2018

Open Source Foundations for Spatial Decision Support Systems

Jochen Albrecht
CUNY Hunter College

GI_Forum 2018, Issue 2
Page: 227 - 239
Full Paper
Corresponding Author:
jochen@hunter.cuny.edu
DOI: 10.1553/giscience2018_02_s227

Open Source Foundations for Spatial Decision Support Systems

Jochen Albrecht
City University of New York, USA

Abstract

Spatial Decision Support Systems (SDSS) were a hot topic in the 1990s, when researchers tried to imbue GIS with additional decision support features. Successful practical developments such as HAZUS or CommunityViz have since been built, based on commercial desktop software and without much heed for theory other than what underlies their process models. Others, like UrbanSim, have been completely overhauled twice but without much external scrutiny.

Both the practical and the theoretical foundations of decision support systems have developed considerably over the past 20 years. This article presents an overview of these developments and then looks at what corresponding tools have been developed by open source communities. In stark contrast to the abundance of OpenGeo software, there is a dearth of open source SDSS. The core of the article is a discussion of different approaches that lend themselves as platforms to develop an open source framework to build a variety of SDSS.

Keywords:

inference engine, knowledgebases, OLAP, operations research, utility theory

1 Introduction

Over the course of the past ten years, geospatial open source software has become mainstream (O'Sullivan *et al.* 2018). From the ubiquitous QGIS to R for spatial analysis (Lovelace et al. 2018), to web service platforms that are now taken for granted, free and open source software (FOSS) is – as a minimum – a viable alternative to commercial off-the-shelf software, and often enough this is the first place we look when we embark on a new project. One of the advantages of FOSS is its reliance on open standards, which often make it easier to mesh it with the functionality that is needed in one's project – be it in an ad hoc manner or for a larger development effort.

Arguably, one of the main uses of geospatial software is in a spatial decision support context, where we are trying to solve non-trivial, multi-criteria, and/or multi-objective problems that require a fair amount of (geospatial) data analysis. It therefore seems surprising that there are

currently only two smaller add-on packages to QGIS (InaSAFE and UMEP) that can be regarded as spatial decision support system (SDSS) in the domain of FOSS, while others like AequilibraE (2015) and Redistricting (2014) prove that it takes a larger organization's sustained support capacity to maintain a multitude of interfaces to ever-changing libraries. This article seeks to find out why this glaring gap exists and what it would take to develop an open source SDSS platform. This will be accomplished by first surveying the scene of widely-used SDSS and creating a list of desiderata that an open source platform would have to fulfil to successfully compete with existing commercial offerings. Second, as 'decision support' is an obvious necessity, we will look at the state of the art in operations research to see what could be learned from this neighbouring discipline. A discussion of development platforms will get us closer to answering the original question of why we still lack a standard go-to package such as we seem to have in almost all other realms of geospatial FOSS. The article ends with a list of steps to be undertaken for development of an open source SDSS platform.

2 SDSS: State of the Art in the 2010s

Other than for a plethora of academic research projects that never survived their initial funding timelines, SDSS is either implemented as an extension to the market-leading GIS software from ESRI, or has been developed for in-house applications only (Gheorghe 2005). In this section, we will briefly describe some of the most popular SDSS, with an emphasis on the qualities that we would like to see replicated in the open source environment.

HAZUS

HAZUS, sho.rt for Hazards US, began in the 1990s as a natural hazards loss-estimation tool; in 2004 it was unified into a multi-hazard analysis software suite for the US Federal Emergency Management Agency (Schneider & Schauer, 2006). Implemented as an (unusually large) extension package to ArcGIS Desktop, it is used worldwide as well as in thousands of US local, regional and national agencies. Two aspects that contribute significantly to the widespread use of HAZUS are: (a) its being packaged with several GBytes of detailed socio-economic data that make it easy to get started with the program, data which is useful beyond the immediate application to hazard management; (b) a plethora of courses that provide a range of specialized introductions to the software.

The HAZUS developers have slowly been trying to move away from their dependencies on ESRI software (personal communication, 2018). First, the traditional ESRI geodatabase turned out to be inefficient for larger database and scenario requirements and the developers adopted an SQL Server. Recent versions moved the source code to C#, and as of summer 2018 the development team has been looking into moving all analysis routines into open source (OpenHazus 2018). As the HAZUS community as a whole is very familiar with the ESRI platform, this change is gradually being effected. The user interface, complicated as it is, cannot be radically changed without upsetting a larger user base.

