TownSim: Agent-based city evolution for naturalistic road network generation

Asiiah Song
University of California, Santa Cruz
julinas@ucsc.edu

Jim Whitehead
University of California, Santa Cruz
ejw@ucsc.edu

ABSTRACT

We describe an agent-based city evolution algorithm creating road networks over time, and explore several approaches for analyzing the malleability of the algorithm to exposed parameters. In addition to qualitatively assessing the generated content, we look at the directionality, connectivity, and curvature of the generated road networks.

CCS CONCEPTS

• **Computing methodologies** → *Agent / discrete models.*

KEYWORDS

road network generation, virtual city generation

ACM Reference Format:
Asiiah Song and Jim Whitehead. 2019. TownSim: Agent-based city evolution for naturalistic road network generation. In *The Fourteenth International Conference on the Foundations of Digital Games (FDG '19), August 26–30, 2019, San Luis Obispo, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3337722.3341852

1 INTRODUCTION

Procedurally generated towns and cities are surprisingly useful. When applied to games, they permit the gameworld to have large amounts of procedural cities [1], or reduce authoring effort for cities that are generated first and then modified by a level designer. *The Sinking City* exemplifies this latter approach, using a procedurally generated 1920s Boston filled with streets, buildings, and submerged areas as a starting point, which is enhanced to meet gameplay goals [34]. *Sunset Overdrive* was developed using a procedural road system, where a human designer specifies roads using curves, from which road, intersection, sidewalk, and gutter geometry are generated [25]. Both projects report substantial improvements in design iteration speed.

In the domain of urban planning, procedural city generators allow planners and architects to explore different urban planning scenarios. Architecture firm Houseal Lavigne used ESRI's *CityEngine* to explore and visualize plans for housing developments in Woodbridge, Illinois, highway landscape buffers in Tulsa, Oklahoma, and a new entertainment district in Oshkosh, Wisconsin [18][2].

FDG '19, August 26–30, 2019, San Luis Obispo, CA, USA

ACM ISBN 978-1-4503-7217-6/19/08…$15.00
https://doi.org/10.1145/3337722.3341852

Autonomous vehicles are an important new application area for procedural cities. Testing autonomous vehicle software in the real world using physical cars and streets is important, but also time consuming and expensive. Companies developing autonomous vehicle software supplement real world testing with simulation based testing in virtual street scenes. *Waymo* is an autonomous vehicle company (a subsidiary of Alphabet) which uses a virtual environment called Carcraft to supplement real-world vehicle testing. As of October 2018, Waymo had logged 10 million miles in the real world, as compared to almost 7 billion miles in simulation [7][21].

While many virtual road networks are obtained by importing real-world road networks from OpenStreetMap or other sources of map information, procedural (synthetic) road networks have an important role to play. Procedural roads can create situations not present in the real world, or which occur very infrequently. They can also simulate road configurations found in different parts of the world. Ideally, road networks could be generated to focus on situations that autonomous vehicle software finds challenging, and hence fully exercise problematic aspects of the software. In the present work, we are primarily motivated by the goal of generating road networks suitable for testing autonomous vehicle software, though our generated towns also have the potential for use in computer games and urban planning.

While there have been many road network generators described in the research literature, they tend to have three drawbacks that make them less than ideal for autonomous vehicle testing. First, the generated roads often look too regular or tidy, and hence artificial. Real world road networks have unusual road situations related to having evolved over many years of development and use. They don't always make sense. We wish to capture some of this real-world naturalness in our generated towns. This has led us to adopt a city generation approach that evolves a city over time, instead of a strictly construction-based approach. Second, existing road network generators do not output streets in formats usable by traffic simulation software, such as the open source *Simulation of Urban Mobility (SUMO)* package. Finally, existing road network generators have not had an expressive range analysis performed on their output [28]. There is hence limited (or no) information available that can be used to guide the generator towards certain kinds of output. For the purpose of testing, being able to intentionally generate certain kinds of road networks is a desirable property.

