SIM-CITY: An e-Science Framework for Urban Assisted Decision Support

Borgdorff, J.; Krishna, H.; Lees, M.H.

DOI
10.1016/j.procs.2015.05.399

Publication date
2015

Document Version
Final published version

Published in
Procedia Computer Science

License
CC BY-NC-ND

Link to publication

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
SIM-CITY: an e-Science framework for urban assisted decision support

Joris Borgdorff1*, Harsha Krishna2, and Michael H. Lees3

1Netherlands eScience Center, Amsterdam, The Netherlands
j.borgdorff@esciencecenter.nl
2Fields of View, Bangalore, India
harsha@fieldsofview.in
3University of Amsterdam, Amsterdam, The Netherlands and ITMO, St. Petersburg, Russian Federation
m.h.lees@uva.nl

Abstract

Urban areas are characterised by high population densities and the resulting complex social dynamics. For urban planners to evaluate, analyse, and predict complex urban dynamics, a lot of scenarios and a large parameter space must be explored. In urban disasters, complex situations must be assessed in short notice. We propose the concept of an assisted decision support system to aid in these situations. The system interactively runs a scenario exploration, which evaluates scenarios and optimize for desired properties. We introduce the SIM-CITY architecture to run such interactive scenario explorations and highlight a use case for the architecture, an urban fire emergency response simulation in Bangalore.

Keywords: decision support system, urban decision support, agent-based simulation, traffic simulation, fire response model

1 Introduction

As cities grow, urban planning is essential to keep them habitable. This is especially true for rapidly urbanising countries such as India, where the urban population has increased fourfold in the last 50 years [1]. With such wildly growing urban areas, disasters such as fires require additional attention. For urban planners to make informed decisions, they need the tools to analyse the current, highly complex, situation and evaluate likely and significant scenarios.

Urban areas are characterised by high population densities and the resulting complex social dynamics. Disasters are prone to cause greater damage to urban areas. Microsimulation approaches are highly data-centric and have demonstrated their ability to capture and model social dynamics and
complex situations. However, employing these techniques is still a challenge in countries such as India due to the increased scale and lack of data and infrastructure. Analysing aspects such as traffic, environment or disease spreading using micro-simulation will invariably have to be done with a large number of parameters and incomplete data. Creating a helpful and faithful model requires fitting the parameters to the data, thereby capturing just enough complexity. Fire safety and fire emergency response is one of the basic services of a modern city. Modelling fire emergency response involves modelling not only the emergency infrastructure, but also road congestion and crowd response. This in turn is based on the social composition of a city, on the weather and the time of day. Running just a few simulations to evaluate all these factors is simply insufficient. Likewise, for urban planners to fully appreciate the effectiveness of an intervention, many possible scenarios resulting from that intervention must be evaluated. Finally, emergency teams may respond to disasters by using well-calibrated real-time simulations as a decision support system. Again, for testing possible outcomes of interventions, multiple scenarios should be evaluated at once by running simulations in parallel.

Computational techniques and tools such as Geographical Information Systems (GIS) and computational environmental models and simulations have been used in the development of urban infrastructure, including disaster management systems since the early 1980s [2]. Such systems have allowed for extensive prior planning and disaster preparedness, but generally focus on a small range of scenarios and parameters.

The emergence of data-driven simulation-based science can be argued as a natural response to technological developments in sensors, detectors and methods for data storage and manipulation [3]-[5]. Data is the new tool of science: Nigel Shadbolt estimated that the amount of data generated in 2010 would be around 1.2 million Petabytes [6]. It is only now through the combination of data availability and sophisticated computational methods such as multi-agent and complex network models, that truly data-intensive scientific discovery is possible.