The hallmark feature of HAZUS is its use as a scenario generator. Emergency management agencies use HAZUS to construct large libraries of scenario runs in order to build repositories of loss measures for as large a combination of variables and parameters as possible. As scenario runs are too computing-intensive to be conducted in real time, it is access to the scenarios and their use in preparatory exercises that allows first responders to make informed decisions. The highly unlikely scenario of Superstorm Sandy in 2015, for instance, was part of the fully calculated repository of New York City's Office of Emergency Management (OEM). While the scenario itself was deemed too unlikely to commit resources towards its prevention, the well-known consequences allowed the storm's effects to be predicted quite accurately and the prevention of major loss of life (Friedman 2017).

Developed for this specific purpose, HAZUS is not meant to be used by people unfamiliar with emergency management procedures, but by higher-level, technically well-versed managers. There is no attempt to make it accessible to first responders directly, for instance. Instead, the emphasis is on resource management for mitigation. The unwieldiness of the user interface is almost a feature because the software is aimed to be used before an emergency event and in a server environment. Public outreach and communication with other software packages, prominent in other SDSS, are not part of the HAZUS design.

CommunityViz

The design intentions of CommunityViz could hardly be more contrary to those of HAZUS. Financed by the Orton Family Foundation, its purpose was and is to facilitate public participation in local planning procedures. The emphasis is on keeping citizens engaged through visualizations and easy-to-follow or interactive scenario builders. Like HAZUS, CommunityViz is an extension of ArcGIS Desktop, but its owner, City Explained, Inc., is continuing its development to a web services environment and creating an API that allows to link to external modelling tools.

CommunityViz projects tend to be smaller than regional-scale HAZUS projects, and the building of scenario libraries emphasizes interactivity for the purposes of garnering support and understanding rather than for subsequent statistical analysis. CommunityViz has two separate and complementary features which set it apart: (a) it has an enviable easy-to-use user interface and (b) it has the additional option of looking under the hood to see the inner workings.

UrbanSim

UrbanSim does not often feature in the lists of common SDSS, although it fulfils all criteria. It is one of the very few SDSS that transitioned successfully from the academic into the commercial realm, and the only one to my knowledge that gave up on its open source origins (although the sources of older versions remain available on https://github.com/UDST/urbansim). UrbanSim has gone through several major rewrites from Java to Python to Jupyter, and the latest commercial version is Cloud-based.

Aimed at metropolitan planning associations, UrbanSim was originally used to model urban land use and transportation demand but now emphasizes real estate markets. Technically, it

comes under the umbrella of microsimulation software, and the model equations are econometric in nature. It is hard to judge the efficacy of its features as virtually all publications (UDST 2018) are project-based and involve a high degree of customization by the developers. Successful adaptations of the source code are rare and purely academic in nature.

There are, nevertheless, a few important lessons to be learned from the UrbanSim project. The model base is quite sophisticated and easier to manipulate (for the initiated) than similar features in HAZUS or CommunityViz. Second, at the time of writing (2018), its software architecture is state of the art, and the thorough embracing of the Python universe opens it up to a large community of potential developers. This comes at the price of the software requiring a technical intermediary; to an even greater degree than HAZUS, it cannot usually be handled directly by the end-user. But with a Pythonista on staff, there are few limits to the customization of UrbanSim. This potential is documented on the GitHub pages of the Urban Data Science Toolkit (UDST 2018), with examples for 3D visualizations, US Census data ETL tools, Open Street Map and GTFS readers.

VisionMaker NYC

Another modern SDSS, with a predominantly pedagogical purpose, is under development by the Wildlife Conservation Society (WCS 2018). VisionMaker NYC is a browser-based scenario-builder that interfaces with an ArcGIS Server for pupils to build SimCity-like models (Terzano & Morckel 2016) based on real-world data. In addition to the environmental advocacy effect, VisionMaker NYC also serves its developers as an ever-growing compilation of user models that can be mined for educational, political and scientific insights. In its current implementation, the project is constrained to where there is an abundance of data; this is not a problem in New York City, but not as easy to accomplish in conservation projects around the world, where the tool would be really useful – for example, to brainstorm with local constituencies about conflicting land uses. Like CommunityViz, the user interface is exemplary in its simplicity. Except for the ArcGIS Server, all components are based on FOSS, but the VisionMaker NYC project itself is not in the public domain.

