Selim and Rachel Benin School of Engineering and Computer Science
The Hebrew University
Jerusalem, 91904
{beeri, yarok, safra, sagiv}@cs.huji.ac.il

Abstract

Given two geographic databases, a fusion algorithm should produce all pairs of corresponding objects (i.e., objects that represent the same real-world entity). Four fusion algorithms, which only use locations of objects, are described and their performance is measured in terms of recall and precision. These algorithms are designed to work even when locations are imprecise and each database represents only some of the real-world entities. Results of extensive experimentation are presented and discussed. The tests show that the performance depends on the density of the data sources and the degree of overlap among them. All four algorithms are much better than the current state of the art (i.e., the one-sided nearest-neighbor join). One of these four algorithms is best in all cases, at a cost of a small increase in the running time compared to the other algorithms.

1 Introduction

When integrating data from two heterogeneous sources, one is faced with the task of fusing distinct objects that represent the same real-world entity. This is known as the *object-fusion* problem. Most of the research on this problem has considered either structured (i.e., relational) or semistructured (notably XML) data (e.g., [1, 10]). In both cases, objects have identifiers (e.g., keys). Object fusion is easier when global identifiers are used; that is, when objects representing the same entity are guaranteed to have the same identifier. When integrating data from heterogeneous sources, however, the object-fusion problem is much harder, due to the lack of global identifiers.

When integrating geographic databases, spatial (and non-spatial) properties should be used in lieu of global identifiers. Since location is the only property that is always available for spatial objects, we investigate location-based fusion for the case of two geographic databases. We assume that each database has at most one object per real-world entity and locations are given as points. Thus, the fusion is *one-to-one.*

Location-based fusion may seem to be an easy task, since locations could be construed as global identifiers. This is not so, however, for several reasons. First, measurements introduce errors, and the errors in different databases are independent of each other. Second, each organization has its own approach and requirements, and hence uses different measurement techniques and may record spatial properties of entities using a different scale or a different structure. For example, one organization might represent buildings as points, while another could use polygonal shapes for the same purpose. While an estimated point location can be derived from a polygonal shape, it may not agree with a point-based location in another database. A third reason could be displacements that are caused by cartographic generalizations.

The motivation for this work was hands-on experience on integrating data sources about hotels in Tel-Aviv. Two of the sources were organizations that obtained their data by different and independent means. Moreover, one organization represented buildings as points while the other—as polygons. It turned out that, in obvious cases, corresponding objects were close to each other, but did not always have identical locations. When locations were not sufficiently close, hotel names could be used to determine some of the corresponding pairs. However, there were also pairs with similar, yet not identical names; and so, some cases remained unresolved.

The current state of the art is the *one-sided nearest-neighbor join* [9] that fuses an object from one dataset with its closest neighbor in the other dataset. The ra-

Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

tionale is that even in the presence of measurement errors, different objects that represent the same entity should have close locations. However, in the one-sided nearest-neighbor join, every object from one of the two datasets is matched with some object from the other dataset, even if each dataset has objects that should not be matched with any object from the other dataset. Thus, the performance is poor when the overlap between the two dataset is small.

Another source of difficulty is due to the fact that in the presence of errors, in a dense dataset, the right match is not always the nearest neighbor. Consequently, one also has to consider objects that are further away. Moreover, there are mutual influences: If an object a is matched with an object b, then both cease to be candidates for possible matches with other objects, thereby increasing the likelihood of other matches between the other objects.

In this paper, we consider the problem of finding one-to-one correspondences among objects that have point locations and belong to two geographic databases. We present several location-based algorithms, with an increasing level of sophistication, for finding corresponding objects that should be fused. We also present the results of extensive tests that illustrate the weaknesses and strengths of these algorithms, under varying assumptions about the error bounds, the density of each spatial database and the degree of overlap between these databases.

The main contribution of our work is in showing that point locations can be effectively used to find corresponding objects. Since locations are always available for spatial objects and since a point is the simplest form of representing a location, additional information (e.g., names or polygonal locations) can only enhance the strength of our location-based algorithms.

The outline of this paper is as follows. In Section 2, we introduce the framework and formally define the problem. Section 3 describes how to measure the quality of the result of a fusion algorithm. In that section, we also discuss the factors that may influence the performance of such an algorithm. Section 4 describes the fusion algorithms that we propose. The tests and their results are discussed in Section 5. In Section 6, we consider further improvements. Section 7 describes how to choose an optimal threshold value. Related work is described in Section 8, and we conclude in Section 9.

2 Object Fusion

A *geographic database* stores *spatial objects*, or *objects* for short. Each object represents a single real-world *geographic entity*. We view a geographic databases as a *dataset* of objects, with at most one object for each real-world entity. An object has associated spatial and non-spatial attributes. Spatial attributes describe the location, height, shape and topology of an entity. Examples of non-spatial attributes are name, address, number of rooms in a hotel, etc.

We assume that locations of objects are recorded as points. This is the simplest form of representing locations. More complex forms of recording locations (e.g, polygons) can be approximated by points (e.g., by computing the center of mass). The *distance* between two objects is the Euclidean distance between their point locations.

When two geographic databases are integrated, the main task is to identify pairs of objects, one from each dataset, that represent the same entity; the objects in each pair should be fused into a single object. In general, a *fusion algorithm* may process more than two datasets and it generates *fusion sets* with at most one object from each dataset. A fusion set is *sound* if all its objects represent the same entity (but it does not necessarily contain all the objects that represent that entity). A fusion set is *complete* if it contains all the objects that represent some entity (but it may also contain other objects). A fusion set is *correct* if it is both sound and complete.

In this paper, we consider the case of two datasets and investigate the problem of finding the correct fusion sets, under the following assumptions. First, in each dataset, distinct objects represent distinct real-world entities. This is a realistic assumption, since a database represents a real-world entity as a single object. Second, only locations of objects are used to find the fusion sets. This is a practical assumption, since spatial objects always have information about their locations. As explained earlier, location-based fusion is not easy, since locations are not accurate.

We denote the two datasets as $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$. Two objects $a \in A$ and $b \in B$ are *corresponding objects* if they represent the same entity. A fusion set that is generated from A and B is either a *singleton* (i.e., contains a single object) or has two objects, one from each dataset. A fusion set $\{a, b\}$ is correct if a and b are corresponding objects. A singleton fusion set $\{a\}$ is correct if a does not have a corresponding object in the other dataset.

In the absence of any global key, it is not always possible to find all the correct fusions sets. We will present novel algorithms that only use locations and yet are able to find the correct fusion sets with a high degree of success. We will discuss the factors that effect the performance of these algorithms and show results of testing them.

