DOI: 10.1111/tgis.12561

RESEARCH ARTICLE

Transactions in GIS WILEY

Optimal spatial data matching for conflation: A network flow-based approach

Ting Lei[1] | Zhen Lei[2]

[1]Department of Geography and Atmospheric Science, University of Kansas, Lawrence, Kansas

[2]Automation School, Wuhan University of Technology, Wuhan, China

Correspondence
Zhen Lei, Automation School, Wuhan University of Technology, Wuhan, 430070, China.
Email: leizhen@whut.edu.cn

Abstract

Spatial data conflation involves the matching and merging of counterpart features in multiple datasets. It has applications in practical spatial analysis in a variety of fields. Conceptually, the feature-matching problem can be viewed as an optimization problem of seeking a match plan that minimizes the total discrepancy between datasets. In this article, we propose a powerful yet efficient optimization model for feature matching based on the classic network flow problem in operations research. We begin with a review of the existing optimization-based methods and point out limitations of current models. We then demonstrate how to utilize the structure of the network-flow model to approach the feature-matching problem, as well as the important factors for designing optimization-based conflation models. The proposed model can be solved by general linear programming solvers or network flow solvers. Due to the network flow formulation we adopt, the proposed model can be solved in polynomial time. Computational experiments show that the proposed model significantly outperforms existing optimization-based conflation models. We conclude with a summary of findings and point out directions of future research.

1 | INTRODUCTION

A fundamental part of geospatial analysis is to combine and synthesize data from different sources in order to obtain the necessary information, make discoveries, and cross-validate results. Conflation is the process of combining "two digital map files to produce a third map file which is better than each of the component source maps"

(Saalfeld, 1988). The datasets involved typically share some common elements that correspond to the same entities in reality and should be unified or merged during the data fusion process. Operationally, the conflation process consists of two main steps: (a) establish a correspondence (i.e., match plan) between counterpart features (feature matching); and (b) merge the geometry and attribute data of matched features (feature merging).

Early examples of conflation research date back to the study of conflating common junctions and edges in creating coherent topological datasets such as the U.S. Census TIGER files. Other prominent examples include a systemic effort by the U.S. Census Bureau to integrate U.S. Geological Survey (USGS) data (Rosen & Saalfeld, 1985; Saalfeld, 1988); a series of conflation-related initiatives by the OGC (Open Geospatial Consortium), including testbeds 5 (Brennan, 2008), 9, 10, and 12; as well as efforts by the NGA (National Geospatial-Intelligence Agency) in developing open-source conflation platforms (Canavosio-Zuzelski, Surratt, Bower, Governski, & Sorenson, 2015).

Spatial datasets often come from different agencies and vendors, each having a different role and scope. Some datasets may be produced even at different points of time in history. A commonly studied application of conflation is that of transportation networks. This is probably due to the important function of such networks as corridors of movement and as a common reference system. A plethora of sources provides data about road networks. Public agencies such as the U.S. Census Bureau (and the USGS) maintain regularly updated databases containing a rich set of socio-economic attributes. Private companies such as TeleAtlas (now TomTom) and Navteq provide high-fidelity network datasets for navigation, planning, and other purposes (often with improved geometric accuracy and more specific attributes about the transportation infrastructure). The advancement of sensor technology and open-data projects (such as Open Street Map, OSM) have made an increasingly rich set of street network data available to the public in the form of volunteered geographic information (VGI). Transportation research often needs comprehensive information about the transportation infrastructure and population characteristics from all these data sources in order to study travel behavior, predict traffic flow, and develop future policies and plans.

Despite its potential utility, conflation has only limited application except in large organizations. Even in transportation planning, conflation has been considered a complex and time-consuming process, to the extent that some planning organizations (C. Gorugantula, pers. comm., October 2, 2015) have decided to abandon some of the datasets they have already collected. Several reasons contributed to the difficulty in practical application of conflation methods.

First, the success of existing conflation methods depends heavily on the accuracy of input map data, due to their reliance on basic GIS operations such as buffer analysis and proximity. However, spatial datasets are often "problematic" for these methods, because they have non-systematic trends of displacement in different parts of a map (Church et al., 1998). Figure 1a demonstrates such an example using an overlay of the TeleAtlas and TIGER datasets in Los Angeles, CA. While most road features in the two datasets align well with each other, there exists a significant shift for the polylines representing Palos Verdes Drive, to the extent that the two polylines representing the opposite sides of the road almost coincide with each other. Such displacement will most likely defeat existing methods based on buffer analysis (Goodchild & Hunter, 1997) or choosing the closest pairs of features (Ahmadi & Nascimento, 2016), because they will unify the opposite sides of the road as one feature. A similar situation is illustrated in Figure 1b for an overlay of TeleAtlas and OSM data in Santa Barbara, CA, in which two polyline representations of Viola Way are displaced so much that the first segment of Viola Way in the TeleAtlas data (red) almost coincides with the second segment in the OSM data (blue).

The above cases illustrate extreme displacements that would confuse many existing conflation algorithms. Apparently, this was what motivated the development of the rubber-sheeting method and its various extensions: ideally, one can identify anchor points around the areas of extreme displacement and apply a local (affine) transformation to remove the data shift therein. One can then apply one of the simpler methods (e.g., based on buffer analysis). However, this procedure introduces a chicken and egg problem, because the identification of the counterpart anchor points is itself a (lesser) conflation problem, which often requires manual labor and hinders automated conflation.

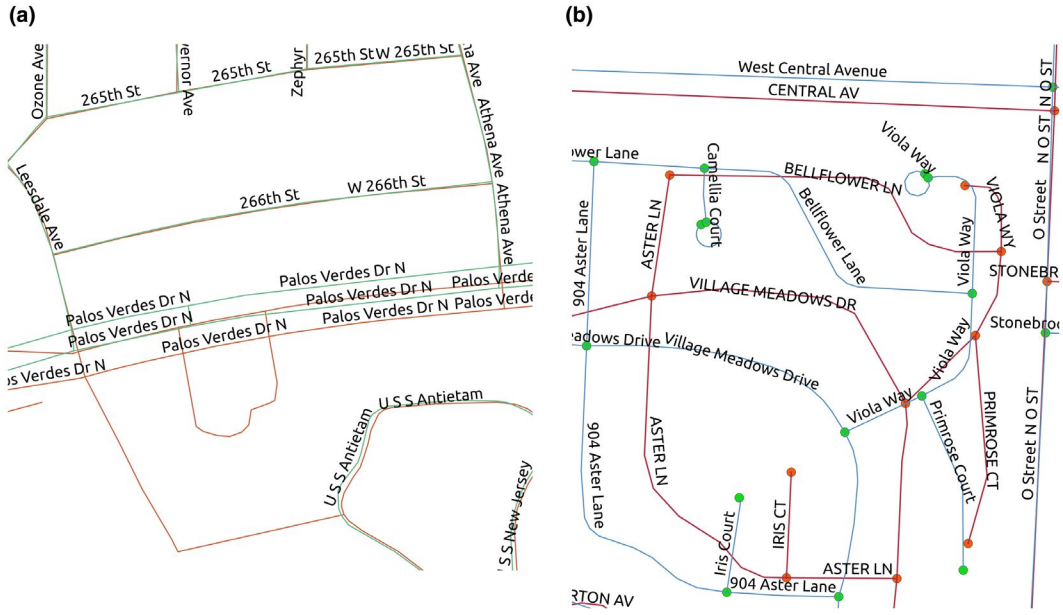

FIGURE 1 Different patterns of positional shift between two different road datasets: (a) Los Angeles road network; and (b) Santa Barbara road network

Second, existing conflation methods generally follow sequential matching of counterpart features, which may be sub-optimal (Tong, Liang, & Jin, 2014). As has been demonstrated in Figure 1a, traditional methods based on buffer analysis can become "greedy" and choose to match the wrong sides of a street merely based on proximity. Once such an erroneous match is made, there is often no mechanism to undo the false match and re-match features. Due to the complexity and limitations of existing methods, planners and analysts often have to employ a heavily manual conflation process, which is time-consuming and often prohibitively expensive.

To overcome these shortcomings, there is a need for a conflation methodology that is reliable, efficient, and can be adapted for different settings. The goal of this article is to introduce a framework for automatic, optimal data fusion that satisfies the following requirements. First, the tool should be well-defined, easy to understand, and easy to modify. In particular, the tool should have a clear measure of the conflation error and a description of the requirement of matching. Contrary to the black-box approach of traditional methods, the tool should expose the criteria of the conflation model and the structure of match relations through standard languages. Second, the tool should be automatic and efficient. The tool should require minimum or no human intervention once the modeler has identified the criteria and constraints of conflation. The tool should be robust and insensitive to systematic or local map errors.

2 | BACKGROUND AND PREVIOUS RESEARCH

Most existing matching methods in conflation are sub-optimal by construction, in the sense described in Section 1. Optimal matching methods do exist, but they are limited both in number and function. They are largely unsuccessful due to their inflexibility and high computational costs. This section will review both optimal and sub-optimal approaches. From the outset, it should be noted that conflation itself is a complex process involving multiple phases, including feature matching, attribute merging, and geometry merging. We refer the reader to the comprehensive review by Ruiz, Ariza, Urena, and Blázquez (2011) for a detailed description and classification of conflation

methods. In the discussion here, we focus on the relevant methods for the feature-matching phase as well as related theories needed for explaining the proposed model.

Existing conflation methods can be classified according to several factors. Depending on the type of geometry, different conflation methods have been developed for matching point features (e.g., gazetteers and points of interests [POIs]; McKenzie, Janowicz, & Adams, 2014), lines (e.g., transportation networks; Pendyala, 2002), and polygons (e.g., building footprints, parcels, census tracts; Masuyama, 2006), respectively. Depending on the properties used to compute the similarity between features, different conflation methods may focus on geometry, attribute (e.g., McKenzie et al., 2014), and topological properties. Similarity in geometry is the most widely used metric. Such similarity measures compare the lengths, shapes, and orientations of two features or compute certain generalized distances such as the Hausdorff distance. Attribute similarity measures compare the difference of two features using string (or numerical) attributes such as street names. Topological measures compare two features based on (topological) properties such as the number of edges that a node is connected to.

In addition, an important distinction between conflation methods is the "cardinality" of the relation between corresponding features that is assumed to exist. The concept of the cardinality (or degree) of a relation refers to the number of times entities in one dataset can be linked to entities in another. As shown in Figure 2, a one-to-one matching (Figure 2a) indicates that two features involved represent the same object in reality. This type of matching is the simplest case and can be handled successfully by most existing methods.