In the SIM-CITY project, we propose an e-Science infrastructure for urban planners to evaluate, analyse, and predict complex urban dynamics in rapidly urbanising countries. The infrastructure aims to provide a general urban decision support framework, applied to e.g. emergency services, social mobility or epidemiology. It may launch simulations based on real-time data, and dynamically alter its course based on user input. The infrastructure in totality is dubbed an assisted decision support system.
In this contribution, the properties of such a system are explored and its technical implementation is discussed after a review of the relevant software. Its utility is then demonstrated with a fire response model for Bangalore. We believe the system will allow a large range of domain researchers to explore geographic phenomena in depth. For implementation, mainly open source software components are selected so researchers in developing countries are able to utilise it.

In Section 2, the properties of an assisted decision support system are discussed. In Section 3 an initial implementation of the SIM-CITY infrastructure is proposed. In Section 4, the use case urban fire response is posed and its relevance to the SIM-CITY architecture shown.

2 Assisted decision support

An assisted decision support system consists of modeling and simulation aspects combined with a number of technical components. The functional aspects of this system are shown in Figure 1, and its characteristics and motivations listed below.

First, the intrinsic characteristics of an assisted decision support system for an urban environment are:

1. The evaluation of complex scenarios by means of computer simulations.
2. The exploration of a large number of scenarios.
3. The steering of the exploration by an expert user and/or by sensor information.
4. Feedback with real-time analyses of current simulation ensembles.

Point 1 entails constructing a simulation environment for a given scenario, by combining a simulation engine with certain parameter settings or data sets. Choosing which phenomena to model and which parameters to include is a specialized task for a modeler who knows the application domain well. For large cities, these models generally include social, geographic and planning components. The complexity of these models implies the use of computer simulations.

Exploring a large number of scenarios, in point 2, means exploring parameter ranges, repeatedly doing stochastic initialization with fixed parameters, or testing combinations of a set of possibilities. The exploration may be done on the basis of testing all possibilities, determining the uncertainty of the simulations, or optimizing an evaluation function. The interactions in an urban environment are complex and non-linear, so a large number of simulations are necessary to explore the parameter space.

In point 3, steering consists both of changing parameter ranges or possible scenarios and of starting or stopping a simulation if a dynamic condition is met. The dynamic condition could be a sensor value reaching a certain threshold or a user determining where to further explore simulations. A user may want to start detailed simulations close to an optimal value or to decrease the number of possibilities by manually disabling unlikely results or by incorporating the real-time value of a certain parameter. To ensure timely results needed in emergency response, the scenario exploration must be able to evaluate multiple scenarios in parallel. The exploration may be a simple sweep over all parameter ranges, or otherwise an optimization algorithm or an uncertainty quantification algorithm.

Point 4, feedback to the user, may range from raw data to specific metrics from current simulation results. These may be statistics or plots from ensembles of simulations, or an overview of the actual results of individual simulations. As the number of simulations increase, individual results become less and less relevant but for the initial stages of modeling and conceptualization these individual stages are essential. Moreover, real-time feedback is necessary to have any impact on an emergency response.
The combination of points 3 and 4 makes an addition to traditional simulation and decision support approaches. Its interactivity allows for deep and timely insights in the dynamics of a phenomenon and its complexity.

Besides these intrinsic characteristics of the system, there are many technical and organizational challenges to such a system:

1. On the input side, a user interface should allow meaningful interaction with the system.
2. A comprehensive user interface is needed to explore all results.
3. A computational infrastructure must be available to run the simulations on.
4. Data must be well organized and data access times must be reasonable.
5. It must be easy to switch out models, scenario exploration algorithms, user interfaces, and computational resources.

The user interface to the decision support, point 1, must allow efficient input of parameter ranges and scenario possibilities. For geographic data, this could be through a point and click interface to add, remove or edit map features, while viewing other relevant geographic data, or even the results from earlier or current simulations. For numeric data, this could be through sliders, ranges or textual input. In any case, each type of data needs an appropriate input mechanism. Because this is an assisted decision support system, having a good user interface in points 1 and 2 is essential.

The computational resources in point 3 may vary from a cluster to HPC to cloud; in all cases, many simulations must be done. If these simulations are short relative to the queuing time of the system, using a Pilot-Job framework is necessary. Such a framework starts a relatively small number of jobs on a cluster to process a large number of tasks, in order to avoid queuing time for each individual task. On the cloud, such a Pilot-Job framework may be used as the sole queuing mechanism, since virtual machines do not have a queue by default.