In the current paper, we present an agent-based town simulator called TownSim, which is capable of generating a wide range of natural-looking street layouts. An expressive range analysis of TownSim provides a detailed understanding of how it responds to various changes to its input parameters, focusing on directionality, connectivity, and curvature of generated road networks.

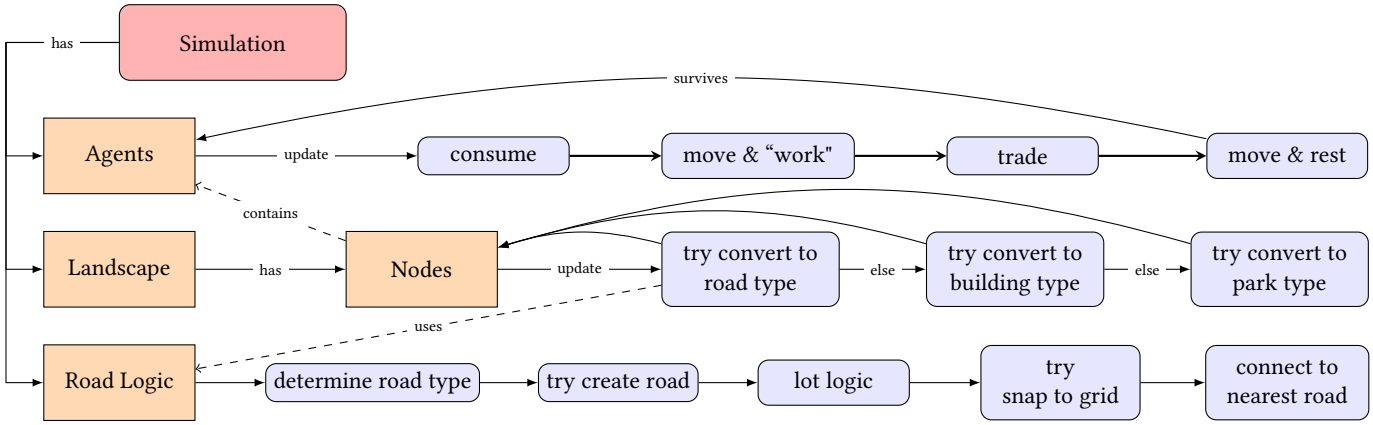

Figure 1: A simplified flow chart of the algorithm.

2 RELATED WORK

Kim, Kavak and Crooks recently argued that procedural roadmap generation can be used beyond games and in real-world applications such as social simulation, and urban testbeds [17]. Surveys of road network generators can be found in [26][14]. Our work was inspired by Batty's research on patterns in complex real-world cities [4] and Emilien et al.'s work on generating naturalistic rural-like settlements adapted to hilly terrain [8]—we're ultimately interested in generating naturalistic urban settlements adapted to terrain. Some of the same authors in [8] (Galin, Peytavie) wrote earlier papers on creation of road networks adapted to terrain, in particular rural highways [9][10].

2.1 Map data-based methods

Parish and Müller's CityEngine is an early influential paper, and uses extended L-systems based on socio-statistical maps containing information of water bodies, elevation, and population density [23]. Karagiorgou et al. directly extracts real-world road networks using vehicle tracking data segmented at multiple granularities [13][12]. Sun et al. [29], CitiGen [15], and Nishida et al. [22] are "template-based", pulling from templates found in the real world. CitiGen and Nishida et al.'s methods can both be described as sampling real-world road patterns which are then placed in maps and grown to connect to each other.

2.2 Agent-based methods

Lechner et al. is an early paper which featured many agents that drive their generation algorithm, but utilized what was essentially a top-down approach [19]. Land developer agents are driven by profit to develop parcels and connect those parcels via roads, with residential, commercial, and industrial zones which each afford different profitability calculations. Road patterns are controlled by explicit griddedness-variables applied directly to an area of a map, also allowing users to erase areas that are zoned undesirably.