Based on these four examples, we can now draw up a list of desiderata for a FOSS SDSS:

1. Handling large amounts of data, including scenario setups and results	☞ Specialized database support, incl. ETL, OTP and OLAP
2. A GUI that draws users in	☞ High degree of interactivity, dynamic graphics, dashboards and platform transparency
3. Models and algorithms that go way beyond what is offered by GIS	☞ A model description language that allows for semantic imports of models from many disciplines and application areas[1]
4. Leveraging of operations research (OR) tools	☞ While the models are domain-specific, the decision-making procedures should follow current state-of-the-art tools developed in OR

3 Operations Research Tools

One potential reason why SDSS have not yet found a firm footing in the FOSS environment is that this would require bridging two rather separate communities – the geospatial one and that of operations research (OR). OR has been dealing with geospatial applications for many years, but it has not embraced them beyond a simple tool level. Instead, regional science, graph theoretical or econometric algorithms have been incorporated, based on what could be found in textbooks. There is no joint forum comparable to ones that spatial statisticians and environmental GIS researchers have established. This mutual ignorance is also reflected in a lack of SDSS literature, after a flurry of publications in the 1990s (Carver, 1991; Densham, 1991; Diamond & Wright, 1988; Goodchild et al., 1992; Jankowski, 1995; Malczewski, 1999). A brief introduction to OR is therefore in order.

There are two distinct parts to the analysis that underlies SDSS. One is a reasoning chain of processing steps that can be abstracted into a Markov process. The other concerns choices. Most GIS-based SDSS are about the creation of scenarios, each one being the result of a Markov process. The component that is not captured by GIS (or only barely in the form of a weighted linear combination) is how to weigh alternative solutions against each other. Readers are likely to be familiar with some of the theoretical foundations from their statistics classes: Pascal's expected value theorem (1670), Wald's general decision problem (1939), or von Neumann and Morgenstern's expected utility theory (1947).

In general, we distinguish between multi-criteria and multi-objective methods. On the multi-criteria side, the one we are all familiar with from the GIS world is weighted linear combination (Eastman 1999, Malczewski, 2014). It is noteworthy, however, that although the SDSS literature is replete with implementations of Saaty's Analytical Hierarchy Process and its generalization as an analytical network process (Saaty, 1996), none of these have been implemented as part of any common GIS package. It is also odd that beyond the Idrisi

[1] Classic SDSS application areas include location-allocation, site selection, land use suitability evaluation, transportation problems, environmental impact assessment, districting, fire and other hazards.

package, none of the freely available add-ons to ArcGIS or QGIS have ever been incorporated into the core software either.

More squarely in the realm of traditional OR tools are the 'ideal point' and the 'outranking' methods. Ideal point methods, also known as compromise programming, evaluate alternatives by measuring the distance from the ideal solution in a multi-criteria solution space. A power parameter is applied to balance criteria, to weight them according to their deviation, or to emphasize the greatest deviations. The logic is sometimes turned on its head by measuring the distance to the worst possible solution. Hwang & Yoon's (1981) technique for ordering performance by similarity to the ideal solution combines the two approaches and is now a standard tool in OR.

Outranking methods are more commonly used in French-speaking areas (or at least in Europe, where francophone OR schools play a dominant role) and can be applied to rank-ordered phenomena, tilting them more towards qualitative approaches (Roy, 1968). An alternative outranks another one if it is better for at least half the criteria. Rather than just summarizing across all ranks (referred to as concordance), the outranking method then also calls for a discordance measure, which measures the discomfort experienced by moving from one level to the next level down for each criterion. Concordance and discordance are then expressed in the form of a matrix that helps to determine which pair of alternatives is the most satisficing one.

On the multi-objective side, the goal is to find a utility-maximizing function across a set of criteria, alternatives and decision variables. Given that criteria are often conflicting, the classic solution has been described by Pareto (1896) as finding the one set in a multi-objective space in which no vector improves some objective without deteriorating another one. It turns out that such a Pareto front optimization tends either to be dominated by extreme values of one objective (typically not a feasible solution in practice), or to result in there being no single best compromise solution. OR has therefore developed a set of weighting and constraint solvers that maximize only one objective while converting the others into inequality constraints. Another set of tools parallels the distance metrics, encountered in the ideal point method for multi-criteria problems, which minimizes the distance between the desired and the achieved solution. Typically, such algorithm collections include goal programming (Charnes & Cooper, 1961), compromise programming (Zeleny, 1982), and the reference point method (Wierzbicki, 1998, which is favoured in the world of finance. Goal programming is particularly popular because it employs the *Lp* norm, which is widely used in physics, statistics, finance and engineering.