Beyond the one-to-one case, however, matching becomes more complicated and less well-defined. The same cardinality can mean different things to different researchers. For example, one-to-many (1:*m*) matching can mean two different things: (a) a part–whole relationship (Figure 2c), in which features in one dataset belong to a feature in the second dataset; and (b) a parallel edge case (Figure 2b), in which two parallel lines in dataset 1 correspond to the same line in dataset 2. The parallel edge case can happen, for example, in a lane-based representation of transportation networks. A conflation method may be suitable for one case and inapplicable to another. The many-to-many case is even more complex and controversial. Some many-to-many cases are two-way one-to-many

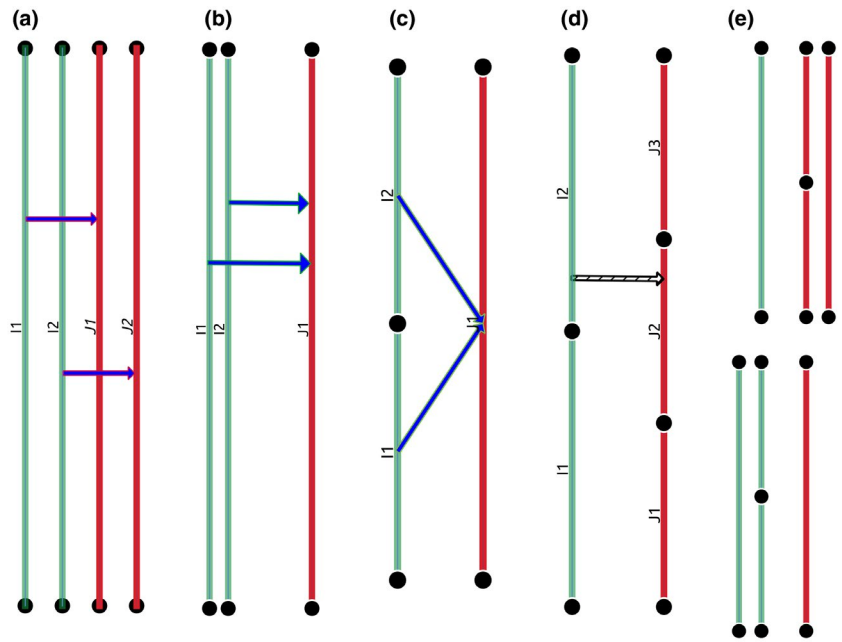

FIGURE 2 Cardinality of match between dataset 1 (green) and dataset 2 (red): (a) one-to-one; (b) one-to-many (parallel); (c) one-to-many (part–whole); (d) many-to-many; and (e) two-way one-to-many

relations (Figure 2e), while others cannot be reduced to one-to-many cases and are treated as errors by some researchers (Brown, Rao, & Baran, 1995).

One of the earliest examples of practical conflation methods was the rubber-sheeting method adopted by the U.S. Census Bureau (Rosen & Saalfeld, 1985; Saalfeld, 1988) in an effort to integrate USGS data. This method uses a selected set of counterpart points (including end nodes and interior vertices of polylines) as "anchors" to link two datasets. The algorithm then divides the study area into smaller triangular regions (using Delaunay triangulation) between the anchor points. It then applies a continuous transformation locally in each region to reduce discrepancies between objects in the region. The algorithm may iteratively choose more anchor points inside a region and divide it further to reduce errors, or terminate when the discrepancy is small enough.

Numerous subsequent algorithms (Cobb et al., 1998; Filin & Doytsher, 2000; Masuyama, 2006; Pendyala, 2002) followed this bottom-up approach of Saalfeld (1988). Pendyala (2002) developed a method for Florida DOT to conflate their Transtat basemap with FSUTMS (Florida Standard Urban Transportation Modeling Structure) networks. The method extends the algorithm of Saalfeld (1988) and allows segments of lines to be matched between mileposts. In addition, a top-down matching procedure is developed to match edges that are not captured in the bottom-up phase. Filin and Doytsher (2000) proposed a path-based conflation procedure capable of detecting one-to-many matches. Their procedure begins with the detection of counterpart anchor nodes based on proximity, the number of edges emanating from the nodes, and the orientation of the edges. The set of candidate anchor nodes were then validated, disqualified, or augmented by a so-called "round-trip walks" procedure. Masuyama (2006) developed a manual bottom-up conflation procedure for matching and comparing polygon datasets between the 1990 and 2000 Census. Similar to Pendyala (2002) and Saalfeld (1988), his procedure started by identifying anchor points and performing local affine transformations to remove systematic positional errors as much as possible. Then, the author manually converted a match relationship into one-to-one matching by merging smaller polygons into larger ones until one-to-one correspondence could be found.

Another group of widely used conflation methods is based on buffer analysis. The simple buffer method of Goodchild and Hunter (1997) measures the similarity of two features as the percentage of one feature that falls in the buffer of the other feature. Walter and Fritsch (1999) developed a more sophisticated conflation method consisting of a buffer-growing phase to group polylines into longer paths, and identify possible counterpart path pairs using their buffers. This is followed by a tree-search phase that enumerates all possible ways of taking counterpart pairs of paths to create a match plan.

The *k*-closest pairs queries (KCPQ) is aimed at finding *k* pairs of counterpart features with the smallest distances (Ahmadi & Nascimento, 2016). It is studied widely in the database literature, with a focus on computational efficiency (Corral, Manolopoulos, Theodoridis, & Vassilakopoulos, 2004). It has been applied to matching road networks and other spatial datasets. However, there is a certain greediness in the algorithm. As illustrated in the parallel edge example in Figure 1a, KCPQ can match the wrong pair of features without considering the spatial context and the potential impact on matching other features.

Many of the above conflation methods have been implemented in commercial or open-source software. Esri (2018) developed a conflation toolset that provides many traditional conflation operations based on buffer analysis and similarity of shapes. This includes local transformations and rubber-sheeting, geometric alignment and attribute merging. A more sophisticated tool called MapMerger (ESEA, 2018), developed based on Esri ArcGIS, provides a richer set of conflation operations. Their process is iterative and requires manual matches of anchor points, before alternating iterations of automatic matching and manual editing. In the open-source world, a conflation plugin (OSM, 2018) has been developed for Java OpenStreetMap Editor. The plugin is based on the Java Conflation Suite (JCS) (Blasby, Davis, Kim, & Ramsey, 2002) and is capable of finding one-to-one matches between POIs, such as addresses and buildings. More complex matching cases are not supported. The conflation methods and tools discussed so far are generally procedural and non-optimal, and follow a sequential, bottom-up approach. In general, these methods can construct good match relations but cannot guarantee that another algorithm would not find a better match.

A major departure from the traditional procedural conflation methods is the optimization-based method of Li and Goodchild (2011). Instead of constructing the match relation incrementally, they formulated the conflation problem as a global optimization problem of optimally assigning features in one dataset to counterpart features in the other dataset. To this end, they employed the well-known assignment problem, a classic optimization model in operations research. This optimization-based approach is closely related to the proposed framework and is therefore discussed in greater detail below.

Given a set I of n workers, a set J of n jobs, and a set of costs c_{ij} for assigning worker i to job j, the assignment problem seeks to find the best plan to assign each worker to a job that minimizes the total assignment costs. The assignment problem was conventionally formulated using linear programming (LP) and expressed as a set of linear equations and inequalities. The conflation problem can be posed as an assignment problem in the following sense. One can define the cost of assigning each feature i to a potential counterpart feature j to be their (generalized) distance or a similarity measure (Li & Goodchild, 2010) (Figure 2a). One then minimizes the total assignment cost. An advantage of such an approach is that the model is solvable by the well-known simplex algorithm. Li and Goodchild used the Hausdorff distance between two polylines i, j as assignment cost c_{ij}. The distance metric measures the maximum deviation of a point in one feature from the other. They formulated the conflation problem as follows:

$$\text{Minimize} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1}$$

Subject to:

$$\sum_{j=1}^{n} x_{ij} = 1, \text{for each } i \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \text{for each } j \tag{3}$$

$$x_{ij} = \begin{cases} 1, \text{if } i \text{ and } j \text{ are matched} \\ 0, \text{otherwise} \end{cases} \tag{4}$$

In this formulation, the decision variables x_{ij} in constraint (4) represent the match decision and take a value of 1 if feature i is assigned to candidate feature j, or 0 otherwise. Constraint (2) maintains that each feature i can be matched to exactly one counterpart feature (the *uniqueness* of assignments). Constraint (3) maintains that each feature j in dataset J can be the counterpart of only one feature in dataset I (the *exclusiveness* of assignments). The objective function (1) minimizes the sum of all costs c_{ij} for assignments x_{ij}. In essence, this formulation establishes the optimal *one-to-one* correspondence between datasets (Figure 2a). Li and Goodchild (2010) modified the assignment problem by changing the uniqueness constraint (2) into an inequality form (5), which still guarantees uniqueness of assignment but does not require each feature i to assign:

$$\sum_{j=1}^{n} x_{ij} \leq 1, \text{for each } i \tag{5}$$

This modification allows the assignment problem to handle one-to-one situations in which one dataset with n elements corresponds to a subset of another, larger dataset with m elements.

Li and Goodchild (2011) attempted to solve the two-way $1:m$ matching problem by creating two assignment problems using a directed Hausdorff distance measure (i.e., two sub-models to handle right-to-left and left-to-right assignments). To allow such partial matchings, they replaced the exclusiveness constraint with a capacity constraint (6) and a new constraint (7) as follows:

$$\sum_{i=1}^{m} l_i x_{ij} \leq \alpha l_j, \text{ for each } j \tag{6}$$

$$\sum_{i=1}^{m} x_{ij} + \delta_j \geq 1, \text{ for each } j \tag{7}$$

Constraint (6) requires that the total length (l_i) of the parts that a longer line j can "contain" should not exceed its own length l_j, up to a certain level α (e.g., 110%) of errors in lengths. When matching lines of equal lengths, constraint (6) reduces to the exclusiveness constraint (5) that a line j cannot be assigned to twice. The capacity constraint (6) suits "part–whole" type $1:m$ matches (Figure 2c). Unfortunately, it cannot handle other $1:m$ cases such as "parallel edges" cases depicted in Figure 2b. In constraint (7), δ_j is a logical parameter for feature j which equals 1 if j has no nearby feature within a "cutoff" distance, and 0 otherwise. Constraint (7) maintains that if a feature j is not isolated, j must "contain" at least one of the nearby features. Constraint (7) enforces a minimum level of assignment to prevent the optimization model generating an empty matching plan as the "optimal" solution.