Vanegas et al. generates city maps based on population density and predefined elements, such as user-defined placement of particular roads. Population density here is not taken from a real-world socio-statistical map, but rather simulated agents which are placed on the map following a value and accessibility algorithm [30].

Figure 2: Examples of 100x100 generated maps: major roads are black, minor roads are gray, light yellow areas are buildings, blue is water, and green is vegetation.

2.3 Timeline-based methods

UrbanSim is an example of urban modelling meant to model the future evolution of existing urban communities for city planning, using detailed demographics and zoning and development information [32]. Williams and Headleand use Unity3D to produce a cityscape that developed over different time periods, where each time period would have different, user-definable architectural styles and road patterns [33].

2.4 Other methods

Approaches with a mathematical bent include Chen et al., based on tensor fields [6], and Campos et al. which uses underlying graph structures to create 3-dimensional road networks [5]. Kim et al. uses SMT-based constraint solving methods [16]. Roglà et al. uses semantically tagged component elements to procedurally compose virtual cities [24]. Vanegas et al. discusses interesting work on procedural division of lots into parcels [31].

Hartmann et al. uses generative adversarial networks in what is essentially style transfer applied to road network generation [11]. Lipp et al. [20] and Smelik et al. [27] focus on interactive authoring of road networks subject to geometric and terrain constraints.

3 SIMULATION ALGORITHM

At a high level, the TownSim simulation algorithm (Figure 1) enables naturalistic growth of roads by allowing simulated agents to tread their own paths. The resulting road network is a record of agent

activity rather than assigned development. Agents are motivated by a consume-work-trade-rest routine loosely inspired by the day-night routines of human workers. Ideally, the map and the agents are updated in parallel every cycle, but in the code this is done successively in a random order.

TownSim outputs images of the generated road network (Figure 2), and can also generate road network files compatible with the SUMO traffic simulator. A separate utility[1] imports SUMO files into the Unreal game engine, procedurally creating road and intersection geometry. TownSim is implemented in Python, available at GitHub[2].

3.1 2-Dimensional Landscape

The sandbox is a 2-dimensional pixel map where each pixel is a node. Each node can contain a limited number of agents and is labeled with a set of types. Types include the following: *water, forest, greenfield, building, park, major road, minor road, bridge*, and *bypass*. Most of the types are mutually exclusive, except for bridge and bypass which are also major road. Water nodes yield the *water* resource, while forest, greenfield, and building yield the *generic* resource. In future versions of the algorithm, we would like to include a city-authority entity which would enforce policies for city planning—this was not implemented in the current version of TownSim. To support the possibility of city planning activities, we envisioned letting *forest* types to become *greenfield* due to agent activity, allowing building only on *greenfield*, as well as an additional *brownfield* type (not in the previous list) that would be created when built node types (*building, major road, minor road, bridge, bypass*, and *park*) are destroyed. This interactivity, like the city-planning policy enforcer, was not implemented, and the only difference between *forest* and *greenfield* right now is that *forest* yields a little more generic resource than *greenfield*.

The decision to distinguish a *park* type is due to small, leftover bits of undeveloped, enclosed gaps in the middle of generated towns. The small gaps are too close to other roads for a new road to form, and don't accumulate enough activity to meet the minimum to become a building type. The real-life analogy would be underdeveloped areas where rent is cheap and traffic is sparse. If we leave them as *forest*/*greenfield* types, agents will continue to perform rural-like gathering activities on those nodes, which is undesirable. Therefore, we turn them into park types, which are the most common type of broadly vegetated area in the middle of a city.

Imitating a settlement phase, the map is seeded with a randomly placed patch of water that follows the shape of a low-dimension polynomial and a small stretch of straight road and building area close to the water. The rest of the map is random-uniformly distributed with forest and greenfield types.