All of the above methods are still a bit idealistic, in the sense that they require more knowledge or certainty in the choice of parameter values than there is in real-world situations. Proper decision support systems, therefore, do not provide just libraries implementing the aforementioned algorithms (referred to as 'solvers' in the OR world), but also room for interactive programming similar to good data mining software. This gets us into the realm of heuristics, some of which the geocomputational community is familiar with (Vanegas et al. 2011). Church et al.'s (2003) region growing heuristic or the so-called 'greedy algorithms' that we use in the traveling-salesman problem are good examples. Other approaches employ a Lagrangian Relaxation (Fischer, 1981), where a group of constraints

that are perceived as troublesome are placed into an objective function and assigned weights (the Lagrangian multiplier).

OR Solvers

All of the above OR methods are not simply confined to descriptions in academic literature: they have been implemented in a number of software tools, in both commercial and open source forms. The first commercial integer programming package became available in the early 1970s (Forest et al., 1974), but only in the late 1990s did robust dual simplex methods leave the academic realm (CPLEX, 1999). Gurobi Optimization, named after its founders (Gu, Rothberg and Bixby), was founded in 2009 and revolutionized the mixed-integer programming world by focusing on solving a public library of over 2,000 real-world problems that was compiled by the Zuse Institute and resulted in an open source C library called SCIP (Gleixner et al., 2018; Gurobi Optimization, 2018). Similar to ESRI's dominance in the commercial GIS world, Gurobi is the standard against which all other OR tools, commercial or otherwise, are measured. Also as for ESRI, a generous academic licensing scheme helped to spread the software fast and wide.

SCIP, the more generic GNU Linear Programming Toolkit, and the computational infrastructure for operations research (COIN-OR) are examples of algorithms that are typically called from C(#) libraries. Some Java libraries also exist, with Python running a distant third. This is something to be kept in mind by developers of a FOSS SDSS, because it constrains the platform options if the 'D' in SDSS is to be taken seriously.

For an extreme example of how far the development of a fully-fledged (open source) decision support system can be taken, readers are invited to have a look at DAKOTA. Under development at Sandia National Laboratory, this optimization software project enables design exploration, model calibration, risk analysis, and quantification of margins and uncertainty via computational models (Sandia National Laboratory 2018). Designed for engineering applications, its iterative system analysis methods, all implemented in C++, are flexible enough to be extended to geospatial simulations.

4 Mining Analysis Results

As we have seen from the SDSS examples in Section 2, in addition to significantly-sized input data, a future-proof SDSS needs to be able to store and analyse the results of hundreds if not thousands of analysis runs. Currently, this requirement is the Achilles heel of existing packages. Except for the non-spatial DAKOTA, the design and organization of model runs is left to the user, as is the analysis of model results, meaning that the user must rely on external tools. This section therefore discusses the database needs and possible tools for the exploration of simulation results.

Database needs

The geospatial data community's default database management systems (DBMS) rely on relational approaches. For smaller projects, these might be SQL Server Express or SpatiaLite

solutions; for larger projects, the question is usually determined by whatever vendor the project sponsor is aligned with. In the open source world, especially if geospatial data is involved, there seems to be only one solution: Postgres/PostGIS. These relational databases are good for traditional ETL tasks but rather limiting given the many different types of data we want to store, and the kind of analyses that need to be run on simulation results. At some point, traditional relational DBMS will not suffice when it comes to storing procedures (all the parameters that describe models and their respective runs) or providing rapid access to model results. Postgres is also a less than ideal solution for cloud-based access to large spatial databases. OLAP databases address the rapid access to analysis results question, allowing the interactive analysis of multidimensional data from multiple perspectives. OLAP hypercubes are dissections of the multidimensional database, similar to how an octree tessellates a 3D space. Each cube contains aggregated data related to elements along each of its dimensions. A typical user interface to interact with the data cubes is a Pivot table. The combination of all possible aggregations plus the base data contains the answers to every query that can be answered from the data. The universe of all possible aggregations is, of course, huge. Consequently, only the more likely ones get pre-calculated to guarantee a very fast response to queries.

