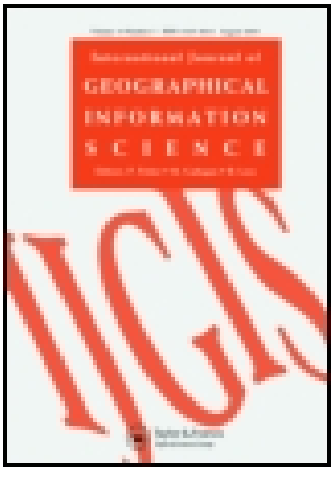

International Journal of Geographical Information Science

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tgis20

Metadata and provenance for spatial analysis: the case of spatial weights

Luc Anselin[a], Sergio J. Rey[a] & Wenwen Li[a]

[a] GeoDa Center for Geospatial Analysis and Computation School of Geographical Sciences and Urban Planning, Arizona State University, Tempe, AZ, USA

Published online: 15 May 2014.

To cite this article: Luc Anselin, Sergio J. Rey & Wenwen Li (2014): Metadata and provenance for spatial analysis: the case of spatial weights, International Journal of Geographical Information Science, DOI: 10.1080/13658816.2014.917313

To link to this article: http://dx.doi.org/10.1080/13658816.2014.917313

International Journal of Geographical Information Science, 2014
http://dx.doi.org/10.1080/13658816.2014.917313

Taylor & Francis
Taylor & Francis Group

Metadata and provenance for spatial analysis: the case of spatial weights

Luc Anselin*, Sergio J. Rey and Wenwen Li

GeoDa Center for Geospatial Analysis and Computation School of Geographical Sciences and Urban Planning, Arizona State University, Tempe, AZ, USA

(*Received 20 February 2014; accepted 18 April 2014*)

Within a CyberGIS environment, the development of effective mechanisms to encode metadata for spatial analytical methods and to track the provenance of operations is a key requirement. Spatial weights are a fundamental element in a wide range of spatial analysis methods that deal with testing for and estimating models with spatial autocorrelation. They form the link between the data structure in a GIS and the spatial analysis methods. Over time, the number of formats for spatial weights implemented in software has proliferated, without any standard or easy interoperability. In this paper, we propose a flexible format that provides a way to ensure interoperability within a cyberinfrastructure environment. We illustrate the format with an application of a spatial weights web service, which is part of an evolving spatial analytical workbench. We describe an approach to embed provenance in spatial weights structures and illustrate the performance of the web service by means of a number of small experiments.

Keywords: CyberGIS; GIScience; spatial analysis; spatial weights; metadata; provenance

1. Introduction

Cyberinfrastructure, or e-science, (as it is known in the United Kingdom) was outlined prominently in the much-cited 'Atkins' blue-ribbon report of the US National Science Foundation. It was envisaged as an encompassing computing framework for the integration of 'enabling hardware, algorithms, software, communications, institutions and personnel' (NSF 2003, p. 5). This has resulted in a renewed emphasis not only on developing high performance computing (HPC) infrastructure, but also on enabling access to distributed data and sensor information, enhancing visualization and data analysis, and facilitating the establishment of collaborative networks of scientists. Cyberinfrastructure is thus viewed as an integrated end-to-end solution to support the study of complex scientific problems in a collaborative fashion.

The geospatial sciences have embraced cyberinfrastructure (CI) as a means to leverage existing capabilities in geoprocessing and spatial analysis and tackle a new class of challenging scientific problems (Goodchild 2010). Efforts that extend cyberinfrastructure frameworks to take into account the special characteristics of geospatial data (as well as space–time data) are variously referred to as CyberGIS (Wang 2010), spatial cyberinfrastructure (Wright and Wang 2011) or geospatial cyberinfrastructure (Yang *et al.* 2010). These terms encompass not only access to HPC (e.g., grid networks or cloud computing)

*Corresponding author. Email: luc.anselin@asu.edu

but also the combination of distributed geospatial data (e.g., data repositories, sensor networks) and distributed geoprocessing (e.g., spatial data manipulation, geovisualization, pattern detection, process modeling) by means of software systems (the so-called middleware) that seamlessly integrate these resources (Wang *et al.* 2013, Li *et al.* 2014b).

Increasingly, advanced spatial analytical methods are becoming a more prominent component of CyberGIS. For example, Anselin and Rey (2012) proposed the notion of a *spatial econometrics workbench* as a framework for supporting spatial econometric research in the cyberscience era (see also the overview in Anselin 2012). In this paper, we extend this notion to encompass a wider range of techniques and refer to it as a *spatial analytical workbench*, as an environment that supports a scientific workflow, in the sense of a template of the sequence of tasks and the dependencies between them that are required to accomplish a specific goal (Deelman *et al.* 2009, Chebotko *et al.* 2011). In spatial analysis, a typical workflow consists of data pre-processing, data transformation and data integration, visualization, exploration, model specification, estimation, diagnostic testing, validation, and reporting. Within such a CyberGIS environment, the development of effective mechanisms to encode metadata for spatial analytical methods and to track the provenance of operations is a key requirement (e.g., Wang *et al.* 2008b, Yue *et al.* 2011, Anselin 2012, Wen *et al.* 2013).

We focus more specifically on the role of spatial weights in a spatial analytical CyberGIS. Spatial weights are a key component in a wide range of spatial analysis methods that deal with testing for and estimating models with spatial autocorrelation (e.g., Cliff and Ord 1973, 1981, Anselin 1988). As the formal representation of the spatial arrangement of the observations, spatial weights form the link between the data structure in a GIS and the operations associated with particular spatial analysis methods. Over time, the number of formats for spatial weights implemented in GIS and spatial analytical software has proliferated, without any standard or easy interoperability. For example, the PySAL library for spatial analysis (Rey and Anselin 2007) supports no less than 11 different file formats for spatial weights (see http://pysal.org). The existence of multiple incompatible file formats limits interoperability and forms an important impediment for the incorporation of spatial autocorrelation analysis in a CyberGIS.

The objectives of this paper are twofold. First, we propose a new lightweight and extensible means to encode spatial weights information that includes metadata and allows effective handling of provenance. Second, we incorporate the new format into a *spatial weights web service* as a component of the spatial analytical workbench and assess its performance in a realistic setting, taking advantage of HPC infrastructure. In the remainder of the paper, we first outline our vision of the role of a spatial analytical workbench in CyberGIS. This is followed by a taxonomy of spatial weights and their characteristics. Next, we outline our proposed format for encoding spatial weights information and illustrate this with an example. We also devote a section to the description of the delivery of the spatial weights operations as a web service, followed by an extensive evaluation of its performance. We close with some concluding remarks.

2. CyberGIS and the spatial analytical workbench

Our specific effort is part of an overall endeavor to integrate and sustain a core set of composeable, interoperable, manageable and reusable CyberGIS software elements based on community-driven open source strategies (Wang *et al.* 2013). In the geospatial arena, some progress has been made toward implementing an operational cyberinfrastructure, although the focus has tended to be on distributed geoprocessing (e.g., Yang *et al.* 2008,

Yang and Raskin 2009, Li *et al.* 2010, Zhao *et al.* 2012) and proof of concept applications that illustrate the use of middleware to access HPC infrastructure, such as grid or cloud computing (e.g., Yan *et al.* 2007, Pallickara and Pierce 2008, Wang *et al.* 2008a, Wilkins-Diehr *et al.* 2008, Zhang and Tsou 2009, Harris *et al.* 2010, Wang 2010, Xie *et al.* 2010, Srinivas *et al.* 2011). To date, the integration of advanced spatial data analysis and cyberinfrastructure is still in its infancy and an area of active research (Anselin and Rey 2012, Li *et al.* 2013).