3 Quality of Results

3.1 Measuring the Quality

Similarly to information retrieval, we measure the quality of a fusion algorithm in terms of *recall* and *precision*. Recall is the percentage of correct fusion sets that actually appear in the result (e.g., 91% of all the correct fusion sets appear in the result). Precision is the percentage of correct fusion sets out of all the fusion sets in the result (e.g., 80% of the sets in the result are correct).

Formally, let the result of a fusion algorithm have s^r fusion sets and let s^r_c sets out those be correct. Let e denote the total number of real-world entities that are represented in at least one of the two datasets. Then the precision is s^r_c/s^r and the recall is s^r_c/e.

Note that applying the above definitions requires full knowledge of whether two objects represent the same entity or not. Clearly, this knowledge was available for the datasets that we used in the tests.

A similar definition of recall and precision was used in [13]. However, they considered a somewhat different problem, where the correct fusion sets are always of size two and the fusion algorithm only produces fusion sets of size two. Our definition is more general in that it takes into account the possibility that a fusion algorithm produces correct as well as incorrect singleton fusion sets.

Recall and precision could also be defined in a different way, by counting object occurrences instead of fusion sets and entities. When there are only two datasets (as in our case), there is no substantial difference between the two definitions. However, when fusion sets may have more than two objects, counting object occurrences seems to be a more suitable approach. The details of the alternative definitions are beyond the scope of this paper.

3.2 Factors Affecting Recall and Precision

One factor that influences the recall and precision is the *error interval* of each dataset. The error interval is a bound on the distance between an object in the dataset and the entity it represents. The *density* of a dataset is the number of objects per unit of area. The *choice factor* is the number of objects in a circle with a radius that is equal to the error interval (note that the choice factor is the product of the density and the area of that circle). Intuitively, for a given entity, the choice factor is an estimate of the number of objects in the dataset that could possibly represent that entity. It is more difficult to achieve high recall and precision when the choice factor is large. Generally, we consider a choice factor as large if it is greater than 1; otherwise, it is small. Note that the above factors need not be uniform in the geographic area that is represented by a given dataset. In the general case, these factors may have different values for different subareas.

Suppose that the datasets A and B have m and n objects, respectively. Let c be the number of corresponding objects, i.e., the number of objects in all the correct fusion sets of size 2. Note that the number of distinct entities that are represented in the two datasets is $m + n - (c/2)$. The *overlap* between A and B is $c/(m + n)$. The overlap is a measure of the fraction of objects that have a corresponding object in the other set. One of the challenges we faced was to develop an algorithm that has high recall and precision for all degrees of overlap. Note that when the two sets represent exactly the same set of entities, then the overlap is maximal, i.e., 1. Another special case is when one dataset *covers* the second dataset, i.e., all the entities represented in the second dataset are also represented in the first dataset. We tested the performance of the algorithms on datasets with varying degrees of overlap.

4 Finding Fusion Sets

In this section, we present methods for computing the fusion sets of two datasets, $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$. The methods are based on the intuition that two corresponding objects are more likely to be close to each other than two non-corresponding objects. The recall and precision of each method depend on specific characteristics of the given datasets, e.g., the choice factors of the datasets, the degree of overlap, etc. The methods are compared in Section 5.

4.1 The One-Sided Nearest-Neighbor Join

We start by describing an existing method, the *one-sided nearest-neighbor join,* which is commonly used in commercial geographic-information systems [9]. Given an object $a \in A$, we say that an object $b \in B$ is the *nearest B-neighbor* of a if b is the closest object to a among all the objects in B. The one-sided nearest-neighbor join of a dataset B with a dataset A produces all fusion sets $\{a, b\}$, such that $a \in A$ and $b \in B$ is the nearest B-neighbor of a. Note that every $a \in A$ is in one of the fusion sets, while objects of B may appear in zero, one or more fusion sets. Thus, the one-sided nearest-neighbor join is not symmetric, i.e., the result of joining B with A is not necessarily equal to the result of joining A with B.

We modify the above definition by adding to the result the singleton set $\{b\}$ for every $b \in B$ that is not the nearest neighbor of some $a \in A$. We do that to boost up the recall of this method; otherwise, the recall could be very low.

We say that a dataset A is *covered* by a dataset B if every real-world entity that is represented in A is also represented in B. Recall that the one-sided nearest-neighbor join produces singleton fusion sets just for one of the two datasets, even if neither dataset covers the other one. Thus, the result will include wrong pairs and the precision will be low.

In conclusion, the one-sided nearest-neighbor join is likely to produce good approximations only when one dataset is covered by the other and the choice factors of the datasets are not large.

4.2 New Methods

In the following subsections, we present three novel methods for computing fusion sets. Each method computes a *confidence* value for every fusion set. The confidence value indicates the likelihood that the fusion set is correct.

The final result in each method is produced by choosing the fusion sets that have a confidence value that is above a given *threshold value*. The threshold τ $(0 \leq \tau \leq 1)$ is chosen by the user. Typically, increasing the threshold value will increase the precision and lower the recall, while decreasing the threshold value will increase the recall and decrease the precision. Controlling the recall and precision by means of a threshold value is especially useful when the datasets have large choice factors.

4.3 The Mutually-Nearest Method

Two objects, $a \in A$ and $b \in B$, are *mutually nearest* if a is the nearest A-neighbor of b and b is the nearest B-neighbor of a. The intuition behind the mutually-nearest method is that corresponding objects are likely to be mutually nearest. Note that some objects of A are not in any pair of mutually-nearest objects (and, similarly, for some objects of B). It happens when the nearest B-neighbor of $a \in A$ is some $b \in B$, but the nearest A-neighbor of b is different from a.

In the mutually-nearest method, a two-element fusion set is created for each pair of mutually-nearest objects. A singleton fusion set is created for each object that is not in any pair of mutually-nearest objects.

Consider an object $a \in A$ and let b be the nearest B-neighbor of a. The *second-nearest B-neighbor* of A is the object $b' \in B$, such that b' is the closest object to a among all the object in $B - \{b\}$.

We will now define the confidence values of fusion sets. First, consider a pair of mutually-nearest objects $a \in A$ and $b \in B$. Let a' be the second-nearest A-neighbor of b, and let b' be the second-nearest B-neighbor of a. The confidence of the fusion set $\{a, b\}$ is defined as follows.

$$\mathit{confidence}(\{a,b\}) = 1 - \frac{\mathit{distance}(a,b)}{\min\{\mathit{distance}(a,b'), \mathit{distance}(a',b)\}}$$

That is, the confidence is the complement to one of the ratio of the distance between the two objects to the minimum among the distances between each object and its second-nearest neighbor.