Moreover, the data resources in point 4 may vary, but to a run a large number of simulations data must be organized so simulation results can be interpreted after an exploration has ended. In the simplest case, this involves only file naming conventions, but for a large user base a provenance system should be used.

Recent planning methodologies are already adapting to techniques with more dynamic parameter space in order to be able to respond to such changes that will affect their planning methodologies. For example, Transit oriented development is an urban planning method used for highly density mixed use planning in urban areas to address the problem of last mile connectivity by increasing pedestrian connectivity [7]. This method relies on simulation of the complex interaction among the various stakeholders in urban planning and urban transportation.

Jain and McLean [8] describe a framework for integration of distributed simulation models. Their focus is on a framework that also does fast dissemination of actions to emergency response teams, and the embedding of the framework between stakeholders.

A fine example of a full framework of assisted decision support, also having a strong planning and geographic component, is the UrbanFlood project [9]. As the infrastructure they used Common Information Space. This infrastructure is able to schedule simulations, dynamically compute selected metrics, and include real-time sensor data. An interactive geographic user interface seems to be specifically crafted for this project, and is not part of the Common Information Space.

Another example is an assisted decision support system for surge floods in St. Petersburg using CLAVIRE [10]-[12]. CLAVIRE plays a similar role as Common Information Space. In addition to a scheduler, they also have a communication library for direct visualization of running simulations.
3 SIM-CITY infrastructure

As the basis of the SIM-CITY infrastructure we propose the setup shown in Figure 2. In this architecture, the simulations are modified and started from a web interface, which alongside output of running and finished simulations. Components are mainly coupled using JSON messages for easy testing and replacement.

Initially, each component will be configured for the urban fire response model. As we progress, the components will be made more configurable and automatized, allowing different types of models. The web interface, if built with modular blocks, may be largely reused by selecting relevant blocks per simulation type. The scenario exploration can be prototyped quickly by making use of basic external packages. The software components listed below are selected for their functionality and for the availability of local experience or support.

3.1 Web interface

Many GIS desktop interfaces exist, but open source interfaces have limited functionality and unclear extensibility. A commercial tool that seems to have it all is ArcGIS [13]. Web-based GIS interfaces are increasingly practical. Web-based interfaces require a solid Internet connection and are only possible with limited data sizes. A gallery from cartodb.com\(^1\) shows the possibilities of web based geographical interfaces. A nice example of an interface includes New York transit time\(^2\), which uses Mapbox as an interface, and OpenStreetMap as data provider, and the Leaflet JavaScript library as a layout engine. OpenLayers is a similar JavaScript library as Leaflet, but more mature and heavyweight. For worldwide geographic data, the Cesium JavaScript library supports 3D global views.

For storing and serving geographical data, GeoServer [14] and MapServer [15] are two open and established options. These software products can show geographic data directly from a PostGIS database and do some graphical post-processing, like clustering or coloring. In our architecture we chose GeoServer based on previous experience with the tool.

An example of a full web interface is Common Sense Web [16], incorporating a Leaflet GIS, tabular data and some simple graphs. Internally, it has a number of web components that can be combined to a user interface. However, in-depth customization would be necessary to allow data input and modification and for showing custom metrics. Mapbox provides a user interface builder that ties users to their services.

The SIM-CITY web interface will be fashioned after the Common Sense Web interface. It will consist of a number of building blocks, including an interactive OpenLayers 3 GIS, graphs, some runtime statistics and a number of simulation metrics. Parameters may be input directly on the user interface, or uploaded as a file. In the GIS, selected geographic features that are the output of a simulation may be shown and edited to form the input for another simulation. Individual simulations can be selected to show or download only its results, or a set of simulations may be selected and their aggregate results shown.