In each time step, first, all agents update (details in the next subsection); then, all map nodes update. The updates are implemented sequentially, re-randomizing the order each time. If prosperity and traffic values are above certain thresholds, the type of the node can change: forest and greenfield types, which we call "unbuilt" types, can become road, building, or park; building type nodes can become road. By default, major roads can be motivated by relatively low prosperity, while creating minor roads, bridges crossing water, and new lots for major roads that want to be created outside of existing lots, require relatively higher prosperity. Bypass-type roads are motivated by traffic. The thresholds for when the building of various road types triggered are controllable by exposed parameters in the software, which we discuss later.

Lastly, prosperity and traffic decrease by decay factors, which are also exposed parameters. Figure 3 contains more information about the exposed parameters.

Param	Default	Description
mr	3	The range within which more than one minor road should not be created.
lr	5	The range within which nodes are a "local" group; used to calculate local prosperity and local traffic.
tr	10	The range that an agent can traverse in one movement; more than one major should not be created within this range.
ma	10	Minimum local prosperity for a major road.
mi	400	Minimum local prosperity for a minor road.
br	1000	Minimum local prosperity for a bridge and/or a new lot.
by	2000	Minimum local traffic for a bypass.
bu	400	Minimum local prosperity for a building.
co	5	Amount of correction for the snap-to-grid effect on new roads.
dp	0.75	Decay multiplier for prosperity every cycle.
dt	0.25	Decay multiplier for traffic every cycle.

Figure 3: Parameter labels, default values, and descriptions.

3.2 Agents

In the "settlement phase", 100 agents are scattered into building nodes. For each following time step, each agent undergoes a sequential consume-work-trade-rest routine, a simplified version of basic human activities, although the time scale of the simulation does not follow day-to-day cycles. At the beginning of each time step, each agent **consumes** a fixed amount of water and generic resource. Next, for "**work**", the agent chooses one resource type to gather depending on which stock is low. To gather, the agent randomly selects a node within its explorable range (*tr*) containing the wanted resource and moves to it (for water, a water node, and for the generic resource, a forest, greenfield, or building node).

Water is a non-depletable resource, while generic resources are depletable. For every time step, forest and greenfield nodes start with hard-coded amounts of generic resource, while the amount in building nodes correlates with the current prosperity of the node.

After gathering, the agent randomly selects another agent in the same node, and **trades** if there is a trade profitable to both parties. An agent's valuation of each resource is determined by the ratio of the two resources in its stock (e.g., each water is worth x generic resource, and vice versa). If a trade occurs, the average between the agents' valuations is used as the price. Finally, the last action

[1] https://github.com/AugmentedDesignLab/Sumo2Unreal
[2] TownSim: https://github.com/AugmentedDesignLab/town-sim-py. Generated data: https://github.com/AugmentedDesignLab/town-sim-data.

that an agent performs in a time step is **rest**: moving to a vacant building node within their explorable range.

As a result of each agent movement, the prosperity of the destination node is increased, and the traffic of the midpoint node in the path is increased. An agent fails to survive if it cannot rest at the end of the cycle, or if either of their resources become negative. Instead of implementing a social system enabling reproduction and immigration, as a simplification, an agent with sufficient resources in a node with extra room can spontaneously spawn a new agent with half of its own resources.

3.3 Road Logic

We define *local prosperity* as the sum of the prosperity of all the neighboring nodes in the local range of a node (*lr*). If local prosperity of a node reaches a minimum threshold, the program runs other checks to see if a road will be created. A minor road, the kind that might be alleyways or residential, must be in an existing lot and near existing building nodes. If the local prosperity of a node outside existing lots is quite high (*br*), a major road can be built to it and a new lot created around it. The part of a major road that crosses over a body of water has the bridge type as well as the major road type, and the creation of this bridge requires higher local prosperity; as in reality, bridges are more expensive.

