RODGEN: An Interactive Interface for Road Network Generation

Claudia Perez Martinez
Johannes Gutenberg University
Mainz, Germany
cperezma@students.uni-mainz.de

Panagiotis Bouros
Johannes Gutenberg University
Mainz, Germany
bouros@uni-mainz.de

Theodoros Chondrogiannis
University of Konstanz, Germany
theodoros.chondrogiannis@uni.kn

ABSTRACT

We present RODGEN, an interactive, graphical user interface for generating road networks that adopts the growth-based model. The first step in the generation process is to construct the backbone of the network by either choosing between a grid-based and a ring-based predefined topology or allowing the users to define a custom one. The backbone divides the space into a number of areas, called neighborhoods. The user can populate neighborhoods either by importing existing road networks or adding roads by hand. Besides generating road networks, our interface also provides a platform for analysis. For this purpose, we employ a general-purpose graph analytics library, which allows the users to compute graph statistics, perform connectivity analysis and execute basic routing tasks.

CCS CONCEPTS

• **Human-centered computing** → **Graphical user interfaces**; **Graph drawings**; • **Information systems** → *Geographic information systems.*

KEYWORDS

Road networks, graphs, generator, user interface

ACM Reference Format:
Claudia Perez Martinez, Panagiotis Bouros, and Theodoros Chondrogiannis. 2022. RODGEN: An Interactive Interface for Road Network Generation. In *The 30th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '22), November 1–4, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3557915.3560989

1 INTRODUCTION

Graphs are ubiquitous for modeling data. *Road networks* are an example of using graphs to model road information and connections e.g., between parts of a city. Several generators have been proposed in the past for graphs, e.g., the random model [4], the small-world model [18], the preferential attachment [1, 9] and the Kronecker matrix multiplication [10], but generating road networks has received less attention. Despite, the availability of real road networks on the Web, e.g., extracted from the OpenStreetMap (OSM) project[1] or uploaded on files repositories[2], generated road networks can find application in multiple scenarios:

- **Query processing**. Synthetic road networks can be used for benchmarking (e.g., [15]) the efficiency and the robustness of routing algorithms, e.g., for shortest or fastest path queries. An extensive performance comparison of the algorithms can be conducted by studying the impact of parameters/factors such as the network size and its topology.
- **Digital entertainment**. In digital entertainment industry, game design and Virtual Reality require the generation of realistic road networks, that mimic the characteristics and the structure of real-world networks.
- **Analytics**. The generation of road networks is an important task in transportation analysis for identifying significant characteristics in networks [17].

Previous work. *Procedural* generative methods are among the first techniques used for road network generation. Growth-based models such as [2, 7] create an initial set of primary roads which are then filled with secondary, to create city blocks. The authors in [12] modelled road networks based on L-systems [13]; starting from a single road segment they add more segments to grow a road network, similar to growing a tree. In [3], an interactive model based on tensor fields was proposed. These sensory fields serve as a guide for the generation of the road network by taking into account the terrain and population density. The user can interact in each step of the process and perform high-level modeling operations, such as adding, deleting and modifying roads. As points of criticism, the above techniques employ multiple steps and parameters which need to be carefully and finely tuned to obtain quality results. Further, in many cases they offer little customization options and so, users are often unable to control the types of roads being generated.

Deep generative models [14] that rely on neural networks, have been also developed for road network generation. [6, 8, 11]. StreetGAN [6] builds upon Generative Adversarial Networks (GANs) to create realistic road networks, requiring as input, a real network, e.g., extracted from OSM. RoadNetGAN [11] employs the network in its graph form to feed the generator neural network. The generator learns the distribution of the network through random walks and displacement attributes, and outputs sequences of nodes with their spatial position. These sequences are then used to generate a planar road network, with a structure similar to the original one. As a point of criticism, deep generative methods do not usually offer a graphical user interface and therefore do not allow direct visualization and interaction with the users. Nevertheless, we plan to investigate how such methods can be incorporated in our interface.