This optimization-based conflation approach has several advantages over traditional methods. First, it avoids a sequential or iterative approach in traditional methods, which may be sub-optimal. There is no need to resolve conflicts between iterations of assignments because the model simultaneously makes all matches in one pass. Second, the optimization-based approach is suitable for automated conflation without needing human input (for anchor points). In contrast, automatic computation, at times, may not have considered external factors such as the name of the streets or a satellite image beneath the GIS layer that can be utilized by human experts. On such occasions, the optimization approach allows human intervention in the conflation process. If a human expert has decided that road a should be matched to road b, this external knowledge can easily be incorporated by adding a constraint $x_{ab} = 1$ (or a constraint $x_{ab} = 0$ if a cannot possibly be a match of b). The changed model will generate a constrained optimal solution that is compatible with human input. Another major advantage of the optimization approach is clarity. The conflation problem is represented *declaratively* as a few lines of objectives and constraints [e.g., (1) through (5)]. The procedural aspects of finding the optimal matching are left to well-established algorithms like the simplex algorithm in off-the-shell solvers (e.g., IBM ILOG CPLEX, GNU GLPK). Such a succinct description is appealing as it facilitates effective communication in research and in education.

Since the first experiments over three decades ago, usage of spatial data conflation has been low. By comparison, other spatial computations, such as shortest path and spatial queries with the 9-intersection model, are in much wider use and available in virtually all GIS systems. Current conflation methods have a number of shortcomings that limit their use. These include sub-optimality, and the inability to consider spatial context, as discussed earlier. In addition, a more fundamental issue is the lack of a clear definition of the conflation problem. Formulating the conflation problem as a global optimization problem can potentially overcome some shortcomings of conventional approaches. However, existing optimal methods are rather limited. Essentially, the assignment problem formulation is the only effective optimal method to date. Despite its advantages mentioned earlier, the assignment problem also has significant limitations.

The first of these is its stringent assumptions. Historically, the assignment problem was a huge success in optimal job assignment as it avoided the alternative method then, namely brute-force enumeration. However, the job-assignment design essentially constrained its use case to one-to-one matchings, making it unsuitable for many-to-many relations (Figure 2). Beyond the simple one-to-one matching case, the assignment problem method

performs poorly. Consider the extreme case in Figure 2e, which contains mixed $1:m$ and $m:1$ relations in both directions. One can verify that $1:m$ relations in at least one of these two directions will be missed by the assignment formulation [Equations (1) through (5)]. Due to the difference in data conditions, Tong et al. (2014) found using their data that the assignment model achieved a low correct match rate of 56.5%. They had to develop a hybrid conflation method that used logistic regression to match features in many-to-many cases heuristically. Li and Goodchild (2011) attempted to handle both directions of multi-assignment by employing two sub-models, each being an assignment model for one direction of match. This approach is potentially flawed, however, because the two sub-models could be inconsistent and generate conflicting assignments, as will be discussed later.

The second limitation of the existing models is their inflexibility in structure. For example, when Li and Goodchild (2011) extended the assignment problem by adding the capacity constraint (6), one subtle consequence was that the modified model was no longer an LP problem. Instead, it required integer programming and could take longer to solve due to the associated tree-search (branch and bound) process. In addition, Li and Goodchild (2011) employed a computationally expensive local transformation phase on top of the optimization model. The solution times reported by Li and Goodchild (2011) range from 20 min to 2 hr for matching pairs of datasets with 100 to 400 edges each. Such long computation times could render the method impractical in real applications.

A more serious issue with this modified model is infeasibility. One can verify that for the example in Figure 2b, the right-to-left assignment sub-model [(1), (5), (6), and (7)] will be infeasible. This is because for both of the parallel features $I1$, $I2$ in dataset 1, line $J1$ in dataset 2 is the only compatible feature or possible match (assuming no other feature exists in the vicinity). By constraint (7), both $I1$ and $I2$ in dataset 1 must "contain" $J1$. In other words, $J1$ belongs to both $I1$ and $I2$. This violates constraint (5) that $J1$ can "belong to" (be assigned to) one feature at most. Depending on how big the cutoff distance is, other match cases can be infeasible. For example, Figure 2c can become infeasible if the cutoff distance is so big that the shorter lines $I1$ and $I2$ can each "contain" $J1$.

Of course, one can circumvent the infeasibility issue by using a very large cutoff distance, therefore reducing the likelihood of forcing multiple features ($I1$,$I2$) to contain one common feature ($J1$). But this will only force one of them to be assigned to a distant feature artificially, thereby forcing the model to produce false matches. Li and Goodchild (2011) did not specify the cutoff value used by their model, but the effect of cutoff distance needs further investigation. Due to their potential shortcomings, optimal conflation methods have seen limited applications in practice. They are classified only vaguely as the "Opt" algorithm in recent reviews and research articles (Li & Goodchild, 2011; Xavier, Ariza-López, & Ureña-Cámara, 2016).

3 | METHODOLOGY

Optimization is not automatically a panacea for solving conflation problems. The result of an optimal conflation model is reasonable only if its objective and constraints are sound reflections on the match relation in question. Traditional optimal conflation models use structural constraints that are too strong in many cases. This article proposes to explore more powerful (yet still efficient) optimization models from operations research for tackling the conflation problem. A good candidate is the minimum network flow problem (network flow problem for short hereafter).

The network-flow model is another classic optimization model in operations research, with many applications in transportation, logistics, management science, and natural resource management. It has a flexible model structure and it is possible to write many other optimization models as its special case models. This includes the assignment problem used in current optimal conflation models, the shortest path problem, and the maximum flow problem, among others. The relationship between the network flow problems and some well-known optimization models is depicted in Figure 3.

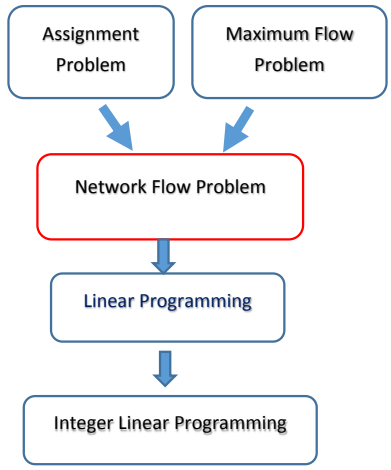

FIGURE 3 Theoretical links between optimization models

The network flow problem itself is a special case of the general LP problem (which in turn is a special case of the integer linear programming, ILP problem). The network flow problem can be formulated as an LP with a simple structure as follows:

$$\text{Minimize} \sum_{e \in E} c_e f_e \tag{8}$$

subject to:

$$\sum_{e \in I_n} f_e - \sum_{e \in O_n} f_e = 0, \text{for each } n \in N \tag{9}$$

$$l_e \le f_e \le u_e, \text{for each } n \in N \tag{10}$$

$$f_e = \text{amount of flow in } e \in E \tag{11}$$

where N,E are the nodes, edges of the flow network; I_n,O_n are the incoming, outgoing edges for n.

Given a directed graph $G = (N,E)$ with node set N and edge set E (see e.g., Figure 4), the decision variable (11) is the flow value f_e along each edge e. Each node corresponds to a feature to be matched, and each edge e corresponds to an assignment of one feature to another. Each edge e has an associated flow cost c_e, a lower bound l_e, and an upper bound u_e for the flow value as defined in Equation (11). The objective function (8) minimizes the total flow cost. Only constraint (9) is one of flow conservation and requires that the total amount of flow entering any node n must equal its outgoing flow. The network flow structure is flexible and expressive enough to formulate conflation problems. This article will focus on conflation problems with one-to-many (and, by definition, one-to-one) cardinality. These include the part–whole relation (Figure 2c) studied by Li and Goodchild (2011) as well as the new parallel edge relation (Figure 2b) that has not been studied before in optimal matching.

3.1 | The *p*-matching problem

We now present a first network flow formulation of the conflation problem, called the *p*-matching problem. The model aims to match p pairs of features while minimizing the total match error. An example flow network for the

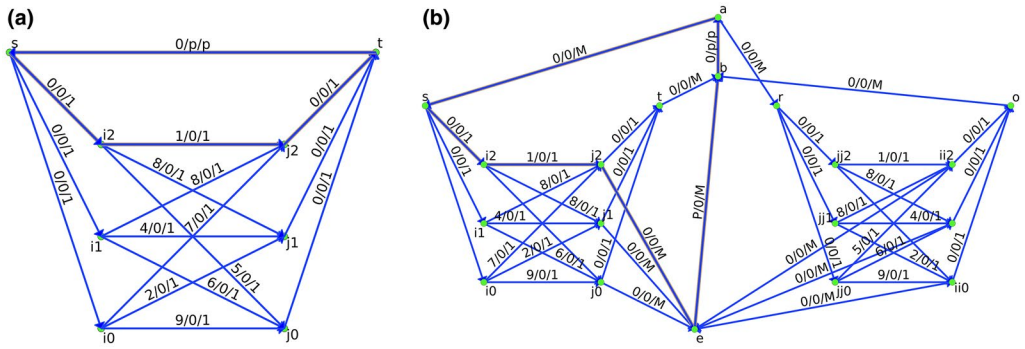

FIGURE 4 Conflation as network flow problems (*p*-matching problems), each arc in the flow network is labeled with its cost, lower bound, and upper bound of flow, respectively: (a) *p*-matching model; (b) bidirectional *p*-double-matching problem

p-matching problem is presented in Figure 4a. We represent the features in both datasets as network nodes in the left and right columns, respectively. We then add a directed edge from a feature in dataset *I* to a feature in dataset *J* if their Hausdorff distance is less than a given cutoff value (e.g., 100 m). The network flow cost for the edge is the Hausdorff distance between the two features. We add a "source" node *s* and special pseudo-edges from *s* to each node in dataset *I*, as well as a "target" node *t* along with pseudo-edges from each node of dataset *J* to *t*. The cost of any pseudo-edge is zero. The lower and upper bounds for all the above edges are 0 and 1, respectively, allowing at most one unit of flow. Finally, a flow balancing edge *ts* is added from the target node *t* to the source *s*, with a cost of zero, a lower bound of *p*, and an upper bound of *p*. With these parameters, we can write the *p*-matching problem either as a linear program using (8) through (11), or as a network flow file as defined by various optimization software packages.