Our particular approach toward developing a *spatial analytical workbench* consists of leveraging the efforts behind the PySAL open source library for spatial analysis that is written in the Python language (Rey and Anselin 2007). In addition to modules that deal with fundamental operations behind any analysis (e.g., file input-output, geocomputation, basic spatial data structures), PySAL contains specialized functionality for exploratory spatial data analysis (e.g., global and local spatial autocorrelation statistics), the creation and manipulation of spatial weights, indicators of spatial inequality, measures of spatial dynamics, point patterns analysis on networks, regionalization, and spatial regression/spatial econometrics (see pysal.org for an extensive overview). The first official release of the PySAL library occurred in August, 2010. The development team adheres to a strict 6-month release schedule, with the latest stable version (Version 1.7) made available in January 2014. During the brief period since its first release, PySAL has gained a growing adoption in the open source GIS/spatial analysis world. This culminated in its inclusion in the Anaconda distribution for Python visualization and data exploration from Continuum Analytics, a highly regarded numerical toolbox in the scientific Python community (see http://continuum.io/index).

Moving the PySAL functionality to a CyberGIS framework as a spatial analytical workbench builds upon and leverages the current development efforts, but converts them from a focus on a desktop environment to providing a collection of software components delivered over the Internet. This adheres to the concept of 'service-oriented science', consisting of distributed networks of interoperating services (Foster 2005). The services can be combined in a scientific workflow to carry out a specific set of tasks. The core idea is to deploy components of the PySAL library as specialized web services that adhere to common and open standards. The workbench then becomes a gateway that provides access to a collection of techniques, data sets, and simulation environments to support a wide range of empirical applications.

In this process, we have to address two major challenges: the development of efficient algorithms and enhancing interoperability among spatial analytical services. Since most current spatial analytical and spatial econometric software is written for computer architectures that consist of a single CPU, it does not naturally take advantage of the high-performance computing power contained in a cyberinfrastructure. Therefore, traditional spatial analytical algorithms need to be reconceptualized to exploit multiprocessing. In particular, the potential for vectorization and parallelization should be explored to improve performance in a *big data* setting. An in-depth discussion of this topic is beyond the scope of the current paper.[1] Instead, we focus on the challenge of interoperability.

In order to ensure interoperability between spatial analytical modules in a CyberGIS framework, it is necessary to focus on both the data component and the modeling (analytical) component. For each of these, the lineage of initial sources and their characteristics, manipulations and analytical operations needs to be recorded and made available to facilitate replication. Some early suggestions pertaining to GIS data provenance were reported in Lanter (1991, 1993). Since then, various information models and standards have been proposed to standardize the representation of data provenance in a

geoprocessing workflow. In 1992, the Federal Information Processing Standards (FIPS) program approved the Spatial Data Transfer Standard (SDTS), in which a lineage model is considered a core component to validate the data quality and to ensure easy transfer of distributed data across different GIS systems (Arctur *et al.* 1998). The International Organization for Standardization (ISO) also defined a lineage model in its ISO 19115:2003 (ISO 2003) and ISO 19115–2:2009 (the extension of 19115 to handle gridded and imagery data) (ISO 2009) metadata standards. This lineage model allows the recording of descriptive provenance, including the data source as well as each processing step. In the computer science community, the Open Provenance Model (OPM) was adopted originally, but in recent years the World Wide Web Consortium (W3C) has led efforts to develop a more flexible provenance ontology (PROV-O) to capture data provenance. Building blocks of this framework include Agent, Activity, and Entity. These are defined and encoded using semantic web languages to enable formal query about data trails (Lebo *et al.* 2012, Li *et al.* 2013).

Besides representation, provenance-aware applications also need to ensure provenance capture, management, and retrieval (Miles *et al.* 2007). Provenance management typically involves the use of a database or a Resource Description Framework(RDF) triple store. The retrieval of provenance data is through a SQL (Structured Query Language) or SPARQL (Semantic Web Query Language) query in a Semantic Web environment. This aspect of provenance has been investigated extensively, e.g., in Buneman *et al.* (2001), Cui *et al.* (2000), Wang *et al.* (2008b), Chebotko *et al.* (2010). In contrast, the capture of data provenance, especially the elements that are included in a geoprocessing workflow is still new to the GIScience community. In part, this is due to the complexity of spatial analysis and the requirements to generate, represent, and transfer the provenance in an automatic manner. For example, Tilmes *et al.* (2013) develop a concept model to present the content and provenance of the national climate assessment report developed by the US Global Change Research Program. This provenance model, using W3C PROV, can be classified as a post-event provenance since the provenance is captured after the assessment on climate change is completed. Yue *et al.* (2011) develop formal mechanisms for sharing existing data provenance through an extension of the Open Geospatial Consortium (OGC) EbXML Registry Information Model (EbRIM). However, provenance capture is not addressed in this framework. More recently, Di *et al.* (2013) presented a state-of-the-art solution for capturing provenance while the data is being generated. The provenance, encoded in ISO19115 lineage model, is dynamically recorded when a web service is executed by a Business Process Execution Language (BPEL) engine.

In this paper, our focus is on developing metadata and provenance tracking for both geospatial data and the analytical operations that are part of the spatial analytical workbench. We develop a provenance engine that is capable to parse and replicate a data set (in this instance, a spatial weights matrix) based on the production process encoded in the provenance metadata. Our objective is to develop a lightweight system that tracks all operations in such a way that there is no ambiguity about the nature of the spatial weights and to ensure full replicability.

3. Spatial weights

A spatial weights matrix, typically denoted by **W**, is a positive square matrix of dimension equal to the size of the data set, i.e., $n \times n$ for a data set with n observations. Its non-zero elements denote the existence of a *neighbor* relationship among spatial observations. More formally, when observations i and j are defined as neighbors, then $w_{i,j} \neq 0$, where

$w_{i,j}$ is the element in the ith row and jth column of the weights matrix. Non-neighbors correspond to spatial weights elements $w_{i,j} = 0$. By convention, self-neighbor relations are excluded, such that the diagonal elements of the weights matrix are zero, i.e., $w_{i,i} = 0$. Higher order neighbors are derived in a recursive fashion, in that any neighbor of order k is a first order neighbor to a neighbor of order $k - 1$, and not already a neighbor of a lower order. This requires that redundancies and circular paths be removed in the construction of higher order spatial weights (e.g., Blommestein and Koper 1992, Anselin and Smirnov 1996).

Whereas the formal expression of the weights is straightforward, the definition of a *neighbor* relation is not. Different criteria may be used, some based on geographic concepts, such as distance or contiguity, others have graph-theoretic origins, or derive from social network and other more general concepts. At its core, the spatial weights matrix is a representation of a graph, where the observations are nodes and the existence of a neighbor relation is expressed as a link (Anselin and Smirnov 1996).

In addition, the simple binary neighbor relation is typically not what is used in spatial analytical operations. Instead, various types of standardization or normalization can be carried out. The most common of these is row-standardization, such that:

$$w^*_{i,j} = \frac{w_{i,j}}{\Sigma_j w_{i,j}}.$$

In other words, the weights are standardized such that they sum to 1 in each row. This facilitates the computation of a *spatially lagged* variable as the weighted sum of the values in neighboring locations (Anselin 1988). Other types of standardization have been suggested in the literature as well, see, for example, Tiefelsdorf *et al.* (1999) and Kelejian and Prucha (2010). In addition, in some contexts the value of the weights is not the result of a standardization or transformation, but instead something with intrinsic meaning, such as the weights based on notions of local clusters advanced by Aldstadt and Getis (2006). For a recent review, see, e.g., Getis (2009).