Now consider a fusion set with one element $a \in A$, i.e., a is not in any pair of of mutually-nearest objects. Let b be the nearest B-neighbor of a and let a' be the nearest A-neighbor of b. The confidence of the fusion set $\{a\}$ is defined as follows.

$$\mathit{confidence}(\{a\}) = 1 - \frac{\mathit{distance}(a',b)}{\mathit{distance}(a,b)}$$

That is, the complement to one of the ratio of the distance between a' and b to the distance between a and b. Note that the confidence cannot be negative, since it would imply that a, rather than a', is the nearest A-neighbor of b. The confidence of a fusion set with a single element from $b \in B$ is defined similarly.

The above formulas are used to compute the confidence value, except when the *distance upper bound* β rules out the possibility that a has a corresponding object in the other set. The parameter β should be equal to the sum of all possible errors in the locations of objects. Thus, if for all $b \in B$, it holds that $\mathit{distance}(a,b) > \beta$, then we define $\mathit{confidence}(\{a\}) = 1$ whereas for all $b \in B$, we define $\mathit{confidence}(\{a,b\}) = 0$.

A threshold can be used to increase the precision of the result by choosing only those fusion sets that have a confidence value above the threshold. Consequently, some objects from the given datasets may not be in the result. A less restrictive approach is to discard two-element fusion sets with a confidence value below the threshold, but to add their elements as singletons.

The main advantage of the mutually-nearest method over the traditional one-sided nearest-neighbor join is lower sensitivity to the degree of overlap between the two datasets. In particular, it may perform well even when neither datasets is covered by the other one. However, it does not perform well when the choice factors are large, since it only takes into account the nearest neighbor and the second-nearest neighbor, while ignoring other neighbors that might be almost as close. In situations characterized by large choice factors, it is likely that an object a will not be the nearest neighbor of its corresponding object b. If this is indeed the case, the pair a and b will not be in the result even if the distance between a and b is only slightly greater than the distance between b and its nearest neighbor.

4.4 The Probabilistic Method

The probabilistic method was devised to perform well when the choice factors are large. In such situations, it is not enough to consider only the nearest and second-nearest B-neighbors of an object $a \in A$, since there could be several objects in B that are close to a.

In the probabilistic method, the confidence of a fusion set $\{a, b\}$ depends on the probability that b is the object that corresponds to a, and that probability depends inversely on the distance between a and b.

Consider two datasets $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$. For each object $a \in A$, we will define the function $P_a : B \to [0,1]$ that gives the probability that a *chooses* $b \in B$. Similarly, for each $b \in B$, we will define the probability function $P_b : A \to [0,1]$.

Formally, the probability function P_{a_i} is defined, as shown below, in terms of the distance, the *distance* *decay factor* $\alpha > 0$ and the distance upper bound β.

$$P_{a_i}(b_j) = \frac{distance(a_i, b_j)^{-\alpha}}{\sum_{k=1}^{m} distance(a_i, b_k)^{-\alpha}} \tag{1}$$

The probability function P_{b_j} is defined similarly.

Although β is not shown explicitly in the above formula, it has the following effect. If $distance(a_i, b_j) > \beta$, then $distance(a_i, b_j)$ is taken to be infinity, i.e., $distance(a_i, b_j)^{-\alpha} = 0$. Recall that the distance upper bound should be equal to the sum of all possible errors in the locations of objects. Thus, if $distance(a_i, b_j) > \beta$, then a_i and b_j are not likely to be corresponding objects and setting $distance(a_i, b_j)^{-\alpha}$ to 0 is justified. Note that the denominator of Formula (1) is 0 if the distance between a_i and every object of B is greater than β. In this case, $P_{a_i}(b_j)$ is defined to be 0.

Due to the numerator in the above formula (and the fact that $\alpha > 0$), the probability that a_i chooses b_j increases when the distance between a_i and b_j decreases. Thus, a_i chooses its nearest B-neighbor with the highest probability. The parameter α determines the rate of decrease in the probability as the distance increases. We performed tests with different values for α and the best results were obtained for $\alpha = 2$. Due to the denominator in the above formula, the probability that a_i chooses one of the b_j's is 1, since $\sum_{j=1}^{m} P_{a_i}(b_j) = 1$ (unless a_i does not choose any b_j, which happens when the distance between a_i and every $b_j \in B$ is greater than β and hence $P_{a_i}(b_j) = 0$ for every $b_j \in B$).

The confidence of the fusion set $\{a_i, b_j\}$ is defined as follows.

$$confidence(\{a_i, b_j\}) = \sqrt{P_{a_i}(b_j) \cdot P_{b_j}(a_i)}$$

Note that $P_{a_i}(b_j) \cdot P_{b_j}(a_i)$ is the probability that a_i chooses b_j and b_j chooses a_i, i.e., the probability that a and b are corresponding objects. The confidence is defined as the square root of that probability in order that it will not be too small.

The confidence of a fusion set that includes a single object a_i is defined as follows.

$$confidence(\{a_i\}) = 1 - \sum_{k=1}^{m} \left(\sqrt{P_{a_i}(b_k) \cdot P_{b_k}(a_i)} \right)$$

The rationale for this formula is that the sum of confidences over all fusion sets that contain a_i should be equal to 1. Note that the singleton $\{a_i\}$ is given a confidence value of 1 if the distance between a_i and every object of B is greater than the distance upper bound β. Also note that in some rare cases the above formula may give a value that is less than 0 and, in such cases, we define $confidence(\{a_i\}) = 0$.

The result of the probabilistic method consists of all fusion sets that have confidence values above the threshold τ.

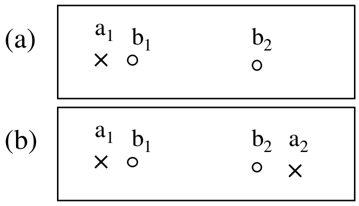


Figure 1: Adding a new object to a dataset.

When the choice factors are large, the probabilistic method performs better than either the mutually-nearest method or the one-sided nearest-neighbor join. The reason for that is that the probabilistic method assigns a confidence value to every pair of objects. Another advantage of the probabilistic methods is the ability to increase the recall by lowering the threshold. Doing so, however, may cause an object to be in more than one fusion set.

In the probabilistic method, the recall is low when the overlap between the two datasets is small. In such a situation, most of the singleton fusion sets, both the correct ones and the incorrect ones, get similar confidence values. Thus, lowering the threshold would introduce into the result many incorrect singletons, while at the same time many correct ones would still remain below the threshold (unless the threshold becomes very low). If the threshold is high, it is likely that many correct singletons would be missing from the result. In conclusion, when the overlap is small, the probabilistic method does not handle correctly objects that should be in singletons. Therefore, this method should only be used when the overlap between the datasets is large.