It can be verified from Figure 4a that the balance link *ts* will force exactly *p* units of flow into the source *s*. The well-known integral theorem for network flow problems ensures that an optimal solution of the mathematical program [Equations (8) through (11)] always exists in which flow values are all integers. This property means that the network flow formulation will be able to match an entire feature (instead of fractions of it) to another feature. Without this property, one would have to write the flow variables as integer/binary variables, making the mathematical program a more difficult ILP.

There are several advantages of the network formulation of the conflation problem. First, it relieves the stringent assumption in the assignment problem that one dataset must correspond to a subset of the other dataset. More importantly, the network flow formulation provides a greater level of control. We constrain the number of feature pairs to match (*p*) as a measure to control the maximum allowable match error. Intuitively, the maximum pairwise match error will increase as one tries to match more feature pairs. A larger maximum error, as the experiments will show, means a greater chance of features being incorrectly matched (i.e., false positive matches). The choice of *p* reflects a fundamental balance in the feature-matching problem. On the one hand, we want to match as many feature pairs as possible in order to be effective (i.e., a larger *p*); on the other hand, we want to limit the match error to a reasonable level (i.e., a smaller *p*). The parameter *p* serves both as a measure of the success of the match and as a proxy for controlling match errors.

The value of *p* has to be determined by externally using a search procedure. During the search, we solve a set of *p*-matching problem instances with a series of increasing *p* values. We can choose a best value *p* that matches as many pairs of features as possible without causing a sharp increase in matching error. Alternatively, we can pre-specify a threshold matching error and use a search process to find the maximum number of features that can be matched without exceeding the threshold. In this article, we adopt the latter strategy. To avoid enumerating every possible value of *p*, we employ a binary search process, as described in Section 3.4.

3.2 | The *p*-double-matching problem

The *p*-matching problem is suitable for solving one-to-one matching problems. We can extend the network flow formulation into a bidirectional *p*-double-matching problem to handle the more complicated two-way 1 : *m* matching problems. This involves several modifications of the *p*-matching problem.

The first modification is to use a semi-distance measure, such as the directed Hausdorff distance, to allow partial matchings in part–whole relations. This is similar to the work of Li and Goodchild (2011). The directed Hausdorff distance is a directional (semi-)distance measure, which measures the maximum deviation of the points on a (usually smaller) feature from a (larger) feature (e.g., the assignment $I_1 \rightarrow J_1$ in Figure 2c). In the extreme case when the smaller feature coincides with a part of the larger feature, the directed Hausdorff distance will be zero indicating a perfect partial match. However, the Hausdorff distance (representing total, unidirectional discrepancy) will be large, indicating for example that $I_1 \rightarrow J_1$ is not a valid one-to-one match. For the partial match case in Figure 2c, the two parallel lines I_1,J_1 are 50 m apart from each other. The directed Hausdorff distance from $I_1 \rightarrow J_1$ is indeed 50 m (from the smaller feature to the larger one). In the opposite direction, the directed Hausdorff distance $J_1 \rightarrow I_1$ is 158 m. The undirected Hausdorff distance is 158 m, the larger of the two directed distances.

The second modification of the *p*-matching problem is to create a reverse network of the *p*-matching network (Figure 4b). The reverse network consists of a mirror image of the original network (which is called the forward network hereafter). The reverse network optimizes opposite assignments from dataset *J* to dataset *I*. Each edge $i \rightarrow j$ in the forward network is replicated in the reverse network as an edge $jj \rightarrow ii$. An example of the mirror network for the network in Figure 4a is depicted in Figure 4b as the subnetwork from *r* to *o*. One can then "concatenate" the two mirror networks by adding a new source node *a*, and a sink node *b*.

Since the new model can match multiple features to one feature, the outgoing edge leaving each node *j* or *ii* should have a capacity greater than one. But instead of increasing the capacity of outgoing edges, we keep the unit-capacity outgoing edges and create a new set of surplus links from each outgoing node (*j* or *ii*) to a new "excess" node *e* that receives surplus flows. Each excess link has a cost of zero and a capacity of *M*, which is a suitably big integer. The excess node *e* then discharges all surplus flows via a special excess link into the sink node *b*. The excess link has a non-negative cost of *P* and a capacity of *M*. The two subnetworks of the new network have almost the same structure as the *p*-matching network, except that they have the "excess" node *e* and surplus edges associated with it.

The third modification pertains to the coordination between the two mirror networks. So far, the two subnetworks (Figure 4b) essentially determine the assignments of the two opposite directions separately. While these assignments are compatible most of the time, they can be inconsistent with each other at times. An example of inconsistent assignments is depicted in Figure 5a using site #5 of the test data to be described in Section 4. In the figure, *I*57 and *J*61 are assigned to each other by the forward and reverse networks, indicating that they belong to each other. Feature *J*59 is assigned to I59; and *I*59 is assigned to *J*61. However, this match solution is inconsistent and impractical for the following reasons. Following the assignment arrows in the figure, *J*59 is linked to *I*59 by the reverse network, and to *J*61 by the forward network, and then to *I*57 by the reverse network. In this trail of assignments, each of the two edges *I*59 and *I*57 in dataset *I* is related to every one of the two edges *J*59 and *J*61 in dataset *J*. The match relation depicted is not well-defined, and is confusing because it relates everything in *I* to everything in *J* in a small neighborhood. Operationally, the subsequent feature-merging operations would not be able to transfer attributes between matched features because of the confusion. A well-defined, consistent solution, depicted in Figure 5b for example, assigns *I*59 to *J*59 and *I*57 to *J*61.

3.2.1 | Detecting inconsistent assignments

The inconsistent assignments described in Figure 5a are intuitively undesirable. But how can they be detected operationally? Comparing the two solutions in Figure 5, one can find that the assignment from *I*59 to *J*61 is

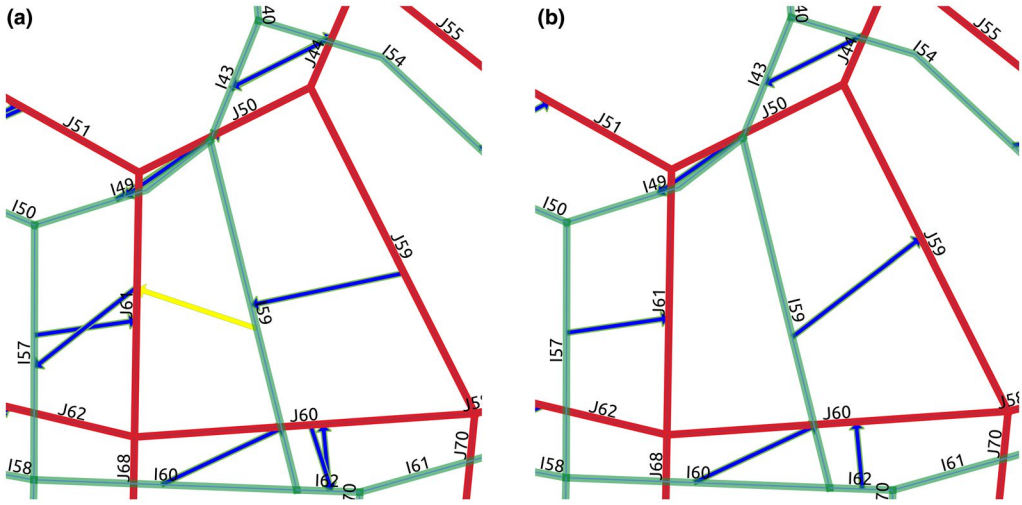

FIGURE 5 Coordination in the *p*-double-matching problem: (a) conflicting assignments; and (b) corrected assignments. Arrows represent model-generated matches. An inconsistent arrow from element I59 to J61 is highlighted

problematic and redundant. Without it, the two one-to-one assignments *I57* → *J61* and *I59* → *J59* are unambiguous and valid. The cause of redundancy is related to the number of times a feature is assigned to, and assigned from, in a match plan. In Figure 5, the problematic assignment *I59* → *J61* has multiple assignments on both ends. That is, *I59* is related to two elements (*J59* and *J61*). So is *J61* (related to *I57* and *I59*). In other words, the problematic assignment is redundant because if this assignment is removed from the match plan, none of the matched features will become isolated. More formally, this condition can be described in the following observation:

> *Observation 1:* In a match plan, the assignments of features between two datasets form a match graph, in which the data elements are nodes and assignments are (directed) edges. The assignment plan is *inconsistent* if it contains an edge for which each of its nodes has degree greater than 1.

3.2.2 | Removing inconsistent assignments

We remove the inconsistent assignments as follows. First, we identify each "clique," which is defined to be a smallest connected component in the match graph containing inconsistent assignments. For each clique, we select *k* assignment edges from it, forming a so-called "elimination" set. We then tentatively remove the elimination set from the clique. We find a successful resolution of the clique if: (a) its elements become non-redundant after removing the elimination set; and (b) no element in the clique becomes unassigned.

Technically speaking, every pair of opposite assignments forms a redundant set by the definition of Observation 1. But it would be an overkill to resort to graph-theoretic measures to remove such redundancies. Therefore, before detecting and removing inconsistent assignments, we first remove one of the two assignments in a pair of opposites. After this step, there are typically few (truly) redundant assignments left. Theoretically, there can be multiple valid elimination sets for a given clique. In this article, we adopt a simple strategy that turns out to be satisfactory: we implicitly enumerate the elimination set and use the first valid one to resolve the clique as follows. We sort the assignment edges in descending order according to the assignment cost. If a valid one-element elimination set is found, we use it and stop; otherwise, we go through two-element elimination sets in the dictionary order of the assignment costs of their elements; then three-element sets and so on, if necessary.

3.2.3 | Penalizing redundant assignments

Detecting and removing redundant assignments can help reduce false assignments and obtain valid match plans. Certain types of redundant assignments can also be avoided in the modeling phase. In Figure 5, the problematic assignment $I59 \rightarrow J61$ can happen because the forward and subnetworks make assignment decisions independently. *J*59 is assigned to *I*59 because *I*59 is the closest feature for *J*59. However, the two features are not necessarily the closest feature for each other. In fact, a feature other than *J*59 (namely *J*61) is the closest feature for *I*59. Consequently, *J*61 is related to multiple edges (*I*59 and *I*57) incorrectly.