Contributions. In this paper, we present RODGEN[3], an interactive graphical user interface for road network generation, which serves a twofold mission. First, it allows users to graphically construct and generate road networks. We build upon the concept of *backbone*

[1] https://www.openstreetmap.org
[2] https://figshare.com/articles/dataset/Urban_Road_Network_Data/2061897, http://www.diag.uniroma1.it/challenge9/

[3] https://rodgentool.github.io/

SIGSPATIAL '22, November 1–4, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9529-8/22/11.
https://doi.org/10.1145/3557915.3560989

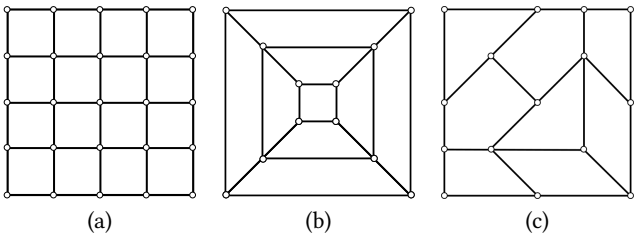


Figure 1: Different types of backbone networks; (a) a 4×4 grid-based, (b) a ring-based of degree 2, and (c) a custom, user-defined backbone.

networks or simply *backbones* previously used in [15], as collections of primary roads (similarly to [2, 7]) that allow for fast traveling between different parts of a city. The backbone essentially defines the basic topology of the generated road network, dividing it into a number of areas called *neighborhoods*. RODGEN offers different options to the users for graphically constructing the backbone of a road network, including two predefined types, and for populating the inner network of each neighborhood, either by manually adding roads or by importing existing networks from external sources. Besides generating road networks, our interface also offers a platform for executing a preliminary graph analysis. The goal is to understand the nature and the structure of the generated network, which offers an intuition how realistic the network is. In addition, our interface can answer basic path-finding queries. For this purpose, the interface communicates with our back-end where NetworkX[4], a general-purpose graph analytics Python library, is installed.

2 NETWORK GENERATION

In what follows, we detail the process of generating road networks, presenting the key features and functionality of RODGEN.

Backbone construction. The process starts with the construction of the backbone network. Our interface offers three options for this purpose. As the first option, the user can choose between two predefined types of backbones, named grid-based and ring-based, which are inspired by real-world road networks such as the ones in many US cities and in many European cities, e.g., Vienna, respectively, Figures 1(a) and (b) exemplify these two types of predefined backbones. For a grid-based backbone, the $n \times m$ granularity of the grid is provided as a user-defined input parameter. The interface then creates a backbone that contains $n \cdot m$ neighborhoods and $(n+1) \cdot (m+1)$ road intersections. Figure 2 illustrates the construction screen of a 2×3 grid-based backbone. For the ring-based, the user inputs the degree δ and the interface creates a backbone of $4 \cdot (\delta - 1) + 1$ neighborhoods and $4 \cdot \delta$ intersections. The second option allows users to import an existing road network which will serve as the backbone for the new. Currently, the user can import the network from a `.gr` formatted file which contains the road segments of the network as graph edges, along with a `.co` file that contains the coordinates of the road intersections[5]. Alternatively, this road network can be extracted from OSM; we exemplify this extraction task in the next paragraph, when we discuss how to

[4]https://networkx.org
[5]Details on the format of these files in http://www.diag.uniroma1.it/challenge9/.