In sum, there are three fundamental properties associated with any spatial weights matrix:

- the initial data source with the *location* of observations (e.g., polygon data file, point data file, data table)
- the type of neighbor relation (e.g., contiguity, distance band, network distance)
- transformation and standardization (e.g., row-standardization, higher order contiguity).

In order to ensure proper provenance tracking for spatial weights, these properties will need to be contained in weights *metadata.* To date, this has only been the case to a very limited extent.

In the early implementations of spatial analytical software (e.g., for the computation of spatial autocorrelation coefficients), the spatial weights were constructed as full (dense) matrices. In practice, however, a weights matrix is very sparse, holding only a few non-zero elements in each row. Starting with SpaceStat Version 1.80 (Anselin 1995), two sparse formats were introduced, GAL and GWT, that quickly gained acceptance and were adopted in a range of other software implementations as well, most notably in GeoDa and the R spdep package (see the discussion in Anselin *et al.* 2006, Bivand *et al.* 2008, Anselin 2012). In addition to supporting GAL and GWT, later software packages, such as ArcGIS, Stata, GeoBUGS, and LeSage and Pace Matlab-based spatial econometrics

Table 1. Spatial weights formats supported by PySAL.

Type	File extension
Sparse contiguity (SpaceStat, GeoDa, R spdep, etc.)	GAL
Sparse general weights (SpaceStat, GeoDa, R spdep, etc.)	GWT
ArcGIS text weights	TXT
ArcGIS dbf weights	DBF
ArcGIS swm weights	SWM
Matlab spatial weights (old version)	DAT
Matlab spatial weights (new version)	MAT
Lotus weights	WK1
GeoBUGS weights	TXT
Stata weights	TXT
MatrixMarket weights	MTX

library (LeSage and Pace 2009) introduced their own formats. This led to a proliferation of incompatible file formats, as illustrated in Table 1.

Most of these file formats only contain the values for the weights together with labels for the neighbor pairs. However, there is typically no way to retrace the source of the data set used to construct the weights, which type of weights was created and any transformations it may have been subject to. GAL and GWT contain a header line with limited information, but this is insufficient for proper provenance tracking. For example, a contiguity-based GAL weights file for Southern US counties starts as:

```
0 1412 south FIPSNO
54,029 1
54,009
54,009 2
54,069 54,029
```

The first line shows that the data set has 1412 observations, is derived from the south.shp shape file and used the FIPSNO variable as a key indicator. However, the uninformed user has no way of finding out which type of contiguity was used, whether the weights were standardized or not, or how, etc. Other than clever ways to name the file, such as south_q.gal which only the user understands, there is no mechanism to implement interoperability.

4. A proposed format for spatial weights metadata

To address these challenges, we propose a metadata structure for spatial weights provenance. Its implementation relies on JavaScript Object Notation (JSON) and a key:value data structure to encode the provenance information. JSON is light-weight and has much less redundancy than an XML-based metadata structure, such as employed by ISO 19115. Moreover, JSON can be matched directly to the dictionary data type in the Python programming language, or to a hash table in other languages, such as Java and C. Therefore, such a structure is easy to parse and interpret.

Figure 1 outlines our proposed metadata structure for spatial weights provenance. Since spatial weights provenance is a special case of generic data provenance, the metadata structure builds upon the typical triple template that includes <input, operation, output>. To allow a flexible encoding of weights operations, this structure is extended to

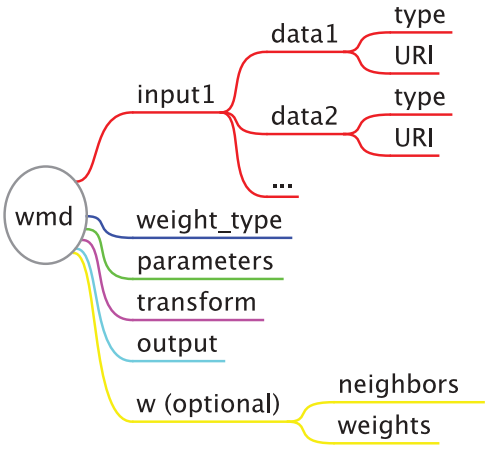

Figure 1. Weights metadata structure (wmd).

include input(s), weight_type, parameters, transform, and output. The weight_type and transform belong to operations, and they are listed separately because these two types of operations require different parameter patterns. The transformation operation, such as row standardization, derives from an existing weights object, while the value of the weight_-type is used to create a specific spatial weights object. The input(s) element identifies one or multiple data sets that are used as sources to create the weights.

Each data set has two properties: type and Universal Resource Identifier (URI). The type refers to the kind of data source used to generate spatial weights. This covers a wide range of possibilities, including a traditional ESRI shapefile, a GML file, or information contained in a geospatial database. These are direct data sources. The other data type is derived, indicating that the input data is generated from another operation instead of being provided directly. For a direct data set, the URI refers to the actual file location. For example, if a file is located on a local machine, it would have the prefix file://. On the other hand, if the file is on the Web, it would have the prefix http:// or ftp://. For a derived data set, the URI refers to another weights metadata file (i.e., a wmd file) so that the creation of the input data can be reconstructed in a recursive fashion.

The output element of the metadata structure provides the format in which the final weights matrix will be encoded. In a standalone application, this output is typically just a weights object in memory, while in a web service environment, the resulting weights object should always be written to an external drive using one of the encoding structures, e.g., gal.

Figure 2 outlines the conceptual semantic framework and describes the value space for all elements in the proposed metadata structure, as well as the relationships between them. In the figure, nodes in different colors refer to instances of the same type within the hierarchy. For example, all nodes in pink refer to operations.

The root node is wmd (Weights MetaData), which is also used as a keyword for a derived data set (leaf node in yellow). This embedded metadata structure allows for the linking of the provenance for the current operation with that of historical operations. This enables provenance backtracking and the replication of weights produced anywhere in the entire workflow.

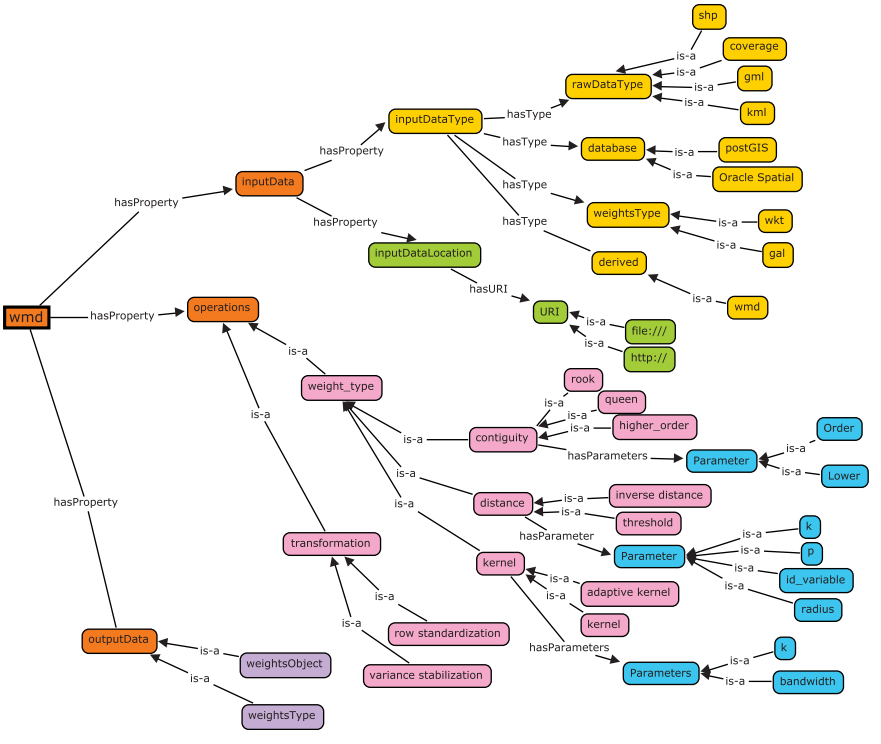

Figure 2. A conceptual model for the value space of spatial weights metadata (wmd).