An example of an open source OLAP database is the Druid (2018) toolkit for event data. Closer to SDSS applications is the column-oriented Pinot datastore developed by LinkedIn (Pinot, 2018). The data is stored in an HDFS (Hadoop distributed file system), which allows it to be linked with SpatialHadoop (2018). Hadoop's Pig Latin can be extended via user-defined functions, which users can write in Java, Python, JavaScript, Ruby or Groovy, and in the case of SpatialHadoop mimic PostGIS's functionality. An alternative open source OLAP implementation is Apache Kylin (Kylin, 2018), which is built on top of Hadoop/Spark and can hence be linked with GeoMesa (2018) for very fast geospatial analytical web services. Kylin's ODBC and JDBC drivers support tools such as Tableau for immediate graphical inspection (if the FOSS SDSS development team wants to avoid dealing with the specifics of the user interface). The European OpenCube Toolkit (2018) is a fine example of how all this can be put together to process RDF data cubes. All user interactions are browser-based, including the Pivot table view and all OLAP operations. The OpenCube Toolkit executes R scripts, has interactive visualization widgets, and even a map interface in support of OLAP operations on the geo-spatial dimension. Finally, without OLAP support but adhering to a number of OGC standards, we have GeoWave (2018), which links sorted big data key-value datastores to GeoTools-compatible data sources such as GeoServer (2018). Rob Emanuele (2016) provides an excellent comparison between the two vector-based geospatial big data frameworks GeoMesa and GeoWave. As larger SDSS implementations will invariably make use of distributed data stores and web services, a future-proof open source SDSS will have to incorporate either of these two or the raster-based GeoTrellis (2018) framework.

An AI data mining toolbox

As the OpenCube Toolkit shows, once the data is stored, the real task is to make sense of all the simulation results. Until recently, SDSS have had no urgent need to provide support for that, as the simulations were handpicked by domain experts and manageable in number. Now, however, long-term users of HAZUS are struggling to keep track of all the scenarios

they have run. Also, with increasing computing power and Cloud processing, we will soon be in a position to run really large numbers of scenarios that need not only to be stored but then also mined. Our solutions space has now become large enough to warrant genetic algorithms, simulated annealing, tabu search and swarm intelligence, in addition to traditional machine learning tools like neural networks, scikit-learn (2018) or Weka (2018).

These three traditional tools are all used to perform multi-directional and global searches to explore the set of Pareto solutions. This is usually done with two or more objective functions and a number of constraints (Li et al., 2018; Liu et al., forthcoming). Classic OR algorithms may require more knowledge than we actually have, in which case these AI tools provide for something like an 'automated heuristic'.

In geospatial circles, swarm intelligence is better known as an agent-based modelling system (ABMS). ABMS are quintessential simulation tools and, as exemplified by our discussion of UrbanSim, lend themselves eminently to be used in or even as SDSS. ABMS deserve a closer look in our investigation of potential SDSS platforms because by definition they fulfil almost all the criteria listed at the end of Section 2. A system like the open source Repast Simphony (2018) includes a database, a model base, model creation (specification) tools, a scheduler, comprehensive metadata management, full import and export support for raster and vector data, and lots of links to external tools like JoSQL, Jung, Pajek, R and Weka. It could be argued that with a tool like this, we would 'only' have to add some OR tools to arrive at a fully-fledged SDSS.

5 Choosing a Platform

We are now in a position to compare five different potential platforms for building a FOSS SDSS from scratch.

GIS-based SDSS

If we were to follow the most successful commercial examples, then the choice is clear: most SDSS are basically extensions of existing GIS, and for a good reason – the learning curve for users who are already well-versed in GIS should be minimal. It therefore seems to make a lot of sense to add SDSS to the list of OSgeo projects and work with some combination of QGIS, SAGA, gvSIG or GRASS. On the other hand, adding fully-fledged database support (preferably of the OLAP kind), scenario builders, model specification tools and OR algorithms is bound to shatter user experience, and it would arguably be more streamlined to build a FOSS SDSS from low-level OSgeo libraries rather than on top of an existing desktop GIS.