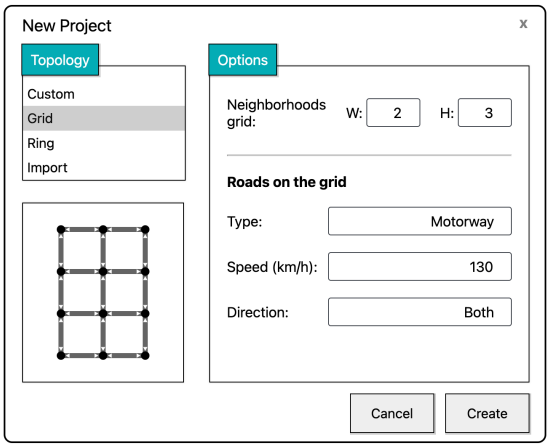


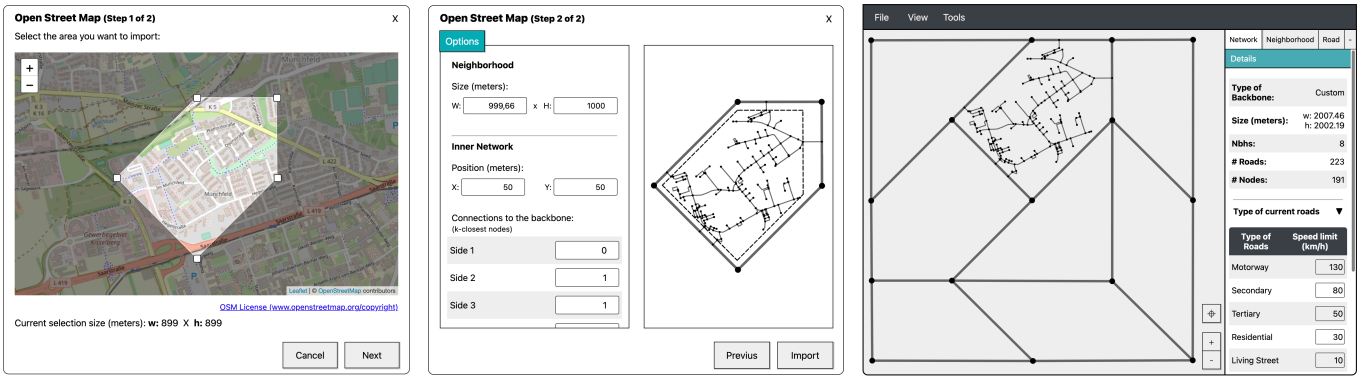
Figure 2: Creating a grid-based backbone network

populate neighborhoods. Lastly, the user can also define a custom backbone by drawing a set of lines as the backbone roads, e.g., the custom backbone in Figure 1(c). In this case, RODGEN automatically creates an intersection for every pair of intersecting backbone roads, rendering the backbone a planar network.

Managing neighborhoods. After constructing the backbone, RODGEN automatically determines the set of contained neighborhoods as closed polygons. Users maintain full control over this set, essentially allowing them to modify the neighborhoods and hence, adjust or expand the backbone at will. Specifically, existing neighborhoods can be split or merged by adding or deleting backbone roads, respectively, while entire new neighborhoods can be also created. Under this premise, the user can start by constructing a backbone of predefined type (e.g., a grid-based) and then alter its topology resulting into a new custom backbone.

Every neighborhood can be populated with an inner network. For this purpose, the same two options to import a backbone network, are offered to the users. However, as a backbone already exists, the user also needs to specify how the inner network of a neighborhood is connected to the backbone. Figure 3 illustrates how to populate the inner network of a neighborhood, using an OSM extract. As the first step, the user navigates to the area of the map where from the road network should be extracted (Figure 3(a)). In the second step (Figure 3(b)), the user can specify the dimensions (in meters) of the entire neighborhood; the numbers are initially set according to the dimensions of the extracted network from OSM. In addition, the user can specify how many connections to the backbone should be created on each neighborhood side. These connections are automatically materialized as new, bi-directional road segments from the closest intersections of the inner network to the backbone roads on each side. To ensure that the inner network is strongly connected to the backbone, one connection is created by default. Finally, users can also modify the positioning of the inner network inside the neighborhood. Figure 3(c) shows the network after populating one of its neighborhoods.

Managing roads. RODGEN distinguishes between two types of roads, namely backbone and inner network roads. The user is allowed to add and delete roads of both types. In case of backbone



(a) (b) (c)

Figure 3: Populating a neighborhood from the backbone in Figure 1(c).

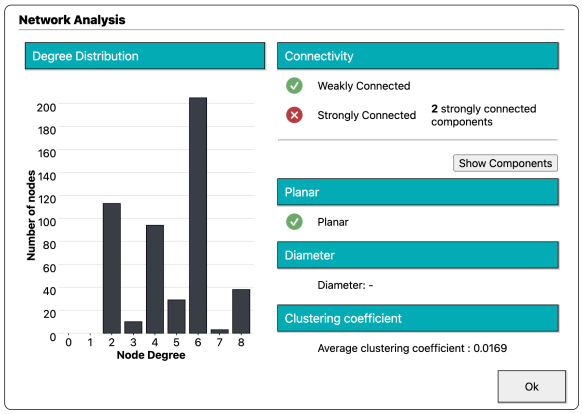
roads, such modifications also affect the neighborhoods of the network and therefore, its topology. It is also possible to modify the characteristics of a road (both backbone and inner); specifically, its direction, the speed limit and its class. For the latter, we consider the classes defined in OSM[6]. Figure 2 shows the characteristics of the roads included in the 2×3 grid-based backbone. By default, backbone roads are bi-directional roads of `motorway` class, but the user can modify these characteristics either collectively for all backbone roads in the screen of Figure 2 or by clicking on a road in the main screen of the interface.