A weights operation weight_type defines the construction of a specific type of spatial weights matrix, such as contiguity-based weights, distance-based weights, and kernel weights, among others. Each of these weights types has the same parameter pattern. For example, there are two contiguity-based weights operations: rook contiguity (polygons that share common borders) and queen contiguity (polygons sharing common vertices or borders). Each requires parameters to determine if a higher order of contiguity is needed (order) and if the latter includes lower order contiguities cumulatively (lower).

The metadata structure only considers the values in the leaf node. In fact, the parent nodes in this framework are abstract definitions, while the leaf nodes correspond to functions directly implemented in the PySAL spatial analysis package.

To illustrate the application of the weights metadata structure in support of the chaining of operations, we consider the construction of a specialized spatial weights matrix required for a regionalization problem (details are spelled out in Li *et al.* 2014a). Figure 3 provides the context for this problem. It consists of the aggregation of 4109 Traffic Analysis Zones (TAZ) from six Southern California counties into 100 compact regions. This regionalization is subject to two hard constraints: (1) each region must consist of a set of connected TAZ; and (2) the TAZ forming a region must all belong to the same county.

The connectivity constraint between neighboring TAZ is constructed by means of a rook operation in PySAL. The combination of the two constraints is satisfied by means of a series op set operations in PySAL. First, a second form of spatial weights is created for the TAZ, in which all zones in the same county are pairwise neighbors of each other. This

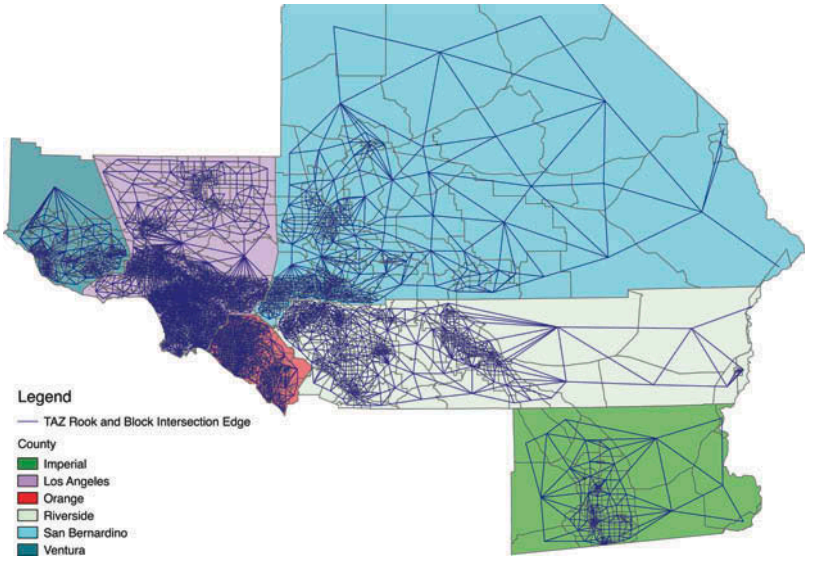

Figure 3. Traffic analysis zones, rook–block intersection contiguity graph, and county boundaries.

precludes two TAZ located in a different county to be defined as neighbors, even if they share a common boundary (as determined by the rook criterion). Such weights are termed block weights in the PySAL terminology. Finally, the required weights are obtained by applying a set operation on the contiguity and block weights to remove the edges from the rook graph that cross county boundaries, as well as edges from the block graph that do not correspond to a contiguity relationship. The graph of the resulting connectivity structure is shown in Figure 3, based on shape files for the TAZ and the counties.

Figure 4(a)–(c) illustrate how this process is implemented in three metadata files. It highlights the chaining of the operations, in which the contiguity and the block weights are constructed directly from source data, whereas the final weights are created by operating on these two initial weights.

As shown in Figure 4(a), the final wmd file has two data sources for input1, each an uri that points to another wmd file, respectively, taz_block.wmd, and taz_rook.wmd. Each of these files in turn have a specific data source specified in the data1 attribute. For the block weights, shown in Figure 4(b), the source is a dBase type file (taz.dbf) that contains the TAZ identifiers together within county field (id_variable: CNTY). Consequently, a block weights matrix can be constructed by grouping the TAZ that belong to the same county (i.e., have the same county identifier), as indicated by the weight_type block. The provenance for the contiguity weights is shown in Figure 4(c), contained in the file taz_rook.wmd. Here, the data source is a shape file (taz.shp) and the weight_type is rook.

The taz_intersection.wmd file structure illustrates how the two wmd sources are combined and subject to the intersection weights type, resulting in a gal encoding. The proposed metadata structure can thus effectively keep track of the chain of operations required to obtain the final product. However, as a workflow becomes increasingly complex, it will be difficult to carry this out as a manual operation without errors. Instead, an automatic tracking of provenance and replication of the various operations in the chain is desired.

Figure 4. Example weights meta data for intersection chaining. (a) taz_intersection.wmd. (b) taz_block.wmd. (c) taz_rook.wmd.

In the next Section, we incorporate our metadata structure into a Web Service framework. In such a framework, the spatial weights provenance is tracked automatically and any of the operations in the workflow can be replicated in a seamless manner.

5. A spatial weights web service

As a first step toward a spatial analytic workbench in a CyberGIS environment, we implemented an experimental spatial weights web service on the servers in the GeoDa Center (http://spatial.csf.asu.edu). This web service constructs the weights from provided input data and manages the provenance, using the wmd format described above.

The web service adheres to the Web Processing Service (WPS) standard from theOGC. This standard provides three interfaces to manage the input and output: GetCapabilities, DescribeProcess, and Execute.

A service is invoked by means of a standard HTTP POST request that adheres to the format for the required inputs and outputs as specified in the DescribeProcess interface. This request spells out all the parameters needed for the execution of the back end process as well as the desired type of response.

The request and response are illustrated in Figures 5 and 6. The main part of Figure 5 consists of the information contained within the <wps:Execute> tags. There are three important parts. One is the <ows:Identifier> for the weights web service, in this instance chain_ws. Next are the <wps:DataInputs> and <wps:ResponseForm> tags. The first contains the metadata information in the wmd format, in between the <wps:LiteralData> tags. The contents are identical to what is shown in Figure 4(a). The <wps:ResponseForm> tags contain the specification for the desired output. In our example, this consists of both a wmd file and a regular weights output file (out), contained between the <ows:Identifier> tags. As specified in the metadata under the 'output' key, the output should be in the gal format.

The response message is illustrated in Figure 6. The most interesting part is contained between the <wps:ProcessOutputs> tags. This consists of two <wps:Output> tags. Each of these has the full URI for the respective output (the wmd metadata file and the gal file) between <wps:LiteralData> tags.

A key aspect of the implementation of the spatial weights web service is the interface between the process request and the specific PySAL functions that actually construct the weights object. This requires an efficient parser of the information contained in the wmd format. This parser needs to interpret the weights metadata, access the relevant files and potentially be able to backtrack through the provenance chain to replicate all the necessary intermediate steps in the workflow.