According to this observation, we can discourage such multiple assignments by adding a significant penalty for them. We can make the cost of assigning multiple edges to one edge greater by an additive constant *P* (to the cost of the excess link). With this penalty, it would now be more expensive for *I*59 to assign to *J*61, which could potentially be assigned to by other edges. By comparison, the objective function would be smaller if one makes assignment to "unused" edges first (e.g., *J*59). Figure 6 demonstrates a correct match plan for the above mentioned edges, which is generated from a *p*-double-matching model with a penalty value of 100 m. The proposed method adopts both methods to cope with redundant assignments: we use the penalty on excess links to discourage redundant assignments in the modeling stage and then use the logic described earlier to detect and remove any remaining redundant assignments.

3.3 | Incorporating direction of features

The directed Hausdorff distance can capture the point-wise deviation of a feature from another. This is particularly effective in matching longer features: if a pair of long features have a small directed Hausdorff distance, it is likely that one of them belongs to the other. However, for shorter linear features, the Hausdorff distance falls short in selecting similar features. For example, a short linear feature that is perpendicular to another could have a small directed Hausdorff distance because a point confined in the short line cannot move far in the first place. In order to avoid matching small features that have very different orientations, we add the difference of directions into the distance measure and define a composite distance measure as follows.

First, we estimate the direction of a linear feature by finding the best-fit line using ordinary least squares (OLS) estimation. Second, for two linear features, we use the difference between the inclination angles of the two fit

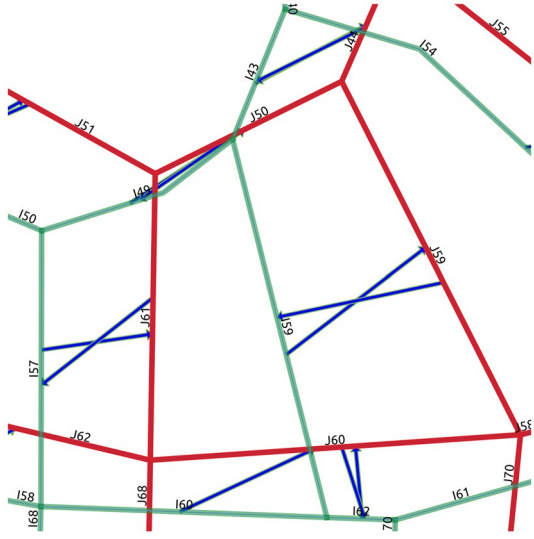

FIGURE 6 Penalizing multi-assignment in the *p*-double-matching problem

lines as an angular distance between the two features. Third, we add the angular distance to a linear distance after appropriately converting degrees to meters. In the matching process, this effectively penalizes feature pairs that are very different in orientation. In this article, we convert a 90° difference in direction (which is the maximum possible) to 100 m equivalent of distance.

3.4 | Overall work flow

To summarize, the overall work flow of the proposed method is as follows:

1. Start a binary search by setting the lower bound $l = 1$ and the upper bound $u = |I| + |J|$, where $|I|$ and $|J|$ are the numbers of features in datasets I and J, respectively.

Set $p := (l + u)/2$.

2. Use Python to write the p-double-matching problem for the p value and call the CPLEX solver to solve it.
3. If the model is infeasible, set $u := p$ and go to **step 1**.

Otherwise (the model is feasible):

if $l = u$ then

go to **step 4**

else

set $l := p$ and go to **step 1**.

4. Apply the logic for "removing inconsistent assignments" in Section 3.2.

The above procedure uses a binary search method and repeatedly calls the CPLEX solver to solve the p-double-matching problem for different p values until the maximum feasible p is found. It then applies the redundancy reduction logic. Note that the binary search procedure will call the CPLEX solver at most $log\,(|I| + |J|)$ times before finding the maximum feasible p value.

4 | COMPUTATIONAL EXPERIMENTS

The conflation models presented earlier are implemented in IBM ILOG CPLEX Studio 12.8. The binary search procedure and redundancy resolution logic are implemented in the Python programming language. The test machine is a computer with an Intel Core i5-6300U CPU at 2.40 GHz and 4 GB memory. For comparison purposes, we have used the same test data as Li and Goodchild (2011), kindly provided by those authors. The test data consist of six test sites covering study areas in Santa Barbara, CA (Figure 7). Each site consists of a pair of datasets from two sources. The first source (dataset 1) is the TIGER/Line (TG) data from the U.S. Census and the second source (dataset 2) is the TeleAtlas (TA) data. The spatial accuracy for both data sources is 10 m in urban areas and 50 m in rural areas (Li & Goodchild, 2011). Both datasets are in shapefile format. Each road segment is represented as a single polyline most of the time. However, in both datasets, in a (small) number of cases, the opposite directions of a major road are represented as two or more polylines (e.g., Figure 1a).

We have also used the ground truth data of Li and Goodchild (2011), which consists of two tables matching line IDs from dataset 1 to the IDs of the corresponding lines in dataset 2, and vice versa. The number of linear features of each pair of datasets for the test sites are shown in Table 1. The datasets for the six test sites are also depicted in

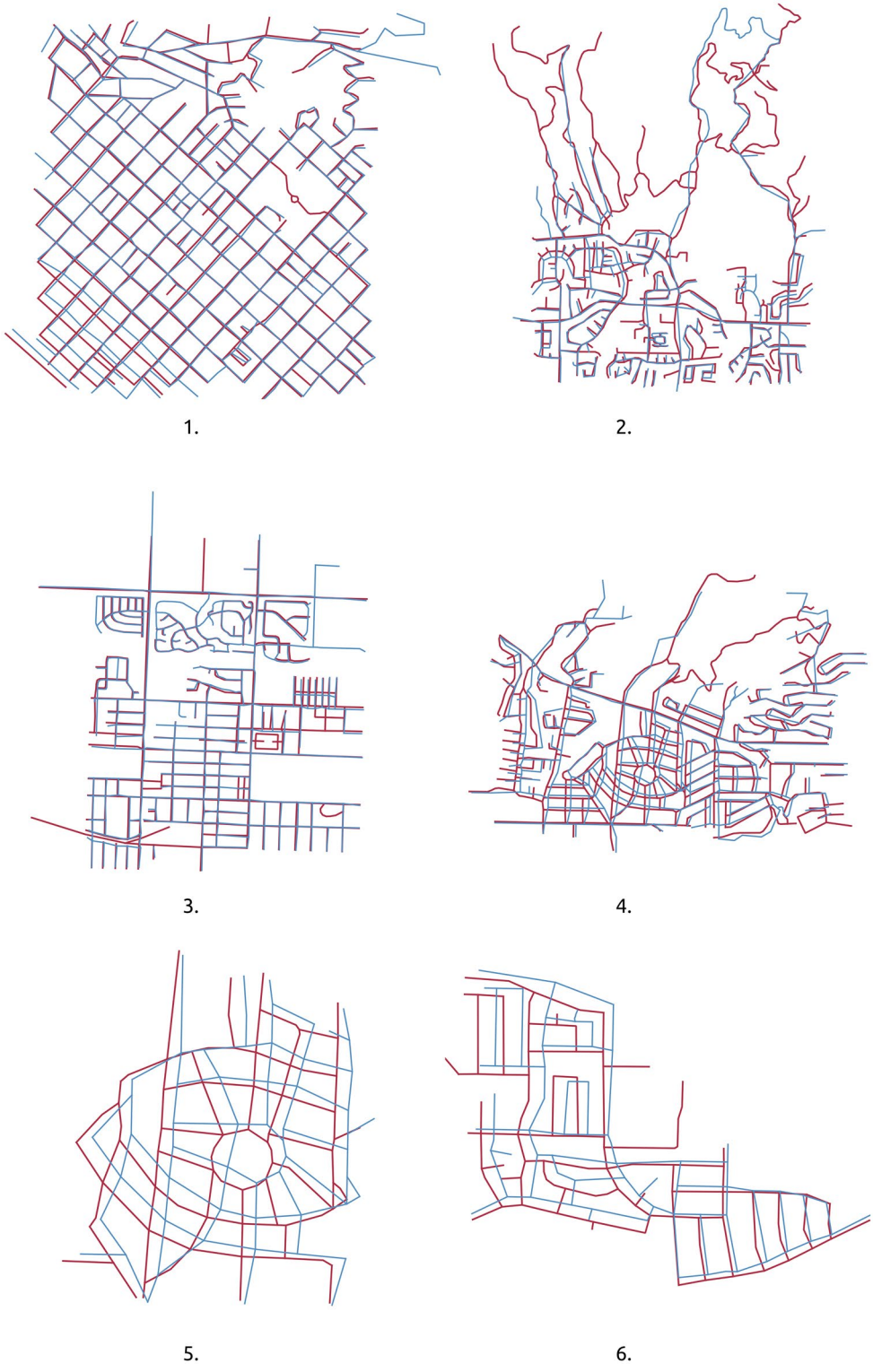

FIGURE 7 Test sites for the computational experiment (Li & Goodchild, 2011)

Figure 7. It should be noted that other test datasets can generally be used for testing conflation methods. One example is the dataset of Xavier, Ariza-López, and Ureña-Cámara (2017), built up from maps by official Spanish mapping agencies.

4.1 | Evaluation criteria

The method we use to evaluate the accuracy of our conflation algorithm is as follows. We count the number of true and false matches by comparing match results with the ground truth data. Each match from feature i to feature j in the other dataset is called an arrow a_{ij}. If a_{ij} is present in the match result as well as the ground truth, it is considered a true match (TM). In order to avoid double-counting a 1 : 1 match as two matches, we remove one of the two opposite arrows in the 1 : 1 match before comparing the arrows. For consistency, we choose to remove the arrow from dataset 2 to dataset 1 on such occasions. If feature i has no match (or is an *unmatched* feature) in the ground truth and is unmatched in the match result, the case is counted as a true unmatch (TU).

In the above two cases, the conflation algorithm correctly matches a feature or leaves it unmatched (per ground truth). However, if a feature is matched to something in the ground truth but unmatched in the generated match plan, the match is counted as a false negative (FN). If a feature is matched by the algorithm to a feature that is either different from the ground truth value or absent in the ground truth, the case is counted as a false match (FM). Based on these definitions, we use two evaluation criteria.

4.1.1 | True match rate

In accordance with the work of Li and Goodchild (2011), we use the true match rate defined as:

$$TMR = TM/AM \tag{12}$$

where TM is the number of true matches and AM is the total number of match arrows in the ground truth. This measure reflects how many of the matches in reality are captured by the conflation algorithm.