Coordinates system. For visualizing the generated network, RODGEN defines a Cartesian two-dimensional coordinates system. The user can expand or shrink the extent of the network in this two-dimensional space by specifying the dimensions of the contained neighborhoods and the length of the contained roads. The coordinates system also models the geometry of the roads as lines; the user can decide whether only road intersections or all road points are shown. Finally, when an existing network is imported from an external source (i.e., to serve as backbone or to populate a neighborhood), the interface automatically transforms the coordinates of its intersections to the underlying coordinates system.

3 NETWORK ANALYSIS

We now switch our focus to the analysis tasks. For this purpose, RODGEN connects to the back-end via Django[7], and passes the generated road network as input to the NetworkX methods.

Statistics and analysis. The goal of our analysis is to compute key graph characteristics of the generated network in an effort to understand how well the network mimics real-world road networks. For this purpose, we compute the degree distribution of the road intersections, the diameter of the network and its clustering coefficient. In addition, we check whether the network is planar and conduct a connectivity analysis. Figure 4(a) displays the results of our analysis for the ring-based road network in Figure 4(b). As expected in a road network, the degree of the road intersections does not exceed 8, which corresponds to the case of connecting 4 bi-directional roads. We also observe that the network is planar and exhibits a low clustering coefficient, also typical for road networks. In this particular example, the generated network is weakly but not strongly connected. In such a case, the interface informs on the number of strongly connected components identified.



(a)

(b)

Figure 4: Network analysis: (a) the results of our network analysis, (b) details on connectivity analysis

[6]https://wiki.openstreetmap.org/wiki/Key:highway
[7]https://www.djangoproject.com

Connectivity analysis. If the generated road network is not strongly connected, by clicking on the "Show Components" button in Figure 4(a), the user returns to the main screen of the interface where all strongly connected components are now highlighted. Figure 4(b) illustrates the results of the connectivity analysis. The two components are colored in green and yellow. In addition, the interface colors in red the road segments responsible for breaking the strong connectivity property. Under this premise, the user can either modify the direction of these segments or add new roads, in order to ensure that the final generated graph is strongly connected.

Path-finding. Finally, users can use the generated road network to test path-finding algorithms. Figure 5 illustrates the computation of the shortest (in blue color) and the fasted path (in beige) between two neighborhoods of the network. For the fastest path, we assume travelling at the maximum allowed speed on each road segment. The figure exemplifies the impact of the backbone network. Travelling longer on the backbone roads which typically have higher speed limits, usually reduces the total travel time while travelling through the neighborhoods usually results in reducing the total distance.

4 DEMONSTRATION SCENARIOS

We plan to demonstrate the creation of networks and the analysis platform of the interface. For this purpose, we will use different types of backbones and populate the inner network of the defined neighborhoods, e.g., by extracting networks from OSM. The attendees will be also able to interact with the interface by modifying roads and neighborhoods, and to use the project managing and export (to `.gr` and `.co` files) features. Regarding the analysis tasks, besides computing the statistics of the generated road network, we will elaborate on how to deal with strongly disconnected networks, e.g., by adding new roads or altering existing. Lastly, the attendees will be also to use the path-finding tasks to study the impact of different types of backbone topologies.