The logic of this parser is illustrated in Algorithm 1. It contains the pseudo-code for a parser function (WMD_PARSER(*meta_data*)) that takes a hash table for the weights metadata as input and returns a spatial weights object (pysal.weights.W).

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wps:Execute service="WPS" version="1.0.0" xmlns:wps="http://www.opengis.net/wps/1.0.0"
        xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/wps/1.0.0/wpsExecute_request.xsd">
    <ows:Identifier>chain_ws</ows:Identifier>
    <wps:DataInputs>
        <wps:Input>
            <ows:Identifier>metadata</ows:Identifier>
              <wps:Data>
                  <wps:LiteralData>
                    {"input1":{"data1": {"type": "derived","uri":
                    "http://toae.org/pub/taz_rook.wmd"},"data2":
                    {"type":"derived","uri":"http://toae.org/pub/taz_block.wmd"}},
                    "weight_type": "intersection","transform": "O","parameters":
                    {}","output":"gal"}}
                  </wps:LiteralData>
              </wps:Data>
        </wps:Input>
    </wps:DataInputs>
    <wps:ResponseForm>
        <wps:ResponseDocument wps:lineage="true" wps:storeExecuteResponse="true"
        wps:status="false">
            <wps:Output asReference="false">
            <ows:Identifier>wmd</ows:Identifier>
            </wps:Output>
            <wps:Output asReference="false">
            <ows:Identifier>out</ows:Identifier>
            </wps:Output>
        </wps:ResponseDocument>
    </wps:ResponseForm>
</wps:Execute>
```

Figure 5. Example WPS POST request.

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:ExecuteResponse xmlns:wps="http://www.opengis.net/wps/1.0.0"
        xmlns:ows="http://www.opengis.net/ows/1.1"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
        http://schemas.opengis.net/wps/1.0.0/wpsGetCapabilities_response.xsd"
        service="WPS" version="1.0.0" xml:lang="eng"
        serviceInstance="http://spatial.csf.asu.edu/cgi-
        bin/wps.py?service=WPS&amp;request=GetCapabilities&amp;version=1.0.0"
        statusLocation="http://spatial.csf.asu.edu/wpsoutput/pywps-138924108429.xml">
    <wps:Process wps:processVersion="2.0">
        <ows:Identifier>chain_ws</ows:Identifier>
                <ows:Title>input provenance information to generate weights
        metadata, a processing chain is enabled</ows:Title>
        <ows:Abstract>input provenance information to generate weights metadata, a
        processing chain is enabled </ows:Abstract>
    </wps:Process>
    <wps:Status creationTime="Sun Feb 9 03:25:45 2014">
        <wps:ProcessSucceeded>PyWPS Process chain_ws successfully
        calculated</wps:ProcessSucceeded>
    </wps:Status>
    <wps:ProcessOutputs>
        <wps:Output>
            <ows:Identifier>wmd</ows:Identifier>
            <ows:Title>Weights metadata/provenance</ows:Title>
            <wps:Data>
               <wps:LiteralData dataType="string">
        http://spatial.csf.asu.edu/wpsoutput/892ff570-9174-11e3-ad2e-
        0050455c0671.wmd
              </wps:LiteralData>
            </wps:Data>
        </wps:Output>
        <wps:Output>
            <ows:Identifier>out</ows:Identifier>
            <ows:Title>Weights output in GAL, GWT or other weight formats as
        requested</ows:Title>
            <wps:Data>
                        <wps:LiteralData
        dataType="string">http://spatial.csf.asu.edu/wpsoutput/892ff570-9174-
        11e3-ad2e-0050455c0671.gal</wps:LiteralData>
            </wps:Data>
        </wps:Output>
    </wps:ProcessOutputs>
</wps:ExecuteResponse>
```

Figure 6. Example WPS response.

The main purpose of the parser is to track the provenance in a workflow of spatial weights operations. There are two basic use cases. In one, referred to as *derived* in Line 2, the input is a URI for another wmd file. The parser then recursively goes through the metadata hash table to create the weights files in question, as illustrated in Lines 2–6. In the second use case, the input to the weights operation is a data file as specified by a URI (Line 8). Here, we distinguish between a local and a remote file. In the former case, the input data file is already on the local computer, and nothing further needs to be done (Lines 9–10). In the latter case, the remote file needs to be downloaded to the local computer, and its URI field replaced with the location of the local file (Lines 12–13).

With all the information available, a task dispatcher invokes the required PySAL functions and returns the final weights product. This is then written out in the specified format and made available to the web service at a URI contained within the < wps: ProcessOutputs > of the WPS response (e.g., as in Figure 6).

6. Performance evaluation

In order to evaluate the performance of the spatial weights web service in a realistic environment, we conducted a number of experiments. These experiments address two important properties of the service: (1) how the processing time changes with the size of the data set to be processed; (2) how the processing time changes with an increase in the number of concurrent requests for the service.

For the second question, we are also interested in the performance profile when moving from a multi-core single machine to a compute cluster of distributed cores, i.e., a traditional HPC approach. We refer to this comparison as a *scale-up* versus a *scale-out* solution. The scale-up solution is to improve system performance by utilizing a high-end multi-core single server.[2]

The scale-out solution deploys the web service framework on the GeoDa Center web server cluster. The cluster architecture, illustrated in Figure 7, consists of a front end and a back end. For this experiment, we only employ the front end, which contains 30 dual-core web servers, interconnected to handle multiple web requests. Each server is configured with 2G Hz processor and 4G memory. The workload on these servers is managed by an Apache load balancer.[3] In addition to comparing the scale-up and the scale-out solutions directly, we are also interested in the evolution of the performance as more compute nodes are made available in the cluster to deal with concurrent requests.

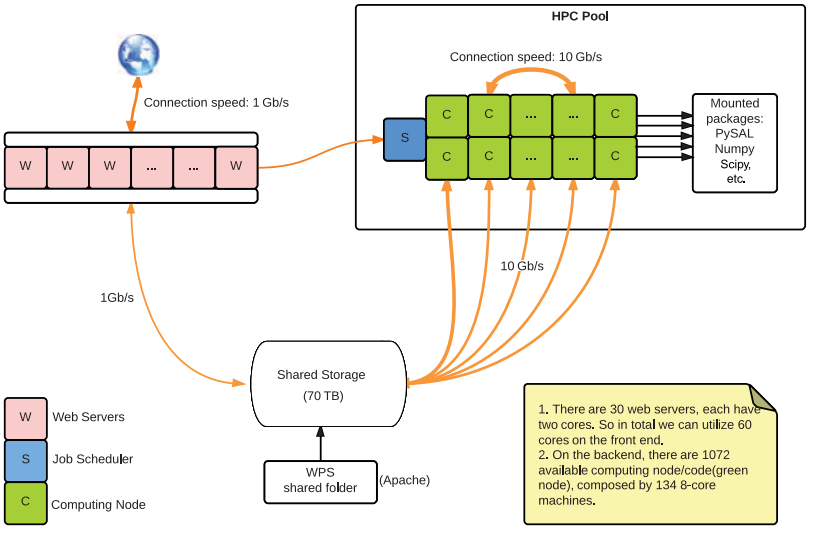

Figure 7. Web cluster architecture.