4.5 The Normalized-Weights Method

The normalized-weights method is a variation of the probabilistic method and it performs better when the overlap is small or medium. As in the probabilistic method, weights (i.e., confidence values) are given to each (correct or incorrect) fusion set, based on the same probability functions P_a and P_b that were defined in the previous section (see Equation 1). The initial weights are normalized by an iterative algorithm that will be described in this section. This algorithm has an effect of mutual influence between pairs of objects, as illustrated in the next example.

Example 4.1 Consider the objects a_1, b_1 and b_2 in Figure 1(a). We assume that a_1 is an object in the dataset A while b_1 and b_2 are objects in the dataset B. Let the distance between a_1 and b_1 be 1 and let the distance between a_1 and b_2 be 5. In the probabilistic method, assuming a distance decay factor of 1 (i.e., $\alpha = 1$), the probability that b_1 will choose a_1 is 1, since a_1 is the only object of A. For a_1, the probabilities to choose b_1 and b_2 are $\frac{5}{6}$ and $\frac{1}{6}$, respectively.

Thus, the confidence value that is given to the fusion set $\{a_1, b_1\}$ is $\sqrt{P_{a_1}(b_1) \cdot P_{b_1}(a_1)} = \sqrt{\frac{5}{6}}$.

In Figure 1(b), the object a_2 is added to the dataset A. The distance between a_2 and b_2 is 1 and the distance between a_2 and b_1 is 5. In the probabilistic method, after adding a_2, the confidence of the set $\{a_1, b_1\}$ is $\sqrt{P_{a_1}(b_1) \cdot P_{b_1}(a_1)} = \sqrt{\frac{5}{6} \cdot \frac{5}{6}} = \frac{5}{6} < \sqrt{\frac{5}{6}}$.

Thus, the addition of a_2 has reduced the confidence of the fusion set $\{a_1, b_1\}$. This reduction is caused by the fact that after the addition of a_2, object b_1 could potentially be the corresponding object of either a_1 or a_2, whereas before the addition of a_2, only the correspondence between b_1 and a_1 was possible. But the addition of a_2 also has an effect of increasing the likelihood of a correspondence between b_1 and a_1, because b_2 is now more likely to correspond to a_2 than to a_1 and, hence, the confidence of the fusion set $\{a_1, b_1\}$ should increase relatively to the confidence of the fusion set $\{a_1, b_2\}$. In summary, the addition of a_2 has two opposite effects: one is an increase in the confidence of the fusion set $\{a_1, b_1\}$ and the other is a decrease of that confidence. The probabilistic method is only capable of capturing the second effect, whereas the normalized-weights method captures both.

Next, we will describe the normalized-weights method. Consider two datasets $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$. The probability functions $P_{a_i}(b_j)$ and $P_{b_j}(a_i)$ are defined as in Section 4.4. The *matchings matrix* M is an $(m+1) \times (n+1)$ matrix, such that the element in row i and column j, denoted μ_{ij}, is defined as follows.

$$\mu_{ij} = \begin{cases} P_{a_i}(b_j) \cdot P_{b_j}(a_i) & : 1 \le i \le m,\ 1 \le j \le n \\ \prod_{k=1}^{n}(1 - P_{b_k}(a_i)) & : 1 \le i \le m,\ j = n+1 \\ \prod_{k=1}^{m}(1 - P_{a_k}(b_j)) & : i = m+1,\ 1 \le j \le n \\ 0 & : i = m+1,\ j = n+1 \end{cases}$$

Note that in the first case (i.e., the top line) of the above definition, μ_{ij} is assigned the probability that a_i and b_j mutually choose each other. In each row i, the element in the last column ($j = n+1$) gives the probability that a_i is not chosen by any $b \in B$. Similarly, in each column j, the element in the last row ($i = m+1$) gives the probability that b_j is not chosen by any $a \in A$.

A row (or a column) r is *normalized* if $s = 1$, where s is the sum of all the elements of r. We can always normalize r by dividing each element by s (since $s > 0$). The *normalization algorithm* is a sequence of iterations over M, such that in each iteration, the first m rows and the first n columns are normalized one by one in some order. Note that the elements of the last column are changed when the rows are normalized, but the last column itself is not normalized; similarly for the last row.

Let $M^{(0)}$ denote the matrix M, as it was defined above, and let $M^{(k)}$ denote the matrix after k iterations. It was shown by Sinkhorn [14, 15] that the normalization algorithm converges and the result does not depend upon the order of normalizing rows and columns in each iteration. We terminate the normalization algorithm when the sum of each row and each column, except for the last row and the last column, is different from 1 by no more than some very small $\epsilon > 0$ (in the tests, ϵ was equal to 0.001).

Let $M^{(t)}$ denote the matrix upon termination of the iteration algorithms. The confidence of the fusion set $\{a_i, b_j\}$ is the value in row i and column j of $M^{(t)}$. The confidence of the fusion set $\{a_i\}$ is the value in row i and column $n+1$ of $M^{(t)}$. The confidence of the fusion set $\{b_j\}$ is the value in column j and row $m+1$ of $M^{(t)}$. Given a threshold τ, the result of the normalized-weights method consists of all the fusion sets with confidence values above the threshold τ.

The normalized-weights method gives good results when the overlap between the datasets is medium or small. However, the results are not as good as those of the probabilistic method when the overlap is large, because the weights that are assigned to singletons (i.e., the weights in the last row and in the last column) are not normalized. Consequently, these weights remain rather large even when they should be almost zero (i.e., when the overlap is large). In Section 6, we discuss how to improve this method by normalizing all the rows and columns of M.

4.6 Tuning the Methods

The performance of the three new methods, which were introduced in the previous sections, can be tuned by choosing appropriate values for the threshold, the distance decay factor and the distance upper bound.

The threshold τ controls the precision and recall. Increasing the threshold increases the precision while decreasing the threshold increases the recall.

The distance decay factor α controls the effect of distant neighbors in either the probabilistic method or the normalized-weights method. In either method, the probability that an object a chooses some object from the other set is split among all the objects of the other set. This probability might be split among many objects, including far away objects that are really no more than "background noise." By choosing $\alpha > 1$, the effect of far away objects decreases exponentially. Thus, increasing α eliminates background noise. However, increasing α too much can cause these methods to act like the mutually-nearest method, i.e., pairs of objects that are not mutually nearest will be ignored.