New road nodes that can be shifted *co* units to make the resulting road exactly vertical or horizontal, would be shifted, creating a weak "snap-to-grid" effect. Although the rest of the simulation, as implemented, tries to imitate unplanned city growth, the snap-to-grid feature very loosely imitates real-life city planning. Real-life city planners, especially in the United States, have been observed to favor grid-like cities. Circular city designs are more common in Europe, where original city centers were built on circular roman campsites. In cities which are so old that their beginnings are before planned development, city centers reflect much more organic patterns [4]. After the simulation makes the decision that a road would be created, major roads are extended toward the edge of the lot, expanding access to previously unexplored areas; minor roads are extended a small amount to reach another street in the other direction, essentially favoring through streets over culs-de-sac.

4 EXPRESSIVE RANGE ANALYSIS

Figure 3 lists and describes the exposed parameters. Default values are a set of parameter settings that reliably produced reasonable outputs in development. We manually selected 4 values spread around the defaults of each parameter to test. For each parameter/test value, we generated 30 100×100 maps using that parameter/test value pair and default values for remaining parameters. Each map was generated over 750 time steps.

4.1 Qualitative Analysis

The outputs of TownSim can be characterized as a range within "semi-gridded". Some generated maps contain more regular blocks, and others have more organic patterns while retaining enough structure to not become quite "dendritic". This variation also occurs within a single map.

In Figure 4, we identify 5 parameters that show interesting visual variations as their values are varied. **The minor road range parameter *(mr)* results in sparser minor roads as it increases**, because this parameter essentially controls the size of the conflict box for minor roads. **Local prosperity/traffic range *(lr)* results in an increased range of density as it increases**. While the road density of the densest generations remains similar, the sparsity of the sparsest generations increases as *lr* increases. There is also varied road density within a single map, especially visible in the figure (lr=4)-left, for example. We reason that this is because this parameter, which controls the size of the area over which prosperity is counted, essentially gives options to the system for where roads should emerge. **Agent traversal range *(tr)* results in greater sprawl as it increases.** Limiting the exploration range stifles growth of the road network. We suspect that this effect would wear off for even higher values of *tr*, although we didn't generate higher values for this data set. In tr=10, two of the 30 simulations halted prematurely after unfortunately losing all agents after a few initial road segments. **Minimum local prosperity for a bridge *(br)* decreases the likelihood of bridges forming as it increases.** This is an expected effect since *br* controls the amount of prosperity needed to incentivize a bridge. By the time *br* is increased to 5000, the roads appear to actively avoid crossing water. **The snap-to-grid correction factor *(co)* increases the prevalence of grid-like sections of roads on the map.** The effect is evident as *co* is increased from 1 to 3 to 5, and seemed to level off at a value of 9.

The ability to control some characteristics of the generated road networks through varying parameter values shows that our rules for agent activity and road generation are good models of road-user activity and road generation at an abstract level. We now examine if the interesting qualities of these parameters are reflected in quantitative metrics, and if the effects of other parameters were overlooked visually, but show up in quantitative metrics.

4.2 Methods of Quantitative Analysis

We use three visualization methods to characterize the sensitivity of the output to different parameter settings. Each method attempts to reflect one important characteristic for road networks: directionality, connectivity, and curvature, respectively. While geographic information systems (GIS) uses data analysis techniques, they have not been carried over to research in procedurally generated road networks. We suggest that the use of quantitative analysis is a contribution of this paper.

4.2.1 Radial Plot. We use radial plots (Figure 5, Figure 6) to visualize the general directionality of generated roads under a parameter setting. The spokes in the plot show the cumulative lengths of the roads running in each direction in a set of generated towns. All generated road segments are straight; curved roads are simply straight segments joined together. The roads are bi-directional, so directions are north-south, east-west, and so on. We create radial plots for minor road types and major road types separately in order to isolate individual patterns or lack of patterns, since they are generated from related but different groups of rules. To highlight patterns, we also plot a subset of that information using only primary (90° and 180°) and secondary (45° and 135°) directions in bar graphs.