3.2 Scenario exploration

The most prominent part of assisted decision support system, parameter exploration, in the simplest case can be done with parameter sweeps, but for complex models likely inverse modeling and global optimization techniques will be necessary. For large-scale parameter exploration, such a framework should be able to start simulations asynchronously and in parallel in order to explore enough scenarios. There are a number of packages with enough exploration algorithms, for Python

\(^1\) URL: http://www.cartodb.com/gallery/
\(^2\) URL: http://project.wnyc.org/transit-time/
examples are: mystic, OpenOpt, emcee, ecsipy, or pypet; for R: FME, adaptMCMC, PGA. To truly allow parallel asynchronous evaluation, some adaptations to the packages will be necessary. For example the MMSODA toolbox for MATLAB [17] does achieve this.

We propose a Python controller to do the actual scenario exploration. It will have built-in exploration algorithms, to begin with a simple parameter sweep. At a later stage it may include user-supplied algorithms and advanced parameter exploration algorithms, possibly with an interface to R. The scenario exploration will create parameter sets and in time create input files. Files are transferred to the computing resources using Osmium; parameter sets are stored in a CouchDB database, as part of the pilot-job script PiCaS. Once all input is ready, a cluster job is started with Osmium to do the actual simulations. In principle, multiple scenario explorations may be running simultaneously. Data from individual simulations may end up as files, while aggregate data and statistics can be added to the running PostgreSQL database.

With a small amount of users, all explorations may be expected to run on a single virtual machine. At a later stage, however, each exploration could run in its own virtual machine if necessary. This would require an adaptation in how the python controller server starts a new exploration.

3.3 Computing infrastructure

The actual SIM-CITY infrastructure will initially be hosted on a single server, a single virtual machine in fact. It will consist of a Nginx web server for the web interface, with a Python backend, a Geoserver for serving geographic data from a PostGIS database, a Python controller server that can manage simulations and scenario explorations, and finally an Osmium server to interface with the
computing infrastructure. Each of the aforementioned services could also be run on separate machines if they need to scale to more users.

To avoid long queuing times on the cluster, simulations are done within a pilot-job script. PiCaS [18] can run a large number of short-running simulations within a single job, reading parameter sets from the CouchDB database [19].

Alternative pilot job software includes RADICAL-Pilot [20], which is a fully integrated framework, or writing a script on top of a thread-safe message queue such as RabbitMQ. The advantage of CouchDB and RADICAL-Pilot, is that they also take care of output handling; the advantage of scripts with RabbitMQ or CouchDB is that they are lightweight and generally easy to install and modify. However, they do require Internet access during job execution.

A general middleware library that can interface with multiple types of distributed infrastructure is Xenon [21]. It is a relatively simple library since it aims for the lowest denominator of infrastructure capabilities. Xenon can be run as a JSON web service, called Osmium, which allows loose coupling to other parts of an infrastructure.

The Common Information Space and its successors, a JavaScript workflow engine, Hyperflow, and a cloud scheduler, Atmosphere, form an alternative to Xenon. Another alternative is CLAVIRE. Both are described in Section 2. Especially when urgent computing for disaster response is taken into account, using the cloud may avoid queuing times present in many cluster resources.

3.4 Real-time data

The implementation outlined above can be considered to lay close to existing decision support frameworks. The real added value lies in being able to automatically start a new scenario exploration or interrupt one that is running. This may happen whenever an event such as a mouse click occurs or when a sensor exceeds a given threshold. The parameter exploration server may facilitate this by acting as a central message bus. Users can then register events that should trigger an exploration with given parameter ranges. In addition, users can register events with a running exploration. A user-defined Python script should then determine what action the parameter exploration should take.

4 Use case: fire response

The main demonstrator for the SIM-CITY assisted decision support system will be an application for fire risk assessment, mitigation, and resolution in Bangalore (Bengaluru), India. This rapidly growing city with 8.5 million citizens [22], suffers from a straggling infrastructure with insufficient fire stations and increasing congestion. The city’s emergency response infrastructure has not been able to keep up with the rising demands and complexity of a large metropolitan response system thus, the city was unprepared for even the most basic of domestic emergencies. The level of preparedness was demonstrated during a fire incident at a location called Carlton Towers. In 2010, Bangalore experienced one of its first high-rise fires in one of its many business districts, in a building called the Carlton Towers [23]. India faces a shortage 97% in the number of required fire stations, an 80% shortage in its number of fire fighting vehicles and a 96% shortage in the number of fire fighters [24].