4.1.2 | Accuracy

The true match rate (TMR) alone does not account for the false matches. Hypothetically, a match algorithm with a high TMR could indiscriminately generate a large number of false arrows (false matches). To take into account the discriminative power of the match algorithm, we also use the accuracy, defined as:

$$ACC = (TM + TU) \,/\, (TM + TU + FM + FN) \tag{13}$$

TABLE 1 Number of features in each dataset for the six test sites

Test site	Features in dataset 1	Features in dataset 2
1	434	423
2	308	264
3	377	374
4	344	322
5	81	80
6	84	77

4.2 | Performance results

As mentioned, we implemented the proposed network flow-based conflation models in CPLEX 12.8 using its Python programming interface. For comparison purposes, we also re-implemented the Li and Goodchild (2011) model (called the LG11 model hereafter) in the same environment. A few notes about the re-implementation are in order. First, the LG11 model constitutes two sub-models for assignments in the two opposite directions. Other than the direction of assignments, the two sub-models are the same. Therefore, we only implemented one LG11 model and applied it to the ordered pair of datasets (TG, TA). We then reversed the order of the data pair (i.e., TA, TG) and applied the same model again, to accomplish what the second sub-model does.

Second, we did not implement the local transformation steps of the LG11 method. More specifically, the LG11 method consists of not only an optimization model, but also a local affine transformation phase similar to the conventional rubber-sheeting method described in Section 2. We did not implement this phase, for simplicity and a better understanding of the optimization aspect of conflation. In contrast, just like the LG11 model, the proposed optimization model can easily be combined with a conventional rubber-sheeting or local transformation method.

Figure 8 presents the results of matching on test site 1 for the *p*-double-matching problem, the LG11 model, and the reversed LG11 model (in which assignments are from dataset 2 to dataset 1). Figures 8a and b present the TMR and accuracy, respectively. The match rates are plotted against cutoff distances at 10, 25, 75, 100, 150, 200, 300, 400, 500, 600, 700, 800, 900, and 1,000 m, respectively. Unless stated otherwise, we use a penalty of 100 m for redundant matches. In Figure 8a, when the cutoff distance is less than 400 m, the LG11 model (blue) is infeasible (with a rendered match rate of zero), for reasons discussed in Section 2. When the cutoff distance is greater than or equal to 400 m, the TMR for the LG11 model was almost constantly 82.6%. When the LG11 model is applied in the opposite direction (black curve), the model is infeasible for all tested cutoff distances. By comparison, the *p*-double-matching model does not suffer from such anomalies. It achieves a low TMR of 27.1% with a 10 m cutoff distance, which is apparently too small and causes a large number of false negatives. Then, the TMR rises to 83.1, 92.5, and 98.6% at the 25, 50, and 75 m cutoff distances, respectively. The TMR for *p*-double-matching saturates at 99.3% with cutoff distances that are greater than or equal to 100 m.

The accuracy curves in Figure 8b reveal a different aspect of the generated matches. For the LG11 model (blue), the match rates were around 75.3% with cutoff distances of 300 m or greater, about 4% lower than the TMR. This difference is due to the fact that the "accuracy" measure takes into account false matches. The accuracy curve for the *p*-double-matching model (magenta) is more interesting. It rises from 19.9% at 10 m to 72.1% at 25 m, to 85.8% at 50 m, and then peaks at 94.3% at 75 m, dropping gradually to a level of 90.9%. The decrease in accuracy after the peak at 75 m is due to two factors. First, it is because the proposed model currently tries to find the maximum number of features (*p*) to match. As the cutoff distance increases, more potential matches are being considered, which results in higher numbers of false matches (FM) even after the redundancy removal step in Section 3.4. Second, some truly unmatched (TU) features may be incorrectly matched to other features due to the increased range of search. Both the higher FM (i.e., lower TM) and lower TU contribute to the decrease in accuracy (ACC) in Section 4.1.2. We tested more aggressive methods that stop increasing *p* at the first occurrence of redundant matches. However, the alternative algorithm often stops too early and leaves a large number of features falsely unmatched (FU). Finding a more intelligent stopping criterion in the work flow (Section 3.4) is an area worthy of future research and under ongoing investigation. For the first site, the network-flow model is very effective at cutoff distances of 75 to 100 m, and outperforms the LG11 model by a large margin (over 10%).

Figure 9 presents the match results for test site 5. For this site, the TMR for both the LG11 model and the *p*-double-matching model is 100%, when the cutoff distance is greater than 100 m. This means that all the true matches (TM) are captured. This is in accordance with the results of Li and Goodchild (2011). The accuracy for both models is 97.5%. The lower accuracy rate is due to two false matches in addition to 79 true matches. The remainder of the match rates are shown in Figures A1–A4.

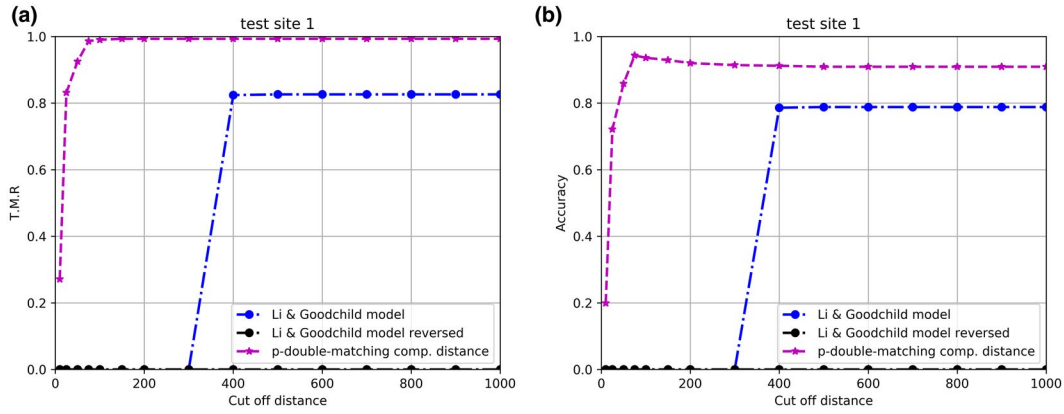

FIGURE 8 (a) True match rate; and (b) accuracy (Section 4.2) of the proposed model on test site #1

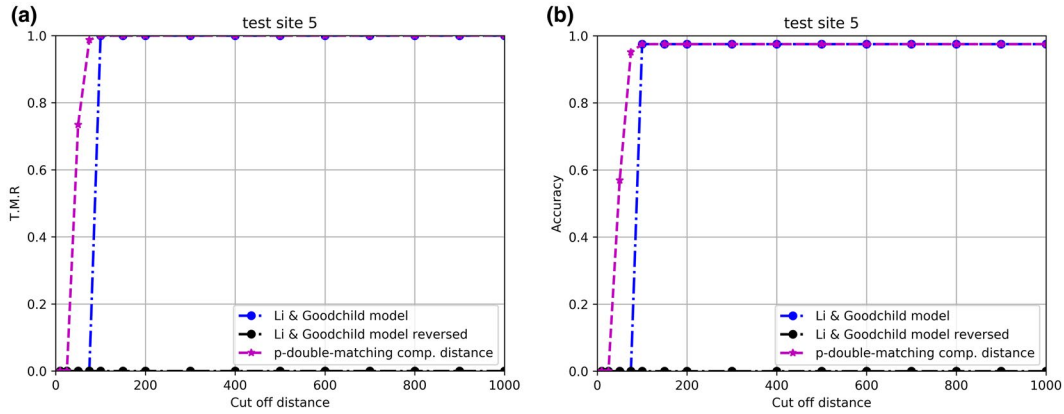

FIGURE 9 (a) True match rate and (b) accuracy (Section 4.2) of the proposed model on test site #5

The agreement of LG11 and the proposed model on test site 5 is largely due to the layout of the two datasets in the match (Figure 10). As shown in Figure 10 and Table 1, the two datasets (from TA and TG) have almost the same numbers of features (81 vs. 80). There is not much ambiguity about candidate matches, because almost every feature is closest to its true match. On more "difficult" test cases, however, the proposed model outperforms the LG11 model as it overcomes some of the limitations of the LG11 model discussed previously.

The match rates for all test sites are shown in Table 2, when the cutoff distance is 100 m. Overall, the average TMR is 97.5%, and the average accuracy is 91.2%. The match rates for test site 2 are lower than for the other sites. This is because the test site had a relatively large number of unmatched features in periphery regions in the ground truth, which were incorrectly matched to nearby features by the model. Such cases indicate the need for a better measure, both for the proposed model (and existing models), to stop making assignments at an appropriate level or to discriminate unlikely candidate feature pairs. We also noticed instances in test sites 2 and 5 where the model found correct matches that were missing in the ground truth. We did not change the ground truth data for comparison purposes.

As discussed earlier, the penalty value in the proposed model discourages redundant matches by adding a cost to the objective whenever such a match is made. A default value of 100 m is chosen in this article, based on the average locational map accuracy in the two dataset and empirical tests. We also tested the effect of different penalty values. Figure 11 presents the TMR and accuracy for the proposed model with a small penalty factor of

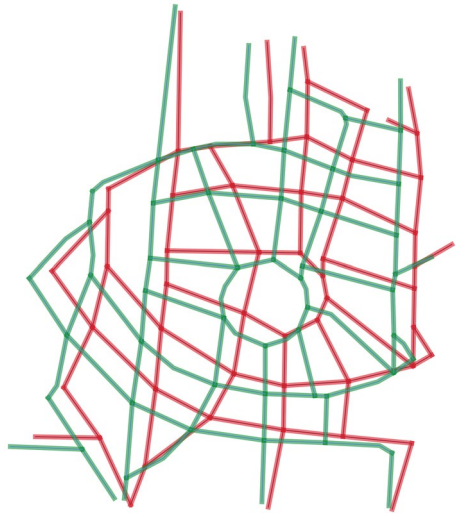

FIGURE 10 Road networks in test site #5

TABLE 2 True match rate and accuracy for six test sites with 100 m cutoff (penalty value of 100 m)

Test site	True match rate (%)	Accuracy (%)
1	99	93.6
2	93.6	83.8
3	99.2	90.7
4	98	89.5
5	100	97.5
6	95	91.9