The distance upper bound β determines which two-element fusion sets $\{a_i, b_j\}$ are a priori deemed incorrect, because the distance between a_i and b_j is too far. In practice, the value of β should be an upper-bound

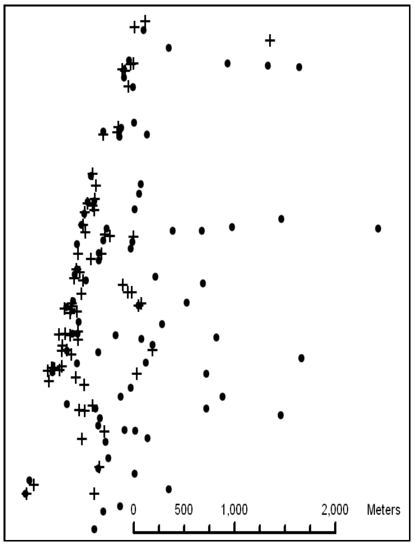


Figure 2: Hotels and tourist attractions in Tel-Aviv (the objects of MUNI are depicted by + and the objects of MAPA are depicted by •).

estimate for the sum of all possible errors in the locations of objects.

5 Testing and Comparing the Methods

We tested the different methods using real-world datasets as well as randomly-generated datasets. The two real-world datasets describe hotels and other tourist attractions in Tel-Aviv. One dataset, called MUNI, was extracted from a digital map that was made by the City of Tel-Aviv. The second dataset, called MAPA, was extracted from a digital map that was made by the Mapa Corp. The randomly-generated datasets were created using a random-dataset generator that we implemented.

5.1 Results for Real-World Datasets

In the MUNI and MAPA datasets, each object has a name, although in MUNI the names are in English and in MAPA the names are in Hebrew. The correctness of the results was checked using the specified names of the objects. However, only locations were used for computing the fusion sets.

The MAPA dataset has 86 objects and the MUNI dataset has 73 objects. A total of 137 real-world entities are represented in these datasets and 22 entities appear in both datasets. The mutually-nearest method was tested with a threshold $\tau = 0$. The probabilistic and the normalized-weights methods were tested with an optimal threshold value (see Section 7). The distance upper bound β was 150 meters and the distance decay factor α was 2. There were 40 objects in MAPA and 6 objects in MUNI without any neighbor in the other dataset at a distance of β or less.

The two datasets are depicted in Figure 2. The performance results are given in Table 1. Note that the table describes both the one-sided nearest-neighbor join of MUNI with MAPA (first column) and the one-sided nearest-neighbor join of MAPA with MUNI (second column). The one-sided nearest-neighbor join performed much worse than the other methods, while the normalized-weights method had the best performance.

5.2 Random-Dataset Generator

There are not sufficiently many real-world datasets to test our algorithms under varying degrees of density and overlap. Moreover, in real-world datasets, it is not always possible to determine accurately the correspondence between objects and real-world entities. Thus, we implemented a random-dataset generator, which is a two-step process. First, the real-world entities are randomly generated. Second, the objects in each dataset are randomly generated, independently of the objects in the other dataset.

For the first step, the user specifies the coordinates of a square area, the number of entities in that area and the minimal distance between entities. The generator randomly chooses locations for the entities, according to a uniform distribution. The generator verifies that entities are not too close to each other, i.e., the distance between entities is greater than the minimal distance specified by the user. This is done to enforce realistic constraints; for example, the distance between two buildings cannot be just a few centimeters.

For the second step, the user specifies the number of objects in the datasets and the error interval. The creation of a random object has two aspects. First, the object is randomly associated with one of the real-world entities that were created in the first step (at most one object, in each dataset, corresponds to any given entity). Second, the location of the object is randomly chosen, according to the normal distribution, in a circle with a center that is equal to the location of the corresponding entity and a radius that is equal to the error interval.

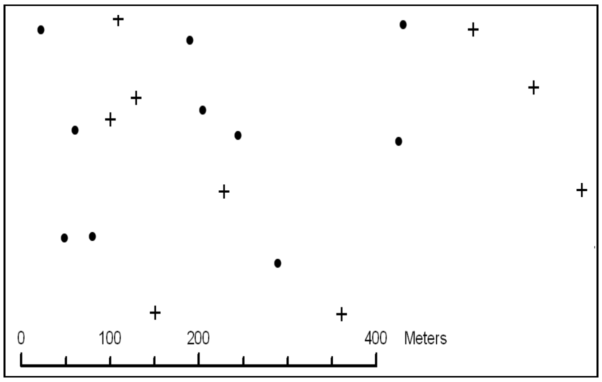
Note that the two datasets have an equal number of objects. However, in each dataset, the association of objects with entities and the locations of objects are chosen randomly and independently of the other dataset. By changing the number of objects, it is easy to control the overlap between the two randomly generated datasets. For example, if each dataset has 500 objects while the number of entities is 500, then there is a complete overlap. However, if there are 50,000 entities, then the overlap between the two datasets is very small, with a very high probability.

5.3 Results for Random Datasets

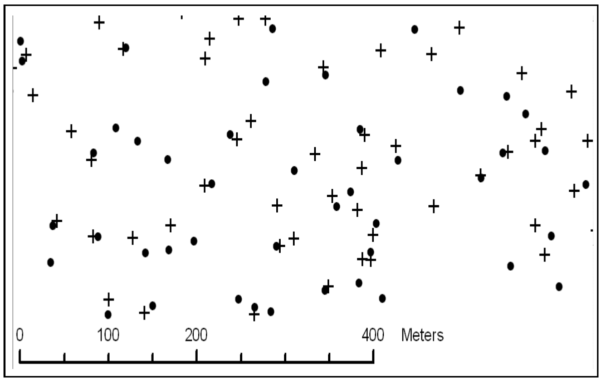
We have experimented with many randomly-generated datasets, but in this section we only describe a few tests that demonstrate the main conclusions about the

	Joining MAPA objects with nearest neighbors from MUNI	Joining MUNI objects with nearest neighbors from MAPA	Mutually Nearest	Probabilistic	Normalized Weights
Result Size	120	119	124	137	129
Recall	0.38	0.48	0.77	0.80	0.85
Precision	0.43	0.56	0.85	0.80	0.90

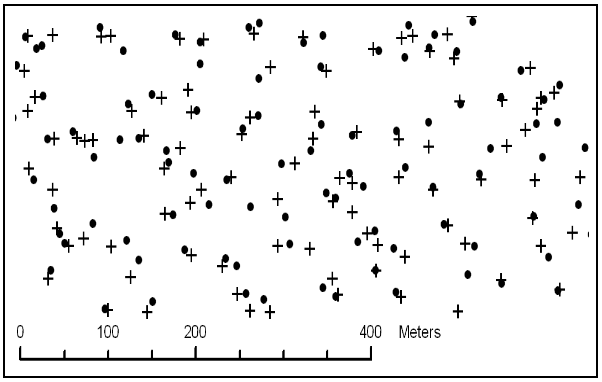
Table 1: Performance of the fusion algorithms for datasets of hotels and tourist attractions in Tel-Aviv.