Database-based SDSS

DBMS have a lot going for them. They have built-in multi-user management and the versioning mechanism would be useful in scenario management. DBMS are extremely robust, as exemplified by well-established ETL tools. They are also ideal for storing process models encoded in something like the business process execution language (WSBPEL,

2017). OR and AI tools can be implemented as extensions, just as we are used to from the spatial extensions that turn Postgres into a GIS. Compared to all these advantages, the list of drawbacks is fairly short: none of the common FOSS DBMS support OLAP, and while SQL can be extended, to do that in support of OR and AI tools would be so unorthodox that a DBMS-based SDSS would be highly unique (i.e., not standards-based). This, in turn, would violate the robustness requirement that standard DBMS fulfil so well. The discussion of the OpenCube Toolkit (Section 'Database needs') shows that it is possible to develop a DBMS-based SDSS – but this example, as convincing as it is for the purpose it was built for, is so specialized that it hardly serves as the basis for a general SDSS.

R-based SDSS

Readers of this article might be surprised that R is considered a general enough platform to allow an SDSS to be built on top of it. But R has a huge developer community, including in all areas singled out in our list of desiderata for an SDSS. There are very active communities for geo-spatial, OR and machine learning tools. Packages like rattle() illustrate how convenient it is to develop a workflow-based user interface. The main drawback to conceiving of R as an SDSS platform is the underdeveloped link to databases. There are plenty of packages that aim to provide links to DBMS, but they are slow and a de facto bottleneck. Calling R routines from a DBMS would be far more convincing.

Eclipse-based SDSS

Eclipse is a comprehensive and comfortable development environment for computer programming. Originally designed for developing Java applications, it now supports more than 25 of the most common programming languages. The agent-based modelling system Repast Simphony, as well as earlier versions or UrbanSim, are examples of Eclipse-based applications. Eclipse is extremely versatile, and its multi-language support and excellent library management make it relatively easy to cobble together all the necessary components of an SDSS. Its advantages, however, carry in themselves the seeds of the drawbacks. It is an environment for programmers – not decision makers. Hiding the platform basics from the end-user is a challenge. The multi-language support, if made use of, poses higher demands on project management. An example of that is the need to identify a development framework, which all the previously-suggested platforms have already solved.

Jupyter-based SDSS

Depending on which industry survey one follows, Python has become the most popular programming language. The advent of Jupyter notebooks likely contributed to this popularity significantly. Jupyter has been expanded and now provides support for R and Julia kernels, with dozens of additional language kernels (including several C dialects) maintained by the user community. Like Eclipse, this multi-language capability opens the doors to just about every geospatial, OR, AI, visualization, GUI-building, etc. library that one can think of. UrbanSim's move from Eclipse to Jupyter gives an indication of where things are heading. Jupyter's integration with web services and its easy user interface suggest that more interesting applications like muck (King, 2018) are just around the corner.

In summary, the choice of development platform is mostly dependent on the background of the people who are getting together to develop a FOSS SDSS. Each of the options discussed here seem feasible; as a matter of fact, non-representative surveys after the author's presentations found a just about equal distribution among the supporters of each platform. The purpose of this article is to provide an overview of the issues involved. The hope is that it provides sufficient background for an informed discussion by the GI_Forum community about open source foundations for spatial decision support systems.

References

AequilibraE (2015). Documentation and repository available online at https://github.com/AequilibraE/AequilibraE, last accessed 22 Sep 2018.

Carver, S. (1991). Integrating multi-criteria evaluation with geographical information systems, *International Journal of Geographical Information Systems*, **5**(3), 321-339.

Chares, A. and W. Cooper (1961). Management models and industrial applications of linear programming. New York: Wiley.

Church, R., Gerrard, R., Gilpin, M. and P. Stine (2003). Constructing cell-based habitat patches useful in conservation planning. *Annals of the Association of American Geographers*, **93**(4), 814–827.

City Explained (2018). *CommunityViz*. http://www.city-explained.com, last accessed 22 Aug 2018.

COIN-OR (2018). Computational Infrastructure for Operations Research Foundation. https://www.coin-or.org/, last accessed 22 Aug 2018.

CPLEX Optimization (1999). *Using the CPLEX Callable Library*. ILOG Inc.

Densham, P. (1991). Spatial decision support systems, In: Maguire, D., Goodchild, M. and D. Rhind, (Eds.), *Geographical Information Systems: Principles and Applications*, Vol.1, Harlow, UK: Longman, pp. 403-412.

Diamond, J. and J. Wright (1988). Design of an integrated spatial information system for multiobjective land-use planning, *Environment and Planning B*, **15**(2), 205-214, 1988.