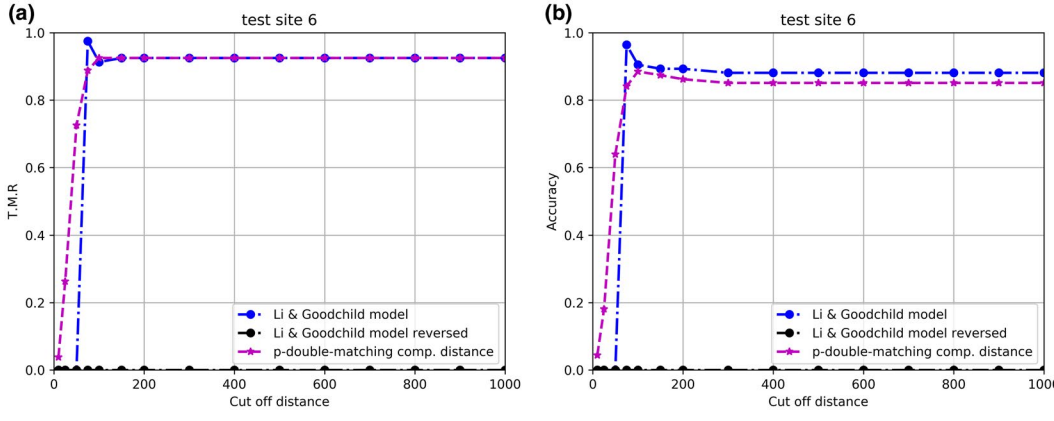

FIGURE 11 (a) True match rate; and (b) accuracy (Section 4.2) of the proposed model on test site #6 with a lower penalty value (25 m)

25 m. Compared with the results in Figure 4b for the same site (but with the default 100 m penalty), the shape and trend of the TMR and accuracy curves are almost the same. The accuracy rates have dropped slightly in Figure 11 (about 2–3%). For all other test data, the variant of the model with the lower penalty value has slightly lower but similar performance. We also tested a variant of the model with a higher penalty factor of 200 m. The accuracy rates for both variants of the model at 100 m cutoff are shown in Tables 3 and 4, respectively. From the two tables, we can observe that generally speaking, increasing the penalty factor results in higher TMR and accuracy. When the penalty factor is increased from 25 to 100 m, the TMR for site 5 has increased from 92.5 to 95%, and the accuracy has increased from 88.5 to 91.9%. However, for site 3, the TMR and accuracy have dropped by 0.5 and 0.2%. When the penalty factor is increased to 200 m (Table 4), all the rates are the same as for the base case (100 m penalty), except for an improvement of 0.3% in TMR and 0.4% in accuracy for site 1.

Overall, the figures and results indicate that the network-flow-based *p*-double-matching model outperforms the LG11 model by a large margin (except for test site 6, in which the two models have similar performance). The proposed method is also efficient. For example, the network-flow model plus the redundancy resolution logic took 2.7 s to run on the largest dataset at site 1, and 1.8 s for the smallest test site (site 5), on average over three runs. This is tested with a cutoff distance of 100 m on three average runs. The solution times for the two sites with larger cutoff distances are shown in Figure 12. The solution times for other sites show a similar pattern, and are omitted here for space reasons. We can observe that there is an increasing trend in solution times with larger cutoff distances. For test site 1, the solution time increased from about 3 to 40 s when the cutoff distance increased from 100 to 1,000 m. The solution time for the smaller test site 5 increased from about 1 to 4 s. A similar trend can be observed for the LG11 models. This increasing trend is due to the fact that with larger cutoff distances, more edges are included in the network of the proposed model.

It should be noted that the solution times for the re-implemented LG11 models are lower than for the proposed model, and the solution times here (generally below 10 s) are much smaller than the times reported in the LG11 paper. There are several possible reasons for the apparent difference. First, as mentioned earlier, we did not re-implement the full workflow of the LG11 paper. We did not implement the iterative local affine transformation

TABLE 3 True match rate and accuracy for six test sites with 100 m cutoff (penalty value of 25 m)

Test site	True match rate (%)	Accuracy (%)
1	98.8	93.2
2	93.6	83.3
3	99.7	90.9
4	98	89.0
5	100	97.5
6	92.5	88.5

TABLE 4 True match rate and accuracy for six test sites with 100 m cutoff (penalty value of 200 m)

Test site	True match rate (%)	Accuracy (%)
1	99.3	94
2	93.6	83.8
3	99.2	90.7
4	98	89.5
5	100	97.5
6	95	91.9

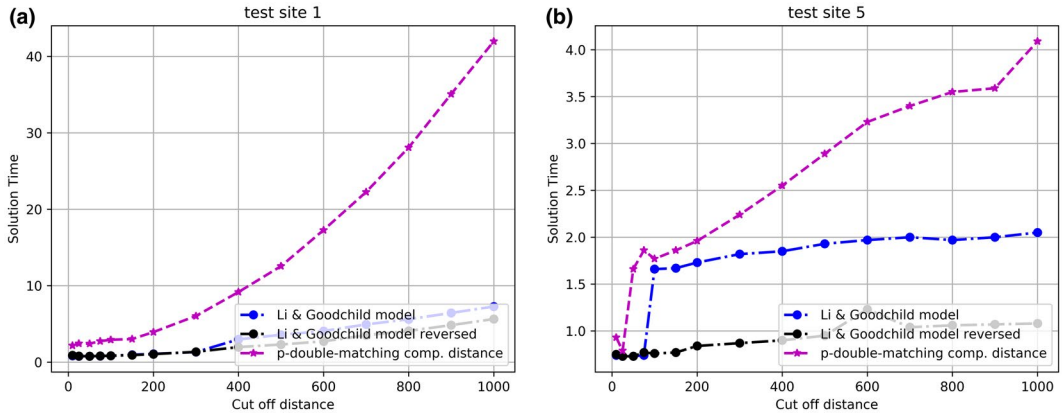

FIGURE 12 Solution times for the proposed model on: (a) test site #1; and test site #5

loop as it is not related to optimization and the iteration scheme described in LG11 may be too expensive. Second, our re-implementation of the LG11 model is more efficient because in implementing constraint (7), we do not generate all constraints and use the parameter δ_j to relax a subset of the constraints (for which $\delta_j = 1$), as LG11 did. Instead, we only write the constraints for which $\delta_j = 0$. Therefore, we avoid having to enter distances between all pairs of features into the optimization model. Instead, we only enter distances for the feature pairs that are within the cutoff distance from each other. Third, differences in computer hardware and software may contribute to the difference in solution time.

5 | CONCLUSIONS

Traditionally, conflation has been deemed a complex process that should be performed by specialists and GIS units in large organizations. In this article, we develop a novel and efficient conflation method that can potentially be used by the average GIS user, researchers, as well as large organizations. We propose to formulate the feature-matching problem as an optimization problem using the classic network flow problem in operations research. As demonstrated in the article, the matching of features can naturally be posed as an optimization problem that seeks to minimize the total discrepancy between two datasets. This optimization-based approach has several advantages:

1. It can overcome the sub-optimality in procedural and bottom-up approaches in conventional methods.
2. Its modeling language provides a formal specification of the criteria and requirements of conflation that is easy to understand, communicate, and adapt.
3. Expert knowledge can be incorporated easily as additional constraints to the model.
4. The method is easy to implement based on standard operations research solvers and can be incorporated into GIS (Lei, Church, & Lei, 2016).

This article contributes to optimal conflation research in the following ways. First, we analyzed the limitation of existing optimization-based conflation methods. We began with an analytic review of related research on optimization-based conflation methods. From the review, we found that existing optimal conflation methods are limited in terms of function and performance. Existing optimal conflation models are primarily based on the assignment problem and its extensions thereof. The assignment model for conflation falls short of solving the general conflation problem because of its stringent assumptions and inflexible model structure. Efforts to extend the assignment problem to date have

potential issues as well. These include potential infeasibility in certain cases and the increased complexity caused by the addition of certain constraints and the dependence on the rubber-sheeting process.

To overcome the issues of existing optimal conflation, we propose an alternative optimization model for the conflation problem, namely the network flow problem, which is computationally efficient and has more structural flexibility and a greater power for expressing the requirements of conflation. The proposed model focuses on solving the two-way $1:m$ matching problem. The model has a parameter p for the number of assignments, which is proportional to the maximum allowable match error.

Through analyses and experiments, we found that the two-way $1:m$ problem is more complex than the $1:1$ matching problem in that it requires establishing a "belongs-to" relationship in two directions of matching, and these two sub-problems, when solved independently, can generate inconsistent and redundant matches. We developed methods to avoid and remove the inconsistent assignments.

We also found that the cutoff distance for including a feature pair is an important parameter of the conflation model. When the cutoff distance is too small, the model can miss true matches and incur false negatives; when it is too big, the model can generate false match pairs that do not exist or are incorrect. For the test data, we found that the accuracy peaks when the cutoff distance is around 100 m. Computational experiments show that the proposed model outperforms existing conflation models by a large margin, and achieves satisfactory match rates on the same test data (with a TMR of 97.5% and an accuracy of 91.2%).

Several directions of research are worthy of investigation in future work. As mentioned above, an important decision in the model is when to stop the search for a good value of p, the number of assignments. The current approach is to stop increasing p when the maximum match error exceeds a prespecified cutoff distance. A more intelligent stopping criterion would certainly make it easier to apply to different datasets in practice. This could, for example, involve a statistical analysis of a sample of the data or the ground truth (if available). Ongoing work is underway in this direction. Another area worth exploring is more efficient implementation of the network-flow models, which can involve tightening the proposed model or using specialized network-flow code for faster solution.

ORCID

Ting Lei https://orcid.org/0000-0003-2385-9128

REFERENCES

Ahmadi, E., & Nascimento, M. A. (2016). *k*-Closest pairs queries in road networks. In *Proceedings of the 17th IEEE International Conference on Mobile Data Management*. Porto, Portugal: IEEE.

Blasby, D., Davis, M., Kim, D., & Ramsey, P. (2002). *GIS conflation using open source tools*. Retrieved from http://lists.jump-project.org/assets/JUMP_Conflation_Whitepaper.pdf

Brennan, P. (2008). *OWS-5 conflation engineering report* (OGC 07–160 r1). Retrieved from https://portal.opengeospatial.org/files/?artifact_id=30064

Brown, J. N., Rao, A. L., & Baran, J. (1995). Automated GIS conflation: Coverage update problems and solutions. In *Proceedings of the 1995 Geographic Information Systems for Transportation Symposium*. Reno, NV.