Algorithm 1: WMD_PARSEr(*wmd_object*)

```
Input: weights metadata as a hash table, examples can be found in Figure 4(a)–(c)
Output: a spatial weights object w: pysal.weights.W
for each input data i do
    if derived then
        uri ← get metadata URI
        meta_data ←load uri, get metadata object as a hash table
        w = WMD_PARSER(meta_data)
        replace i's URI with w in wmd_object
    else
        uri ← get URI of the input data file
        if local file then
            do nothing
        else
            localuri ←download remote data files, save it to a local folder, and generate a
            local URI
            replace i's URI with localuri in wmd_object
weight_type ← get the requested weights operation from metadata
w ←dispatch job to PySAL by providing all input data sets
return w
```

We use the spatial weights web service to create both rook contiguity and *k*-nearest neighbor spatial weights, with $k = 4$. The spatial layout is based on n random spatial points in the unit square, with n varying from 10,000 to 100,000. Consequently, the dimension of the spatial weights ranges in principle from 10,000 × 10,000 to 100,000 × 100,000, but because we employ sparse structures, the actual size of the files involved is much smaller. We create the rook contiguity weights by first forming Thiessen polygons around the random points and then deriving the contiguity from those polygons (a standard procedure in PySAL).

In order to mimic the performance of an actual remote web service, we store the input data sets (ESRI polygon and point shape files) on a data server in Provo, Utah. All computations were carried out on the web servers in the GeoDa Center at ASU (in Tempe, AZ).

6.1. Experiment 1 – scalability

The first experiment compares the overall time (in seconds) to create the rook contiguity and *k*-nearest neighbor weights as the size of the data set moves from $n = 10{,}000$ to $n = 100{,}000$. In Figures 8 and 9, we show the elapsed time on the vertical axis separately for the server overhead, the data transfer, the computation and the total time. Each value reported is actually an average over 5 separate runs to remove potential outliers due to unforeseen system overuse.

Overall, it is clear that the largest share of the time is taken up by the actual computations carried out in PySAL. This time increases in a near-linear fashion with the size of the data set, with the rook contiguity weights running roughly twice as fast as the *k*-nearest neighbor weights. Even with $n = 100{,}000$, the service overhead remains low, even though this includes writing out the weights object from memory to a GAL or GWT format file on the server and making it web accessible. In other words, the service performs well with low latency and small overhead. The major challenge is to fine tune

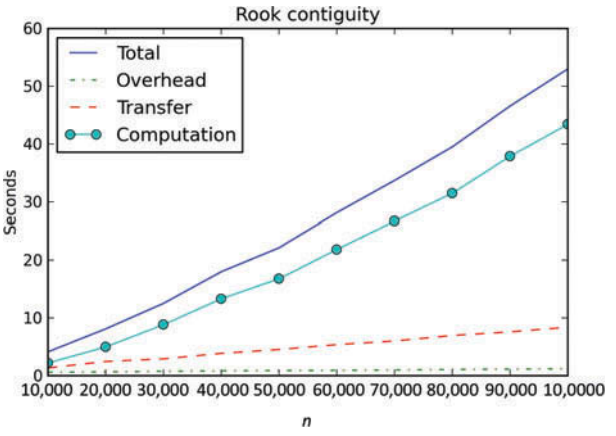

Figure 8. Spatial weights service response time – rook contiguity.

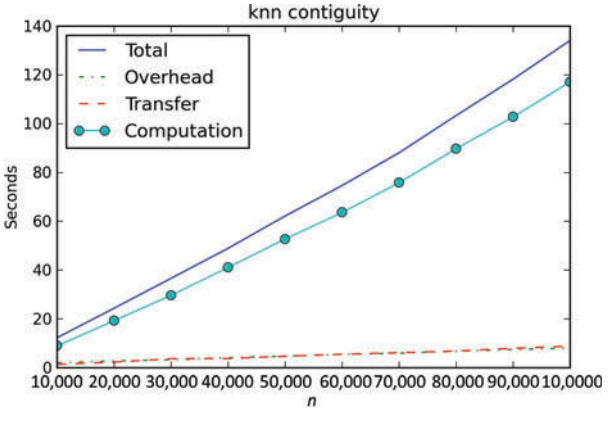

Figure 9. Spatial weights service response time – *k*-nearest neighbors contiguity.

the algorithms used in the weights construction (potentially taking advantage of parallelization), where the greatest speed gains could be obtained.

6.2. *Experiment 2 – scale-up compared to scale-out approach*

In the second experiment, we compare the performance on a desktop server with a total of eight cores (scale-up) to a cluster environment with four compute nodes enabled, also providing a total of eight cores (scale-out). We compute *k*-nearest neighbor spatial weights with $k = 4$ for $n = 10{,}000$.

In Figure 10, we show the total time elapsed (in seconds) on the vertical axis against the number of concurrent service requests, the latter increasing up to 30. The motivation for the comparison is that the two setups handle multiple requests in a different fashion. On the single server, each additional request forks a new process on an idle core to handle the request, up until the point when all cores are busy (in our example, at the point of reaching 8 requests). Any additional requests need to wait until a core is freed up. This

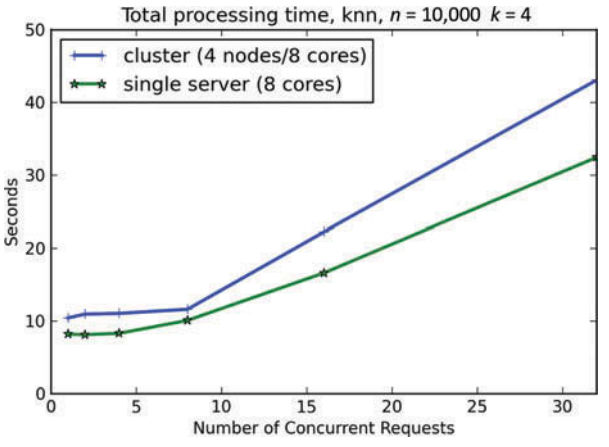

Figure 10. Processing time by concurrent service requests.

creates a dichotomy in processing time between requests that can be satisfied immediately (a compute core is available) and requests that have to wait until a core is available.

In the cluster setup, the allocation of compute time is based on a round-robin strategy implemented by the load balancer. When one computing node receives more requests than the number of CPU cores available on the node, multiple threads are created and scheduled using a round-robin strategy. As a result, all requests are processed concurrently and carried out in an average time. In contrast, the single server solution has some requests satisfied immediately, but others only with high latency.

Figure 10 shows a slight edge for the single server relative to the cluster, which in part is due to the faster processor on the single server. As long as the number of requests is less than eight, the total processing time is near constant, showing the power of the multi-processing. When the number of requests exceeds eight, the processing time increases linearly with the number of requests.

Figure 11 shows a similar picture, but now focused on the speedup curve, i.e., the comparison to the time on a single core. As long as the number of requests does not exceed 8, the speedup curve illustrates a steep increase, illustrating how the multiple processors are being put to good use. Once the number of requests exceeds eight, the speedup curve is essentially flat, showing no further gains. Interestingly, for exactly eight requests, the speedup for the cluster is slightly better than for the single server.

6.3. *Experiment 3 – scalability with number of cores*

As a final experiment, we consider the effect on processing time of using the full capability of a cluster HPC setup. We again compute k-nearest neighbor spatial weights with $k = 4$ and $n = 10{,}000$. We also keep the number of concurrent requests at 128.