5 CONCLUSIONS

We presented an interactive, graphical user interface for generating road networks. RODGEN also offers a platform for analysis tasks to determine graph characteristics of the network and for basic path-finding, i.e., shortest and fastest path computation. In the future, we intend to extend our work towards multiple directions. First, we plan to further investigate how realistic are the generated networks. For this purpose, we will consider additional graph statistics (e.g., the highway dimension) and similarity measures for comparing generated networks to a repository of existing real road networks. An interesting idea for the latter is to study the effectiveness of path-finding techniques specialized for road networks, e.g., the contraction hierarchies in [5]. Moreover, we plan to enrich the generation process by including additional types of predefined backbone topologies, considering deep generative models, e.g., for populating the inner network of a neighborhood, and allowing user to define meta-data for roads and nodes. Finally, we also intend to incorporate additional functionality by integrating external tools, e.g., the CityFlow [19] for traffic generation or CASPER [16] for evacuation planning.

Figure 5: Computing shortest (in blue) and fastest path (in beige); shortest path has a length of 3.15 km and 3 mins travel time, fastest path has 4.02 km and 2 mins, respectively.

ACKNOWLEDGMENTS

Partially supported by Grant No. CH 2464/1-1 of Deutsche Forschungsgemeinschaft (DFG). Panagiotis Bouros is a Carl-Zeiss Stiftungsprofessor for "Big Data: In-Memory Databases and Data Analytics".

REFERENCES

[1] Albert-Laszlo Barabasi and Reka Albert. 1999. Emergence of Scaling in Random Networks. *Science* 286, 5439 (1999), 509–512.
[2] Jan Benes, Alexander Wilkie, and Jaroslav Krivánek. 2014. Procedural Modelling of Urban Road Networks. *Comput. Graph. Forum* 33, 6 (2014), 132–142.
[3] Guoning Chen, Gregory Esch, Peter Wonka, Pascal Müller, and Eugene Zhang. 2008. Interactive procedural street modeling. *ACM Trans. Graph.* 27, 3 (2008).
[4] Paul Erdos and Alfred Renyi. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hungary. Acad. Sci.* 5 (1960), 17–61.
[5] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. 2008. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In *WEA*. 319–333.
[6] Stefan Hartmann, Michael Weinmann, Raoul Wessel, and Reinhard Klein. 2017. StreetGAN: Towards Road Network Synthesis with Generative Adversarial Networks. In *WSCG*. 133–142.
[7] George Kelly and Hugh McCabe. 2007. Citygen: An Interactive System for Procedural City Generation. In *GDTW*. 8–16.
[8] Lin Ziwen Kelvin and Anand Bhojan. 2020. Procedural Generation of Roads with Conditional Generative Adversarial Networks. In *ACM SIGGRAPH*. 12:1–12:2.
[9] Jon M. Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. 1999. The Web as a Graph: Measurements, Models, and Methods. In *COCOON*. 1–17.
[10] Jure Leskovec, Deepayan Chakrabarti, Jon M. Kleinberg, and Christos Faloutsos. 2005. Realistic, Mathematically Tractable Graph Generation and Evolution, Using Kronecker Multiplication. In *PKDD*. 133–145.
[11] Takashi Owaki and Takashi Machida. 2020. RoadNetGAN: Generating Road Networks in Planar Graph Representation. In *ICONIP*. 535–543.
[12] Yoav I. H. Parish and Pascal Müller. 2001. Procedural modeling of cities. In *ACM SIGGRAPH*. 301–308.
[13] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. 1990. *The algorithmic beauty of plants*. Springer.
[14] Lars Ruthotto and Eldad Haber. 2021. An Introduction to Deep Generative Modeling. *CoRR* abs/2103.05180 (2021). https://arxiv.org/abs/2103.05180
[15] Dimitris Sacharidis and Panagiotis Bouros. 2013. Routing directions: keeping it fast and simple. In *ACM SIGSPATIAL*. 164–173.
[16] Kaveh Shahabi and John P. Wilson. 2014. CASPER: Intelligent capacity-aware evacuation routing. *Comput. Environ. Urban Syst.* 46 (2014), 12–24.
[17] Philippe Y. R. Sohouenou, Panayotis Christidis, Aris Christodoulou, Luis A. C. Neves, and Davide Lo Presti. 2020. Using a random road graph model to understand road networks robustness to link failures. *Int. J. Crit. Infrastructure Prot.* 29 (2020), 100353.
[18] Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393, 6684 (1998), 440–442.
[19] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. 2019. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. In *ACM WWW*. 3620–3624.