Figure 4: For each parameter setting of parameters *mr, lr, tr, br,* and *co,* 30 instances of a 100x100 map are generated. From each group of 30 maps, we chose two visually representative examples to be shown in this figure. Green areas are *greenfield* and *forest* types; blue areas are *water*; beige areas are *building*; black nodes are *major road*; gray nodes are *minor road.*

4.2.2 Connectivity Histogram. The connectivity of roads is often analyzed in various ways in GIS research. We adopt three related measures of connectivity from space syntax [3] using road junctions as base units. We define a junction to include intersections and the ends of culs-de-sac.

$$\sum_{s=1}^{m} s \times N_s = \begin{cases} \textit{simple connectivity} & \text{if } m = 1 \\ \textit{local depth} & \text{if } m = k \\ \textit{global depth} & \text{if } m = l \end{cases} \quad (1)$$

where s is the shortest distance between j and the connected junction, N_s is the number of connected junctions with shortest distance s, l is the maximum shortest distance, and $1 < k < l$. We use $k = 3$ following [3]. Simple connectivity measures the number of immediately neighboring road junctions. Local depth and global depth measure the depth of the connectivity graph on different scales.

We find that simple connectivity and local depth do not change significantly across different settings, hence we omit those figures in this paper. For each parameter setting, we plot the global depth values of all nodes in a histogram, which is useful in showing the shapes of interesting distributions (Figure 7).

4.2.3 Stacked Gradient Chart. We are interested in showing the distributions of different road-curvatures in generated maps. Naively composing spatial curvature maps would have variations cancel out by averaging. We attempt to visualize distributions of curvature using stacked gradients. We first create a gradient map for each generated road network. Road-lengths of different curvatures, represented by color, are sorted and stacked in a gradient bar (Figure 8). We normalize ±2 standard deviations of all of our curvature data to 100% of the colored portion of the gradient, reserving gray for positive outliers. Straight roads with 0 curvature are violet.

4.3 Evaluating road angles

Figure 5 shows that most minor roads run in primary directions, while another significant portion of minor roads run in secondary directions. Minor road range *(mr)* appears to have a weak negative correlation with minor roads in all directions. The summary bar graph reveals that there is a strong negative correlation in secondary directions, which is dampened by the absence of a correlation in primary directions. Local prosperity/traffic range *(lr)* appears to have little effect on minor roads, although the summary bar graph indicates a weak positive correlation in secondary directions. Agent