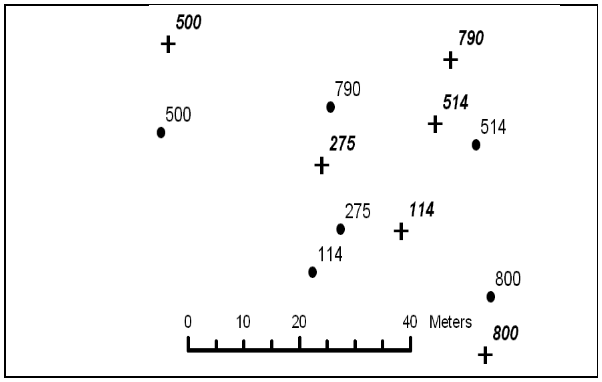
(a) 100 objects in each dataset.



(b) 500 objects in each dataset.



(c) 1000 objects in each dataset.



(d) An enlarged fragment of a pair of datasets.

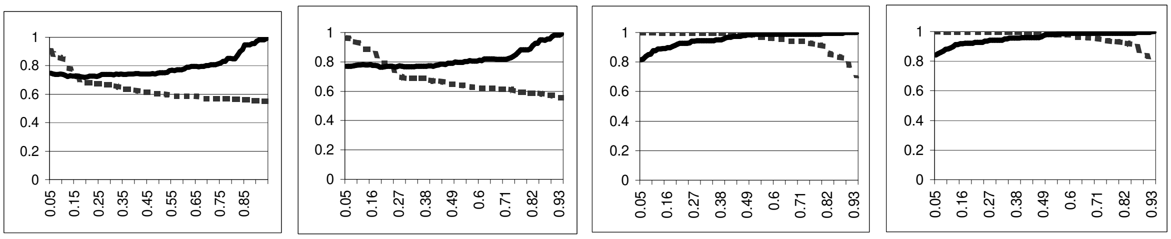
Figure 3: A visual view of the random pairs of datasets, with 100, 500 and 1000 objects.

performance of each method. The following parameters were given to the random-dataset generator. A square area of $1,350 \times 1,350$ meters, 1,000 entities, a minimal distance of 15 meters between entities, and an error interval of 30 meters for each datasets. The above parameters imply that if a dataset represents all 1,000 entities, then the choice factor is 1.55 and each entity occupies, on the average, an area of 1,822 sq m.

In the tests described below, the 1,000 real-world entities were created once and then three pairs of datasets were randomly generated (see Figure 3). The three pairs had 100, 500 and 1,000 objects in each dataset, respectively. Thus, each pairs had a different degree of overlap and density (1,000 objects in each dataset means a complete overlap). To get a sense of the difficulty in finding the correct fusion sets, an enlarged fragment of the pair of datasets from Figure 3(c) is depicted in Figure 3(d), where an entity identifier is attached to each object.

The recall and precision of each method were measured for every pair of datasets. In all the methods, the distance upper bound β was 60 meters and the error interval was 30 meters. In the mutually-nearest method, the threshold τ was 0, since it gave the highest or close to the highest recall. For the datasets with 100 objects, the recall and the precision were 0.68 and 0.81, respectively; for 500 objects, they were 0.72 and 0.77, respectively; and for 1,000 objects, they were 0.74 and 0.63, respectively.

For the probabilistic method and the normalized-

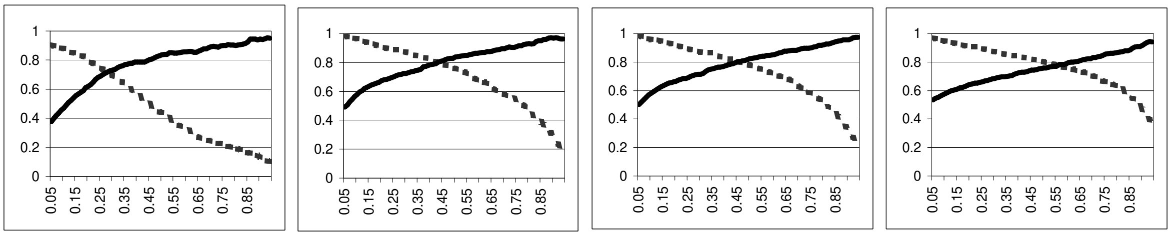


(a) Probabilistic with 100 objects.

(b) Normalized-weights with 100 objects.

(c) Improved normalized-weights with 100.

(d) Combined method with 100 objects.

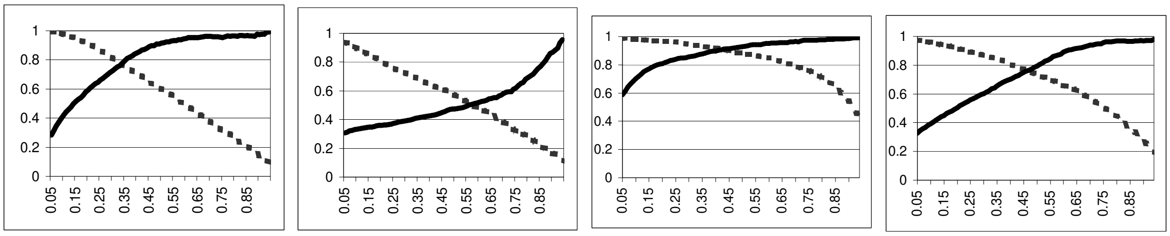


(e) Probabilistic with 500 objects.

(f) Normalized-weights with 500 objects.

(g) Improved normalized-weights with 500.

(h) Combined method with 500 objects.



(i) Probabilistic with 1,000 objects.

(j) Normalized-weights with 1,000 objects.

(k) Improved normalized-weights with 1,000.

(l) Combined method with 1,000 objects.

Figure 4: Recall (dotted line) and precision (solid line) as a function of the threshold value.

weights method (as well as for two additional methods that are explained in Section 6), Figure 4 presents the recall and the precision as a function of the threshold. Tests were made with different values for the distance decay factor (i.e., $\alpha = 0.5, 1, 2, 3$). We only show the results for $\alpha = 2$, since this choice was the best for both methods under all circumstances.

The main conclusions from the tests are as follows. When the overlap between the datasets is small, the mutually-nearest and the normalized-weights methods give good results. The probabilistic method does not perform well in this case, because it does not find the correct singletons and instead creates incorrect pairs. For a medium overlap, the best results are given by the normalized-weights method. This is demonstrated in the case of 500 objects in each dataset. When the overlap is large, the best results are given by the probabilistic method. The normalized-weights method does not perform well in this case, because singletons get weights (i.e., confidence values) that are too large.