Figure 12 shows the total processing time on the vertical axis and an increasing number of compute nodes on the horizontal axis, going from 4 to 64. Since each compute node contains 2 CPU, the 64 nodes match the number of requests to the number of cores (128). We show both the 'theoretical' expected time, based on the assumption that when

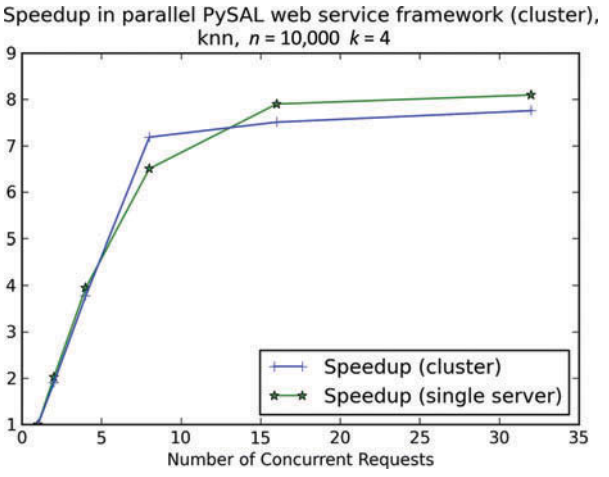

Figure 11. Speedup by concurrent service requests.

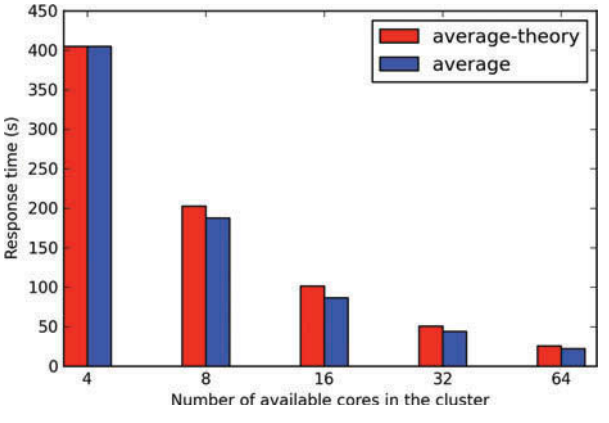

Figure 12. Processing time by number of cores in the cluster.

the available compute cores double, the response time should be reduced by half. The baseline response time is 400 seconds, for the case where four cores are employed.

The processing time goes from 400 seconds to about 20 seconds for a layout with 64 compute nodes. The actual time is slightly less than the theoretical average, suggesting that the complexity of load balancing multiple distributed compute nodes is less than the resource scheduling among CPU cores on a single server. This also highlights the high scalability and low resource management overhead of the spatial weights web services when deployed on an HPC cluster.

7. Conclusion

This article has introduced a framework for representation and capture of provenance in spatial analytical workflows. A lightweight provenance model was developed for the case of construction, manipulation, and the application of spatial weights reflecting neighbor relationships between observational units that are central to many forms of spatial

analysis. This provenance model addresses a critical need of supporting interoperability between different spatial analysis packages by reducing the uncertainty surrounding the lineage of the spatial weights with regard to any operations and manipulations that may have been applied at earlier stages in a scientific workflow.

The framework supports both local and distributed workflows, the latter involving the chaining of data sets and analytical operations that are located on geographically distinct servers. Toward this end, we also embed the provenance framework within a high-performance web service environment to ensure interoperability can be maintained in a distributed context. Timing experiments reveal that the service scales very well both in terms of the size of the problem under consideration as well as in handling simultaneous service requests.

Our spatial weights metadata prototype is an initial attempt at developing standards for provenance tracking in spatial analytical workflows. We hope to explore collaboration with other researchers in spatial analysis and cyberinfrastructure communities in refining these standards and in extending these standards to the broader set of spatial analytical services in future work.

Acknowledgments

This research was supported in part by NSF Award OCI-1047916, SI2-SSI: CyberGIS Software Integration for Sustained Geospatial Innovation.

Notes

1. For a recent example of the implementation of parallelization in PySAL, see Rey *et al.* (2013).
2. Our experimental system consists of a Mac Pro workstation with two 2.93 GHz Quad-Core Intel Xeon processors and 16 GB of 1066 MHz DDR3 ECC memory, running the Mac OS X Lion 10.7.4 operating system.
3. The back end consists of 132 Quad-core HPC nodes with over 1000 computing cores available. The front and back end of the cluster have access to 70 TB shared storage for data exchange.

References

Aldstadt, J. and Getis, A., 2006. Using AMOEBA to create a spatial weights matrix and identify spatial clusters. *Geographical Analysis*, 38, 327–343. doi:10.1111/j.1538-4632.2006.00689.x

Anselin, L., 1988. *Spatial econometrics: methods and models*. Dordrecht: Kluwer Academic Publishers.

Anselin, L., 1995. *SpaceStat version 1.80 user's guide*. Morgantown: West Virginia University.

Anselin, L., 2012. From SpaceStat to CyberGIS: twenty years of spatial data analysis software. *International Regional Science Review*, 35, 131–157. doi:10.1177/0160017612438615

Anselin, L. and Rey, S.J., 2012. Spatial econometrics in an age of CyberGIScience. *International Journal of Geographical Information Science*, 26, 2211–2226. doi:10.1080/13658816.2012.664276

Anselin, L. and Smirnov, O., 1996. Efficient algorithms for constructing proper higher order spatial lag operators. *Journal of Regional Science*, 36, 67–89. doi:10.1111/j.1467-9787.1996.tb01101.x

Anselin, L., Syabri, I., and Kho, Y., 2006. GeoDa: an introduction to spatial data analysis. *Geographical Analysis*, 38, 5–22. doi:10.1111/j.0016-7363.2005.00671.x

Arctur, D., *et al.*, 1998. Issues and prospects for the next generation of the spatial data transfer standard (SDTS). *International Journal of Geographical Information Science*, 12 (4), 403–425. doi:10.1080/136588198241851

Bivand, R.S., Pebesma, E.J., and Gómez-Rubio, V., 2008. *Applied spatial data analysis with R*. New York: Springer.

Blommestein, H.J. and Koper, N.A., 1992. Recursive algorithms for the elimination of redundant paths in spatial lag operators. *Journal of Regional Science*, 32, 91–111. doi:10.1111/j.1467-9787.1992.tb00170.x

Buneman, P., Khanna, S., and Wang-Chiew, T., 2001. Why and where a characterization of data provenance. *In*: J. Van den Bussche and V. Vianu, eds. *Database theory – ICDT 2001*, Lecture Notes in Computer Science Vol. 1973. Berlin: Springer, 316–330.

Chebotko, A., Simmhan, Y., and Missier, P., 2011. Guest editorial: scientific workflows, provenance and their application. *International Journal of Computers and Their Applications*, 18, 130–132.

Chebotko, A., *et al.*, 2010. Rdfprov: a relational RDF store for querying and managing scientific workflow provenance. *Data & Knowledge Engineering*, 69 (8), 836–865. doi:10.1016/j.datak.2010.03.005

Cliff, A. and Ord, J.K., 1973. *Spatial autocorrelation*. London: Pion.

Cliff, A. and Ord, J.K., 1981. *Spatial processes: models and applications*. London: Pion.

Cui, Y., Widom, J., and Wiener, J.L., 2000. Tracing the lineage of view data in a warehousing environment. *ACM Transactions on Database Systems* (*TODS*), 25 (2), 179–227. doi:10.1145/357775.357777

Deelman, E., *et al.*, 2009. Workflows and e-science: an overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25 (5), 528–540. doi:10.1016/j.future.2008.06.012

Di, L., Shao, Y., and Kang, L., 2013. Implementation of geospatial data provenance in a web service workflow environment with ISO 19115 and ISO 19115-2 lineage model. *IEEE Transactions on Geoscience and Remote Sensing*, 51, 5082–5089. doi:10.1109/TGRS.2013.2248740

Foster, I., 2005. Service-oriented science. *Science*, 308, 814–817. doi:10.1126/science.1110411

Getis, A., 2009. Spatial weights matrices. *Geographical Analysis*, 41, 404–410. doi:10.1111/j.1538-4632.2009.00768.x

Goodchild, M.F., 2010. Whose hand on the tiller? Revisiting 'spatial statistical analysis and GIS'. *In*: L. Anselin and S.J. Rey, eds. *Perspectives on spatial analysis*. Heidelberg: Springer-Verlag, 49–59.