The speed of growth for the urban areas is so rapid, that by the time a fire station is designed and built, the surrounding environment and the level of demand would have completely changed. Due to the mixed nature of land use it is extremely difficult to estimate the nature of change. The planning situation resembles the construct of a wicked problem as described by Rittel and Webber [25] where the parameters for planning keep changing. Interventions to improve response times for fire emergency teams to fires include building additional fire stations, relocating existing fire stations or blocking roads to improve the response time of current fire responders. Road and transport network modifications may be used to decrease traffic or travel time between a fire station and the affected
areas. It is also used to limit the size of crowds gathering around a fire. Much of the population growth in Bangalore has been due to migration of people from different parts of the country. The increase in size and diversity of population has added to the complexity of urban design.

As an intervention, blocking roads may increase access of fire emergency teams by decreasing traffic between a fire station and afflicted areas and to limit the size of crowds gathering around a fire. Notable urban factors to be taken into account are the variation of congestion over time and the distribution of fire hazard over the city. An accurate model must take into account these factors and a framework must be able to incorporate interventions under varying conditions.

The decisions are therefore done in both offline planning and online reactive decisions that happen during a fire outbreak. Central to this decision is the dynamic traffic conditions of the city, depending on the current traffic conditions different decisions may be optimal at different times. Also the response of the general public to the emergency can also impact the ability of the emergency responders to arrive on time, like with the Carlton Tower fires.

The purpose of the infrastructure is not to improve on those but to be able to modularly include them. TRANSIMS [26] is one such micro-simulation based traffic simulation engine. Other tools include MATSIM, PTV Vissim [27] and DynaMIT [28]. Among these packages, TRANSIMS is an open source tool with modular simulation capabilities. Its modularity can be used to replace part of a simulation with a custom simulator. The total set of data that can be accessed, manipulated and analysed by this framework for the current use case is summarised in Figure 3.

Parameters that may be explored are number of people travelling, depending on time and weather, as well as transport mode and vehicle behaviour. The number of fire stations may be varied as well as their location. In every scenario, a fire is started at a certain location, using a probability distribution based on the fire hazard map. When a fire is started, spectators may crowd at the site of the fire with different algorithms, and different roads can be selected to be blocked by the police after a certain amount of time after the fire has started. The scenarios may be optimized for their fit to traffic data and known fire emergency response arrival time, as well as to desired values for arrival time. When the location of a fire is known, only scenarios with a fire on that location need to be simulated, thus greatly reducing the search space and decreasing response time.
Integrating this system in the SIM-CITY architecture requires automatic pre- and post-processing of simulations and simulation ensembles, as well as a custom user interface. These steps likely will need to be repeated for other use cases, while keeping the rest of the architecture stable.

5 Conclusions and future work

In this contribution, we have outlined the SIM-CITY architecture for urban assisted decision support. The architecture can help researchers to validate and tune their models, and provide timely feedback in the case of emergencies. Its novelty lies in the automated and steerable scenario explorations, running a large number of simulations in parallel. This automation will be crucial to understand large complex systems like urban regions.

The use case that demonstrates the utility of this architecture is a fire response model for Bangalore. It will incorporate traffic in Bangalore under different conditions with a fire at a stochastic time and location, and an emergency response by fire engines and police. The traffic simulation can be validated using known traffic conditions, the fire emergency response validated based on historical emergency response behaviour.

In next iterations of this architecture, we aim to increase the amount of real-time data incorporated in the model, and the amount of feedback that may be given to stakeholders in real time. This will require several user interfaces, each catered for the specific stakeholder.

The number of use cases will also be increased to epidemiology and poverty modelling, both extremely relevant for the city of Bangalore, India.

References