We will now give a detailed analysis of the graphs in Figure 4. Essentially, a method performs well if it assigns low confidence values to incorrect fusion sets and high confidence values to correct ones. Most of the graphs in Figure 4, except for 4(a), 4(b) and 4(j), exhibit this phenomenon. In the low range of threshold values, a small increase of the threshold value eliminates many incorrect fusion sets and only a few correct ones. Thus, the precision rises sharply while the recall declines slowly. In the high range of threshold values, a small increase of the threshold value eliminates many correct fusion sets and only a few incorrect ones. Thus, the precision rises slowly while the recall declines sharply. Also note that the precision and the recall change more sharply as the overlap grows. A bigger overlap means a bigger density, which causes more pairs to get confidence values that are close to each other.

Figures 4(a) and 4(b) show the performance of the probabilistic and the normalized-weights methods in the case of a small overlap. In that case, many correct singletons get low confidence values while many incorrect pairs get high confidence values. Thus, when a low threshold value is increased, many correct singletons are eliminated and the recall declines sharply. The precision remains about the same, since many incorrect fusion sets are also eliminated. In the high range of threshold values, a small increase of the threshold value eliminates many incorrect pairs and the precision rises sharply while the recall remains about the same. In the medium range of threshold values, there is very little change in the recall and the precision, because the confidence values are either very low or very high, as a result of the low density (i.e., an object has only a few close neighbors in the other dataset).

Figure 4(j) shows the performance of the normalized-weights method in the case of a complete overlap (i.e., there are no correct singletons). In that method, the last row and the last column of the matching matrix are not normalized and, consequently, many incorrect singletons get high confidence values, regardless of the degree of overlap. Thus, the precision rises sharply only in the high range of threshold values.

6 Complete Normalization

In this section, we will show how knowledge about the degree of overlap between the datasets can improve the normalized-weights method.

6.1 The Normalization Value

Ideally, the normalized-weights method should give to singletons high confidence values when the overlap is large and low confidence values when the overlap is small. However, since the last row and the last column of the matching matrix are not normalized, the confidence values of singletons tend to remain high in any case. Recall that each entry in those row and column gives the probability that some object does not have a corresponding object in the other set. Thus, those row and column should not be normalized to 1.

Suppose that the datasets A and B have m and n objects, respectively, and c is the number of real-world entities that are represented in both datasets. Thus, $m-c$ objects of A and $n-c$ objects of B should be in singleton fusion sets. Ideally, an entry in either the last row or the last column should be 1 if its corresponding object is in a correct singleton, and should be 0 otherwise. Consequently, we improve the normalization algorithm by normalizing the last row to $n-c$ and the last column to $m-c$. Note that if $n-c=0$ (or $m-c=0$), then we just set all the elements in the last row (or last column) to 0 and need not normalize this row (or column) anymore.

The improved normalized-weights method always performs better than all the other methods, as shown in Figures 4(c), 4(g) and 4(k). Note that in these three tests, the correct value of c was used.

6.2 Estimating the Overlap

The improved normalized-weights method requires a good estimate for the number c of correct pairs of corresponding objects. If the association of objects with entities in each dataset is independent of the other dataset, then an approximation of c is given by

$$c \approx \frac{m}{e} \cdot \frac{n}{e} \cdot e$$

where e is the number of real-world entities and the two datasets have m and n objects, respectively. If e is not known, then a less accurate approximation of c can be obtained by replacing e with the number of pairs that are produced by the mutually-nearest method.

We conducted a series of tests with the following parameters. The number of entities ranged from 500 to 1,000 in increments of 100. The size of the datasets ranged from 50 to 500 in increments of 50. The minimal distance between entities was either 15, 25 or 35 meters. The conclusion from these tests is that c can be approximated by

$$c \approx 1.2p - \frac{e}{10}$$

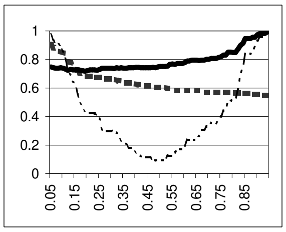
where p is the number of pairs that are produced by the mutually-nearest method and e is the number of entities. In particular, when p is approximately $\frac{e}{2}$, then c is also approximately $\frac{e}{2}$.

The above formula suggests that p is a good estimate for c. Thus, we also tested the *combined method* that applies the improved normalized-weights method with p as the estimate for c. The results for the random datasets of Section 5.3 are shown in Figures 4(d), 4(h) and 4(l). These results are almost as good as the results for the improved normalized-weights method, using the correct value of c.

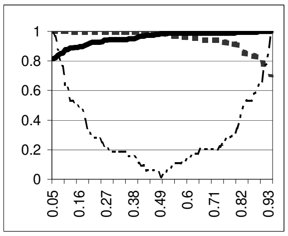
7 Choosing the Threshold Value

For the mutually-nearest neighbor, a threshold value of zero will give the highest recall with a good precision. For the other methods, choosing a good threshold value is more complicated. A threshold value of 0.5 is a reasonable choice, since it usually gives results with good recall and precision, and without duplicates (i.e., each object appears in at most one fusion set). Note that if the threshold is just above 0.5, then there cannot be duplicates. An *optimal* threshold gives the best combination of recall and precision. Formally, an optimal threshold can be defined as the one that gives the largest geometric average of the recall and the precision. In Figure 4, this is usually the threshold at the intersection of the graphs of the recall and the precision.

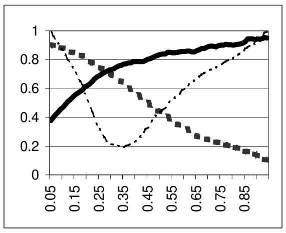
We will now describe how to estimate the optimal threshold without a priori knowledge of the correct fusion sets. The idea is to minimize two types of errors.



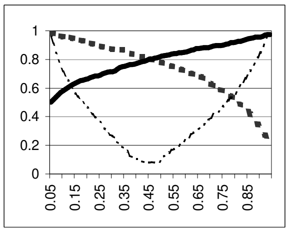
(a) Probabilistic with 100 objects.



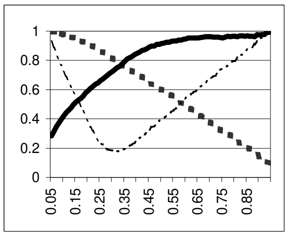
(b) Improved normalized-weights with 100.



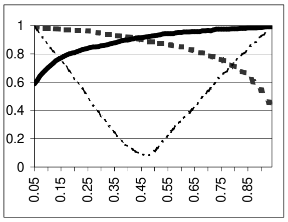
(c) Probabilistic with 500 objects.



(d) Improved normalized-weights with 500.



(e) Probabilistic with 1,000 objects.



(f) Improved normalized-weights with 1,000.

Figure 5: The error count (shown by a dashed line), recall and precision as a function of the threshold.