Druid (2018). https://github.com/druid-io/druid, last accessed 22 Aug 2018.

Eastman, R. (1999). Multi-criteria evaluation and GIS. Chap. 35. In: Longley, P, Goodchild, M, Maguire, D and D Rhind (Eds) *Geographical information systems*. New York: Wiley, pp. 493-502.

Emanuele, R. (2016). GeoMesa and GeoWave Comparative Analysis: Final Report. Online document available at https://github.com/azavea/geowave-geomesa-comparative-nalysis/blob/master/docs/report.md, last accessed 22 Sep 2018.

Fischer, M. (1981). The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, **27**(1), 1-18.

Forest, J., Hirst, J. and J. Tomlin (1974). Practical solution of large mixed integer programming problems with UMPIRE. *Management Science*, **20**: 736-773.

Friedman, J. (2017). *GIS at the NYC Office of Emergency Management*. Presentation to the Geography Department, Hunter College, City University of New York.

GeoMesa (2018). Documentation and repository available online at https://www.geomesa.org/, last accessed 22 Sep 2018.

GeoServer (2018). Documentation and repository available online at http://geoserver.org/, last accessed 22 Sep 2018.

GeoTrellis (2018). Documentation and repository available online at https://github.com/locationtech/geotrellis, last accessed 22 Sep 2018.

GeoWave (2018). Documentation and repository available online at https://github.com/locationtech/geowave/, last accessed 22 Sep 2018.

Gheorghe, A. (Ed) (2005). *Integrated Risk and Vulnerability Management Assisted by Decision Support Systems*. Dordrecht, NL: Springer.

Gleixner, A., Bastubbe, M., Eifler, L., Gally, T., Gamrath, G., Gottwald, R., Hendel, G., Hojny, C., Koch, T., Lübbecke, M., Maher, S., Miltenberger, M., Müller, B., Pfetsch, M., Puchert, C., Rehfeldt, D., Schlösser, F., Schubert, C., Serrano, F., Shinano, Y., Viernickel, J., Walter, M., Wegscheider, F., Witt, J. and J. Witzig (2018). *The SCIP Optimization Suite 6.0*. ZIB report 18-26. Berlin: Zuse Institute.

GLPK (2018). GNU Linear Programming Toolkit. https://www.gnu.org/software/glpk/, last accessed 22 Aug 2018.

Goodchild, M., Haining, R. and S. Wise (1992). Integrating GIS and spatial data analysis: problems and possibilities, *International Journal of Geographical Information Systems*, **6**(5), 407-423.

Gurobi Optimization (2018). *Gurobi Optimizer*, version 8.0. http://www.gurobi.com/, last accessed 22 Aug 2018.

HAZUS (2018). https://www.fema.gov/hazus, last accessed 22 Aug 2018.

Hwang, C. and K. Yoon (1981). Multiple Attribute Decision making: Methods and Applications. Berlin: Springer.

InaSAFE (2018). QGIS plugin for estimating impact from natural disasters. Documentation and repository available online at http://inasafe.org/, last accessed 22 Sep 2018.

Jankowski, P. (1995). Integrating geographical information systems and multiple criteria decision making methods, *International Journal of Geographical Information Systems*, **9**(3), 251-273.

King, G. (2018). Muck, a build tool for data projects. https://github.com/gwk/muck, last accessed 2018.

Kylin (2018). http://kylin.apache.org/, last accessed 22 Aug 2018.

Li, Y., Kou, Y. and Z. Li (2018). An Improved Nondominated Sorting Genetic Algorithm III Method for Solving Multiobjective Weapon-Target Assignment. *International Journal of Aerospace Engineering*, 2018: 8302324, DOI:10.1155/2018/8302324.

Liu, Z., Wang, Y. and P. Huang (forthcoming). A Many-Objective Evolutionary Algorithm with Angle-Based Selection and Shift-Based Density Estimation. *Information Sciences* DOI: 10.1016/j.ins.2018.06.063.

Lovelace, R., Nowosad, J. and J. Muenchow (2018). *Geocomputation with R*. London: CRC/Chapman & Hall.

Malczewski, J. (1999). *GIS and Multicriteria Decision Analysis*, New York, NY: John Wiley and Sons.

O'Sullivan, C., Wise, N. and P. Mathieu (2018). The Changing Landscape of Geospatial Information Markets in Mathieu, P. and C. Aubrecht (Eds), *Earth Observation Open Science and Innovation*, pp. 3-23. Cham, CH: Springer-Nature. DOI: 10.1007/978-3-319-65633-5.