Canavosio-Zuzelski, R., Surratt, J., Bower, D., Governski, J., & Sorenson, M. (2015). Hootenanny: Web enabled geospatial vector-data conflation and map generation. In *Proceedings of the 2015 Annual ASPRS Conference*. Tampa, FL: ASPRS.

Church, R., Curtin, K., Fohl, P., Funk, C., Goodchild, M. F., Kyriakidis, P., & Noronha, V. (1998). Positional distortion in geographic data sets as a barrier to interoperation. In *Proceedings of the ACSM Annual Convention*. Baltimore, MD: ACSM.

Cobb, M. A., Chung, M. J., Foley, H. III, Petry, F. E., Shaw, K. B., & Miller, H. V. (1998). A rule-based approach for the conflation of attributed vector data. *GeoInformatica*, *2*(1), 7–35.

Corral, A., Manolopoulos, Y., Theodoridis, Y., & Vassilakopoulos, M. (2004). Algorithms for processing *k* closest-pair queries in spatial databases. *Data & Knowledge Engineering*, *49*(1), 67–104.

ESEA. (2018). *MapMerger user guide*. Retrieved from http://mapmerger.com/wpcontent/uploads/2015/09/MapMergerUserGuide.pdf

Esri. (2018). *An overview of the conflation toolset*. Retrieved from http://desktop.arcgis.com/en/arcmap/latest/tools/editing-toolbox/an-overview-of-theconflation-toolset.htm

Filin, S., & Doytsher, Y. (2000). Detection of corresponding objects in linear-based map conflation. *Surveying & Land Information Systems*, *60*(2), 117–128.

Goodchild, M. F., & Hunter, G. J. (1997). A simple positional accuracy measure for linear features. *International Journal of Geographical Information Science*, *11*(3), 299–306.

Lei, T. L., Church, R. L., & Lei, Z. (2016). A unified approach for location-allocation analysis: Integrating GIS, distributed computing and spatial optimization. *International Journal of Geographical Information Science*, *30*(3), 515–534.

Li, L., & Goodchild, M. F. (2010). Optimized feature matching in conflation. In *Proceedings of the 6th International Geographic Information Conference*. Zurich, Switzerland.

Li, L., & Goodchild, M. F. (2011). An optimisation model for linear feature matching in geographical data conflation. *International Journal of Image and Data Fusion*, *2*(4), 309–328.

Masuyama, A. (2006). Methods for detecting apparent differences between spatial tessellations at different time points. *International Journal of Geographical Information Science*, *20*(6), 633–648.

McKenzie, G., Janowicz, K., & Adams, B. (2014). A weighted multi-attribute method for matching user-generated points of interest. *Cartography & Geographic Information Science*, *41*(2), 125–137.

OSM. (2018). *OpenStreetMap Wiki JOSM conflation plugin*. Retrieved from https://wiki.openstreetmap.org/wiki/JOSM/Plugins/Conflation

Pendyala, R. M. (2002). *Development of GIS-based conflation tools for data integration and matching*. Tallahassee, FL: Florida Department of Transportation.

Rosen, B., & Saalfeld, A. (1985). Match criteria for automatic alignment. In *Proceedings of the 7th International Symposium on Computer-Assisted Cartography*. Washington, DC: ASPRS.

Ruiz, J. J., Ariza, F. J., Urena, M. A., & Blázquez, E. B. (2011). Digital map conflation: A review of the process and a proposal for classification. *International Journal of Geographical Information Science*, *25*(9), 1439–1466.

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

Tong, X., Liang, D., & Jin, Y. (2014). A linear road object matching method for conflation based on optimization and logistic regression. *International Journal of Geographical Information Science*, *28*(4), 824–846.

Walter, V., & Fritsch, D. (1999). Matching spatial data sets: A statistical approach. *International Journal of Geographical Information Science*, *13*(5), 445–473.

Xavier, E. M. A., Ariza-López, F. J., & Ureña-Cámara, M. A. (2016). A survey of measures and methods for matching geospatial vector datasets. *ACM Computing Surveys*, *49*, 39.

Xavier, E. M. A., Ariza-López, F. J., & Ureña-Cámara, M. A. (2017). MatchingLand, geospatial data testbed for the assessment of matching methods. *Scientific Data*, *4*, 170180.

How to cite this article: Lei T, Lei Z. Optimal spatial data matching for conflation: A network flow-based approach. *Transactions in GIS*. 2019;23:1152–1176. https://doi.org/10.1111/tgis.12561

APPENDIX

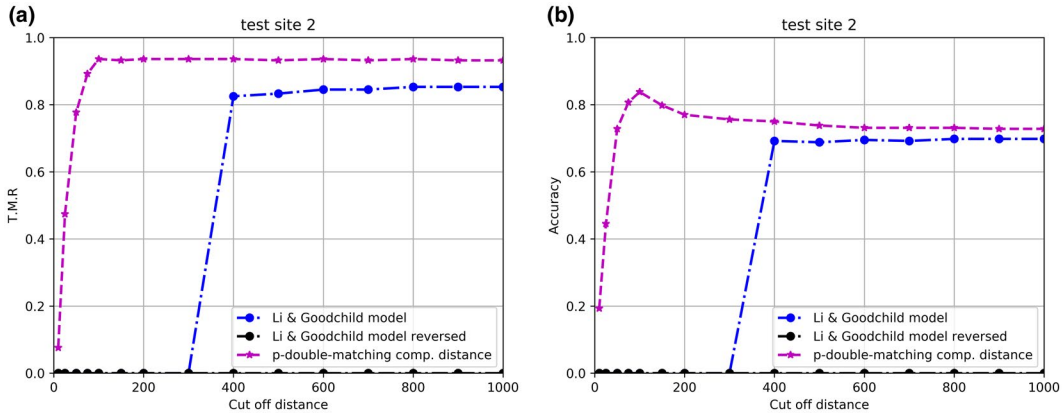

FIGURE A1 (a) True match rate; and (b) accuracy (Section 4.2) of the proposed model on test site #2

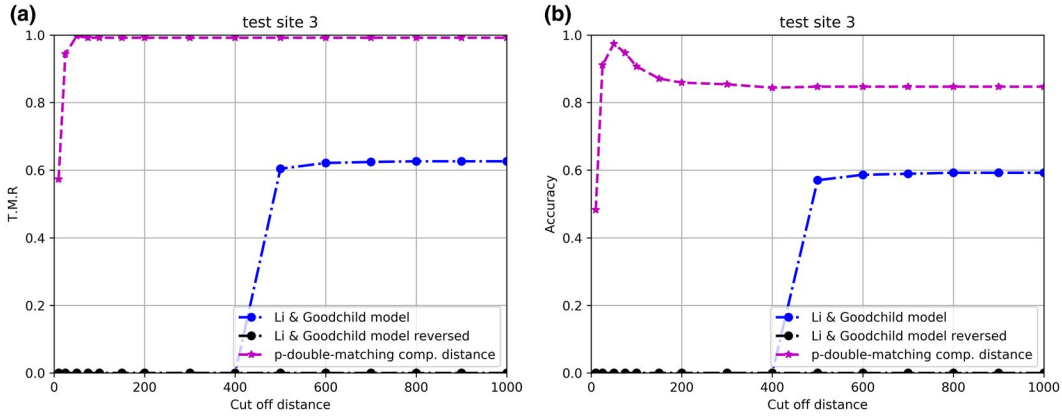

FIGURE A2 (a) True match rate; and (b) accuracy (Section 4.2) of the proposed model on test site #3

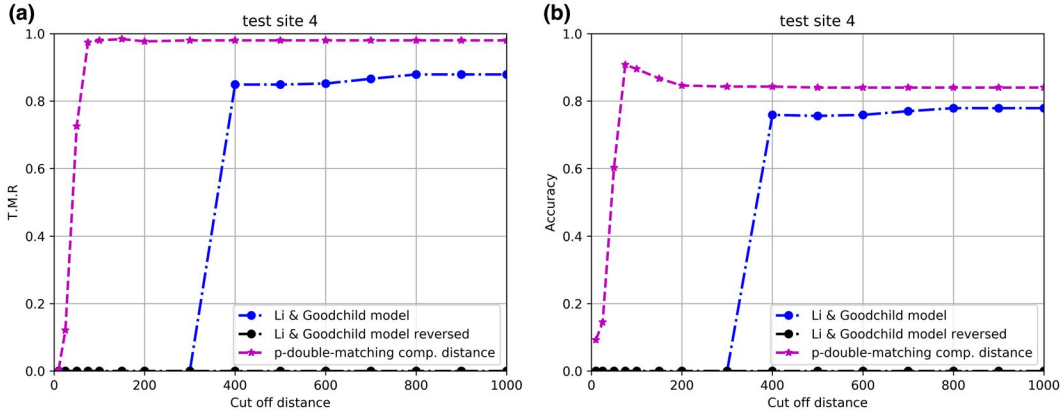

FIGURE A3 (a) True match rate; and (b) accuracy (Section 4.2) of the proposed model on test site #4

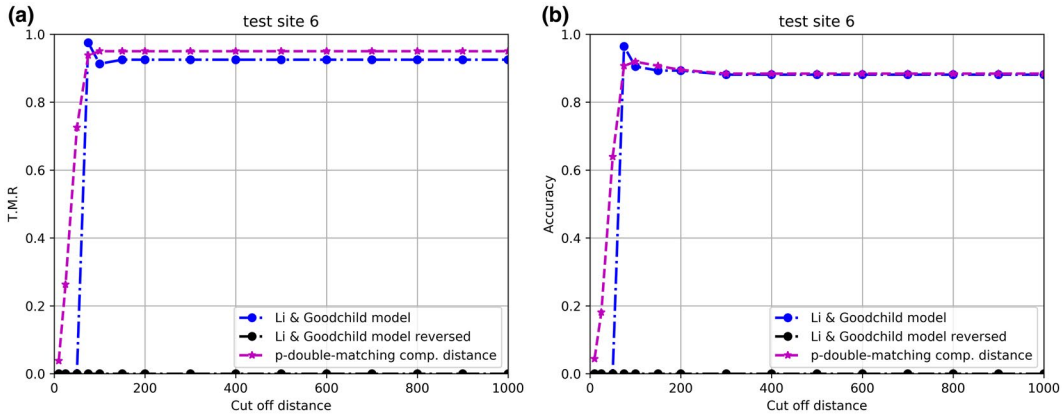

FIGURE A4 (a) True match rate; and (b) accuracy (Section 4.2) of the proposed model on test site #6