Harris, R., *et al.*, 2010. Grid-enabling geographically weighted regression: a case study of participation in higher education in England. *Transactions in GIS*, 14, 43–61. doi:10.1111/j.1467-9671.2009.01181.x

ISO, 2003. *Geographic information-metadata, ISO standard 19115*. International Organization for Standardization Technical report.

ISO, 2009. *Geographic information-metadata-part 2: extensions for imagery and gridded data, ISO standard 19115-2*. International Organization for Standardization Technical report.

Kelejian, H.H. and Prucha, I.R., 2010. Specification and estimation of spatial autoregressive models with autoregressive and heteroskedastic disturbances. *Journal of Econometrics*, 157, 53–67. doi:10.1016/j.jeconom.2009.10.025

Lanter, D., 1991. Design of a lineage-based meta-database for GIS. *Cartography and Geographic Information Science*, 18, 255–261. doi:10.1559/152304091783786718

Lanter, D., 1993. A lineage meta-database approach toward spatial analytic database optimization. *Cartography and Geographic Information Science*, 20, 112–121. doi:10.1559/152304093782610315

Lebo, T., Sahoo, S., and McGuinness, D. (2012). PROV-O: the PROV ontology. W3C working draft 3 May 2012. *World Wide Web Consortium.* Available from: http://www.w3.org/TR/prov-o [Accessed 5 May 2014].

LeSage, J.P. and Pace, R.K., 2009. *Introduction to spatial econometrics*. Boca Raton, FL: CRC Press.

Li, W., Church, R.L., and Goodchild, M.F., 2014a. An extendable heuristic framework to solve the p-compact-regions problem for urban economic modeling. *Computers, Environment and Urban Systems*, 43, 1–13. doi:10.1016/j.compenvurbsys.2013.10.002

Li, W., Yang, C., and Yang, C., 2010. An active crawler for discovering geospatial web services and their distribution pattern – a case study of OGC web map service. *International Journal of Geographical Information Science*, 24 (8), 1127–1147. doi:10.1080/13658810903514172

Li, W., *et al.*, 2013. A geospatial cyberinfrastructure for urban economic analysis and spatial decision-making. *ISPRS International Journal of Geo-Information*, 2, 413–431. doi:10.3390/ijgi2020413

Li, W., *et al.*, 2014b. A service-oriented smart CyberGIS framework for data-intensive geospatial problems. *In*: S. Wang and M. Goodchild, eds. *CyberGIS: fostering a new wave of geospatial discovery and innovation*. Berlin: Springer.

Miles, S., *et al.*, 2007. The requirements of using provenance in e-science experiments. *Journal of Grid Computing*, 5 (1), 1–25. doi:10.1007/s10723-006-9055-3

NSF, 2003. *Revolutionizing science and engineering through cyberinfrastructure, report of the National Science Foundation Blue-Ribbon Advisory Panel on cyberinfrastructure*. Washington, DC: National Science Foundation.

Pallickara, S.L. and Pierce, M., 2008. SWARM: scheduling large-scale jobs over the loosely-coupled HPC clusters. *In*: *IEEE f4th international conference on eScience*. Indianapolis, IN: IEEE, 285–292.

Rey, S.J. and Anselin, L., 2007. PySAL, a python library of spatial analytical methods. *The Review of Regional Studies*, 37 (1), 5–27.

Rey, S.J., *et al.*, 2013. Parallel optimal choropleth map classification in PySAL. *International Journal of Geographical Information Science*, 27, 1023–1039. doi:10.1080/13658816.2012.752094

Srinivas, J., *et al.*, 2011. Geoportal – a spatial cloud information service. *International Journal of Engineering Science and Technology*, 3, 7930–7933.

Tiefelsdorf, M., Griffith, D.A., and Boots, B., 1999. A variance stabilizing coding scheme for spatial link matrices. *Environment and Planning A*, 31, 165–180. doi:10.1068/a310165

Tilmes, C., *et al.*, 2013. Provenance representation for the national climate assessment in the global change information system. *IEEE Transactions on Geoscience and Remote Sensing*, 51, 5160–5168. doi:10.1109/TGRS.2013.2262179

Wang, S., 2010. A CyberGIS framework for the synthesis of cyberinfrastructure, GIS, and spatial analysis. *Annals of the Association of American Geographers*, 100 (3), 535–557. doi:10.1080/00045601003791243

Wang, S., *et al.*, 2008a. Grid computing of spatial statistics: using the TerraGrid for Gi*(d) analysis. *Concurrency and Computation: Practice and Experience*, 20, 1697–1720. doi:10.1002/cpe.1294

Wang, S., *et al.*, 2008b. Towards provenance-aware geographic information systems. *In*: *Proceedings of the 16th ACM SIGSPATIAL international conference on advances in geographic information systems (ACM GIS 2008)*, 5–7 November. Irvine, CA: ACM, 70–75.

Wang, S., *et al.*, 2013. CyberGIS software: a synthetic review and integration roadmap. *International Journal of Geographical Information Science*, 27, 2122–2145. doi:10.1080/13658816.2013.776049

Wen, Y., *et al.*, 2013. Prototyping an open environment for sharing geographical analysis models on cloud computing platform. *International Journal of Digital Earth*, 6, 356–382. doi:10.1080/17538947.2012.716861

Wilkins-Diehr, N., *et al.*, 2008. Teragrid science gateways and their impact on science. *Computer*, 41 (11), 32–41. doi:10.1109/MC.2008.470

Wright, D. and Wang, S., 2011. The emergence of spatial cyberinfrastructure. *Proceedings of the National Academy of Sciences*, 108 (14), 5488–5491. doi:10.1073/pnas.1103051108

Xie, J., *et al.*, 2010. High-performance computing for the simulation of dust storms. *Computers, Environment and Urban Systems*, 34 (4), 278–290. doi:10.1016/j.compenvurbsys.2009.08.002

Yan, J., *et al.*, 2007. Parallelizing MCMC for Bayesian spatiotemporal geostatistical models. *Statistics and Computing*, 17, 323–335. doi:10.1007/s11222-007-9022-2

Yang, C. and Raskin, R., 2009. Introduction to distributed geographic information processing research. *International Journal of Geographical Information Science*, 23, 553–560. doi:10.1080/13658810902733682

Yang, C., *et al.*, 2008. Distributed geospatial information processing: sharing distributed geospatial resources to support digital earth. *International Journal of Digital Earth*, 1, 259–278. doi:10.1080/17538940802037954

Yang, C., *et al.*, 2010. Geospatial cyberinfrastructure: past, present and future. *Computers, Environment and Urban Systems*, 34 (4), 264–277. doi:10.1016/j.compenvurbsys.2010.04.001

Yue, P., *et al.*, 2011. Sharing geospatial provenance in a service-oriented environment. *Computers, Environment and Urban Systems*, 35, 333–343. doi:10.1016/j.compenvurbsys.2011.02.006

Zhang, T. and Tsou, M.-H., 2009. Developing a grid-enabled spatial web portal for internet GIServices and geospatial cyberinfrastructure. *International Journal of Geographical Information Science*, 23 (5), 605–630. doi:10.1080/13658810802698571

Zhao, P., Foerster, T., and Yue, P., 2012. The geoprocessing web. *Computers & Geosciences*, 47, 3–12. doi:10.1016/j.cageo.2012.04.021