OpenCube Toolkit (2018). http://opencube-toolkit.eu/, last accessed 22 Aug 2018.

OpenHazus (2018). *OpenHazus – FEMA's Loss Estimation Model*. Program goals for 2018-2022 in alignment with FEMA strategic plan. PowerPoint presentation shared with the author by the OpenHazus development team.

Pascal, B. (1670). Pensées. Christian Classics Ethereal Library. http://www.ccel.org/ccel/pascal/pensees.html, last accessed 22 Aug 2018.

Pinot (2018). https://github.com/linkedin/pinot, last accessed 22 Aug 2018.

Rattle (2018). Rattle: a graphical user interface for data mining using R. Togaware. https://rattle.togaware.com/, last accessed 22 Aug 2018.

Redistricting (2014). Documentation and repository available online at https://github.com/seanclaude/Redistricting, last accessed 22 Sep 2018.

Repast Simphony (2018). https://repast.github.io/, last accessed 22 Aug 2018.

Roy, B. (1968). Classement et choix en présence de points de vue multiples (la méthode ELECTRE). RIRO, 2, 57–75.

Saaty, T. (1996). *Decision Making with Dependence and Feedback: The Analytic Network Process*. Pittsburgh: RWS Publications

Sandia National Laboratory (2018). DAKOTA. Explore and predict with confidence. Project website: https://dakota.sandia.gov/, last accessed 22 Aug, 2018.

Schneider, P. and B. Schauer (2006). HAZUS—Its Development and its Future. *Natural Hazards Review*, **7**(2): 40-44.

SCIP (2018). Solving Constraint Integer Programs. http://scip.zib.de/, last accessed 22 Aug 2018.

SpatialHadoop (2018). A MapReduce Framework for Spatial Data. http://spatialhadoop.cs.umn.edu/, last accessed 22 Aug 2018.

Sugumaran, R., Meyer, J. and J. Davis (2004). A Web-based Environmental Decision Support System for Environmental Planning and Watershed Management, *Journal of Geographical Systems*, **6**(3):307-322.

Terzano, K. and V. Morckel (2016). SimCity in the Community Planning Classroom. *Journal of Planning Education and Research*, **37**(1): 95-105. doi: 10.1177/0739456X16628959.

UDST (Urban Data Science Toolkit) (2018). http://www.urbansim.com/udst/ and https://github.com/UDST, last accessed 20 Aug 2018.

UMEP (2018). Urban Multi-scale Environmental Predictor. Documentation and repository available online at https://umep-docs.readthedocs.io/en/latest/, last accessed 18 Sep 2018.

Vanegas, P., Cattrysse, D. and J. Van Orshoven (2011). A multiple criteria heuristic solution method for locating near to optimal contiguous and compact sites in raster maps. In Murgante, B., Borruso, G. and A. Lapucci (Eds.), *Geocomputation, sustainability and environmental planning: Studies in computational intelligence*, pp. 35–56. Berlin/Heidelberg: Springer.

von Neuman, J. and O. Morgenstern (1947). *Theory of Games and Economic Behavior*. Princeton, NJ: Princeton University Press.

Waddell, P., Boeing, G., Gardner, M. and E. Porter (2018). *An Integrated Pipeline Architecture for Modeling Urban Land Use, Travel Demand, and Traffic Assignment*. Technical Report for U.S. Department of Energy SMART Mobility Urban Science Pillar: Coupling Land Use Models and Network Flow Models. Berkeley, CA: University of California.

Wald, A. (1939). Contributions to the Theory of Statistical Estimation and Testing Hypotheses. *Annals of Mathematical Statistics*, **10**(4): 299-326.

WCS (Wildlife Conservation Society) (2018). *VisionMaker NYC*. Documentation and repository available online at https://visionmaker.us/nyc/, last accessed 22 Sep 2018.

Wierzbicki, A. (1998). Reference point methods in vector optimization and decision support. (Interim Report IR-98-017, Laxenburg, Austria: International Institute for Applied Systems Analysis.

WSBPEL (2017). *OASIS Web Services Business Process Execution Language*, version 2. Organization for the Advancement of Structured Information Standards. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, last accessed 22 Aug 2018.

Zeleny, M. (1982). Multiple Criteria Decision Making. New York: McGraw-Hill.