A *missing object* is an object that does not appear in any fusion set. A *duplicate object* is an object that appears in more than one fusion set. The *error count* is the number of missing objects plus the number of duplicate occurrences of objects (if an object appears in k fusions sets, then it is counted $k-1$ times). The threshold value should be chosen so that the error count is minimal. Figure 5 shows the error count as a function of the threshold for two methods, using the random datasets of Section 5.3 (the graphs for the other two methods are similar). It can be seen that a threshold value that minimizes the error count is close to the optimal threshold. This was also confirmed for the real-world datasets, MUNI and MAPA. For the probabilistic method, the optimal threshold was 0.31 and the threshold that minimized the error count was 0.4. For these two threshold values, there was only a small difference (0.796 vs. 0.784) between the geometric averages of the recall and the precision. For the normalized-weights method, the optimal threshold was 0.43 and the threshold that minimized the error count was 0.45, while the geometric averages were about the same (0.873 and 0.868). Thus, for a real pair of datasets, the threshold value can be chosen by trying several different values and choosing the one that minimizes the error count.

8 Related Work

Map conflation is the process of producing a new map by integrating two existing digital maps, and it has been studied extensively in the last twenty years. Map conflation starts by choosing some *anchors,* i.e., pairs of points that represent the same location. A triangular planar subdivision of the datasets with respect to the anchors (using Delaunay triangulation) is performed and a rubber-sheeting transformation is applied to each subdivision [4, 5, 6, 11, 12].

Map conflation cannot be applied without initially finding some anchors. Thus, our methods can be used as part of this process. Unlike map conflation, which is a complicated process, our algorithms are most suitable for the task of online integration of geographic databases using mediators (e.g., [2, 17]).

A feature-based approach to conflation, which is similar to object fusion, has been studied in [13]. Their approach is different from ours in that they find corresponding objects based on both topological similarity and textual similarity in non-spatial attributes. Once some corresponding objects have been found, they construct some graphs and use graph similarity to find more corresponding objects.

In [3], topological similarity is used to find corresponding objects, while in [7, 8, 16] ontologies are used for that purpose.

The problem of how to fuse objects, rather than how to find fusion sets, was studied in [10]. Thus, their work complements ours.

9 Conclusion and Future Work

The novelty of our approach is in developing efficient algorithms that find fusion sets with high recall and precision, using only locations of objects. The mutually-nearest method is an improvement of the existing one-sided nearest-neighbor join and it achieves substantially better results.

We also developed a completely new approach, based on a probabilistic model. We presented several algorithms that use this model. The improved normalized-weights method achieves the best results under all circumstances. This method combines the probabilistic model with a normalization algorithm and information about the overlap between the datasets. We showed that the mutually-nearest method provides a good estimate for the degree of overlap. We also showed how to tune the methods and how to choose an optimal threshold value.

Several interesting problems remain for future work. One problem is to deal with situations in which an object from one set could correspond to many objects in the other set. This may happen, for example, when one dataset represents shopping malls while the other dataset represents shops.

A second problem is to develop fusion algorithms for more than two datasets. This problem is similar to the previous one in that both require finding fusion sets that have several, rather than just 2 objects. In principle, this problem can be solved by a sequence of steps, such that only two datasets are involved in each step (where one of those two datasets is the result of the previous step). However, it is not clear whether a long sequence can still give good recall and precision.

A third problem is how to utilize locations that are given as polygons or lines, rather than just points.

A fourth research direction is to combine our approach with other approaches, such as the feature-based approach of [13], topological similarity (e.g., [3]) or ontologies (e.g., [7, 8, 16]).

Acknowledgments

We thank Michael Ben-Or for pointing out to us the work of Sinkhorn [14, 15]. We also thank the anonymous referees for helpful suggestions. This research was supported in part by a grant from the Israel Ministry of Science and by the Israel Science Foundation (Grant No. 96/01).

References

[1] B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-based integration of XML Web resources. In *Proceedings of the First International Semantic Web Conference*, pages 117–131, Sardinia (Italy), 2002.

[2] O. Boucelma, M. Essid, and Z. Lacroix. A WFS-based mediation system for GIS interoperability. In *Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems*, pages 23–28, 2002.

[3] T. Bruns and M. Egenhofer. Similarity of spatial scenes. In *Proceedings of the 7th International Symposium on Spatial Data Handling*, pages 31–42, Delft (Netherlands), 1996.

[4] M. A. Cobb, M. J. Chung, H. Foley, F. E. Petry, and K. B. Show. A rule-based approach for conflation of attribute vector data. *GioInformatica*, 2(1):7–33, 1998.

[5] Y. Doytsher and S. Filin. The detection of of corresponding objects in a linear-based map conflation. *Surveying and Land Information Systems*, 60(2):117–128, 2000.

[6] Y. Doytsher, S. Filin, and E. Ezra. Transformation of datasets in a linear-based map conflation framework. *Surveying and Land Information Systems*, 61(3):159–169, 2001.

[7] F. T. Fonseca and M. J. Egenhofer. Ontology-driven geographic information systems. In *Proceedings of the 7th ACM International Symposium on Advances in Geographic Information Systems*, pages 14–19, Kansas City (Missouri, US), 1999.

[8] F. T. Fonseca, M. J. Egenhofer, and P. Agouris. Using ontologies for integrated geographic information systems. *Transactions in GIS*, 6(3), 2002.

[9] M. Minami. *Using ArcMap.* Environmental Systems Research Institute, Inc., 2000.

[10] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *Proceedings of the 22nd International Conference on Very Large Databases*, pages 413–424, 1996.

[11] B. Rosen and A. Saalfeld. Match criteria for automatic alignment. In *Proceedings of 7th International Symposium on Computer-Assisted Cartography (Auto-Carto 7)*, pages 1–20, 1985.

[12] A. Saalfeld. Conflation-automated map compilation. *International Journal of Geographical Information Systems*, 2(3):217–228, 1988.

[13] A. Samal, S. Seth, and K. Cueto. A feature based approach to conflation of geospatial sources. *International Journal of Geographical Information Science*, 18(00):1–31, 2004.

[14] R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

[15] R. Sinkhorn. Diagonal equivalence to matrices with perscribed row and column sums. *The American Mathematical Monthly*, 74(4):402–405, 1967.

[16] H. Uitermark, P. V. Oosterom, N. Mars, and M. Molenaar. Ontology-based geographic data set integration. In *Proceedings of Workshop on Spatio-Temporal Database Management*, pages 60–79, Edinburgh (Scotland), 1999.

[17] G. Wiederhold. Mediation to deal with heterogeneous data sources. In *Introperating Geographic Information Systems*, pages 1–16, 1999.