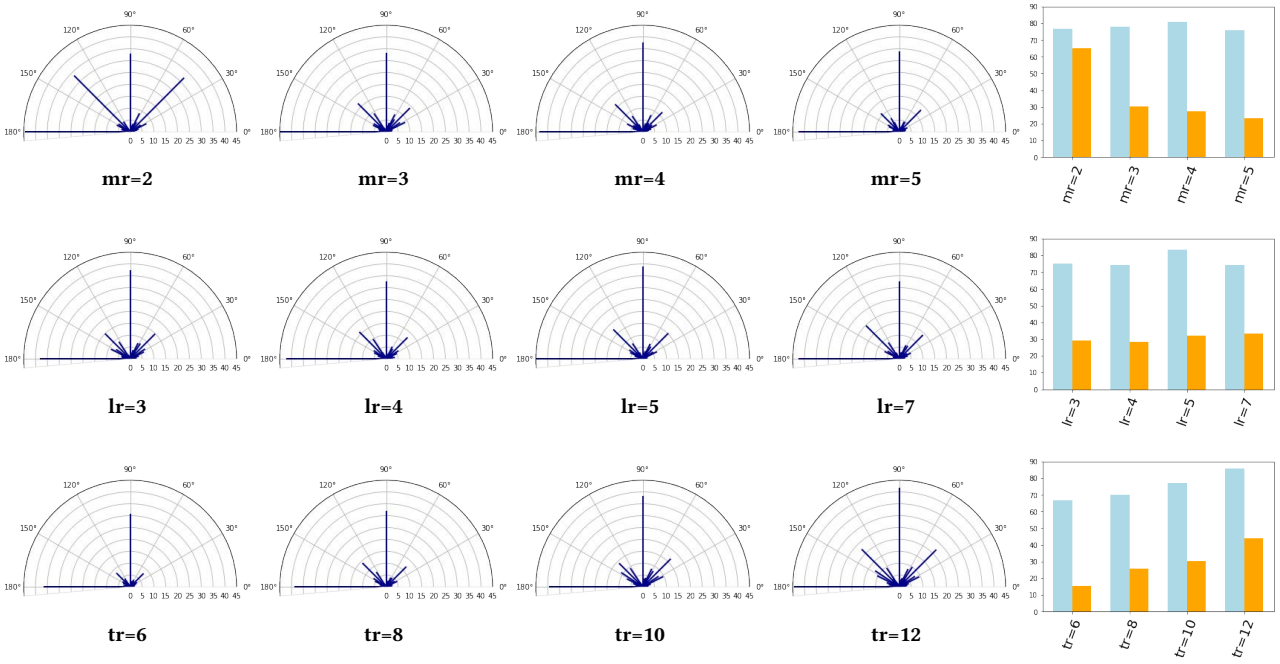

Figure 5: Road angles for minor roads in 30 runs of each setting..

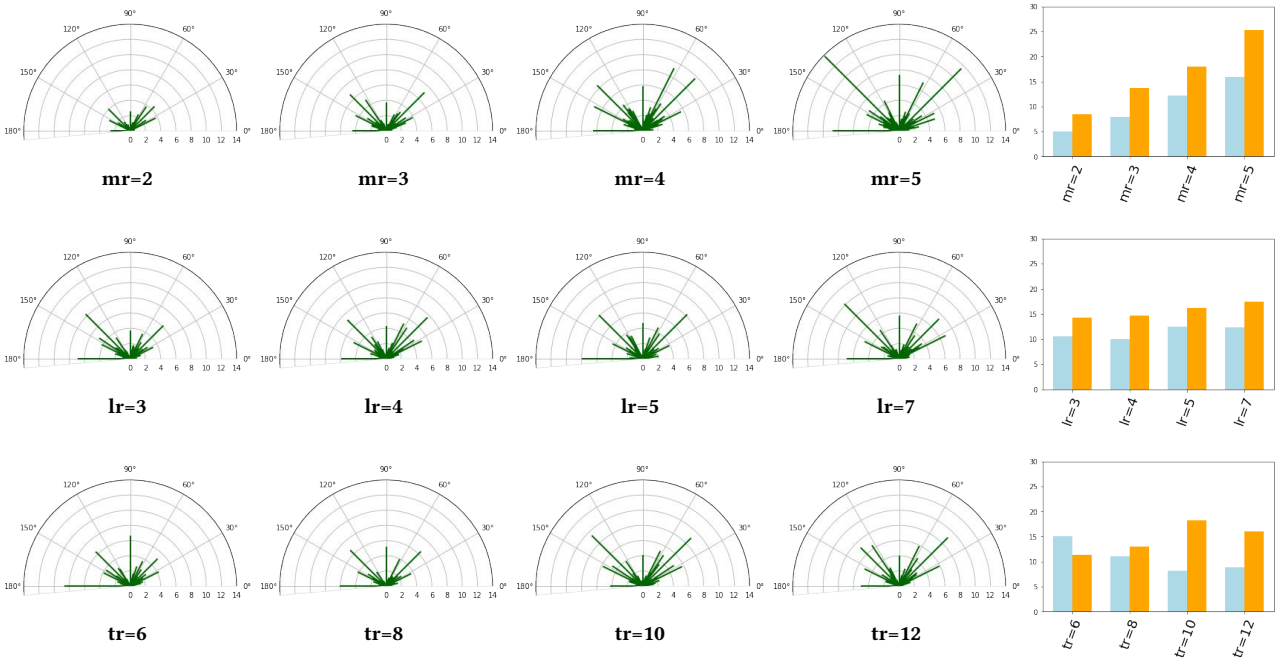

Figure 6: Road angles for major roads in 30 runs of each setting.

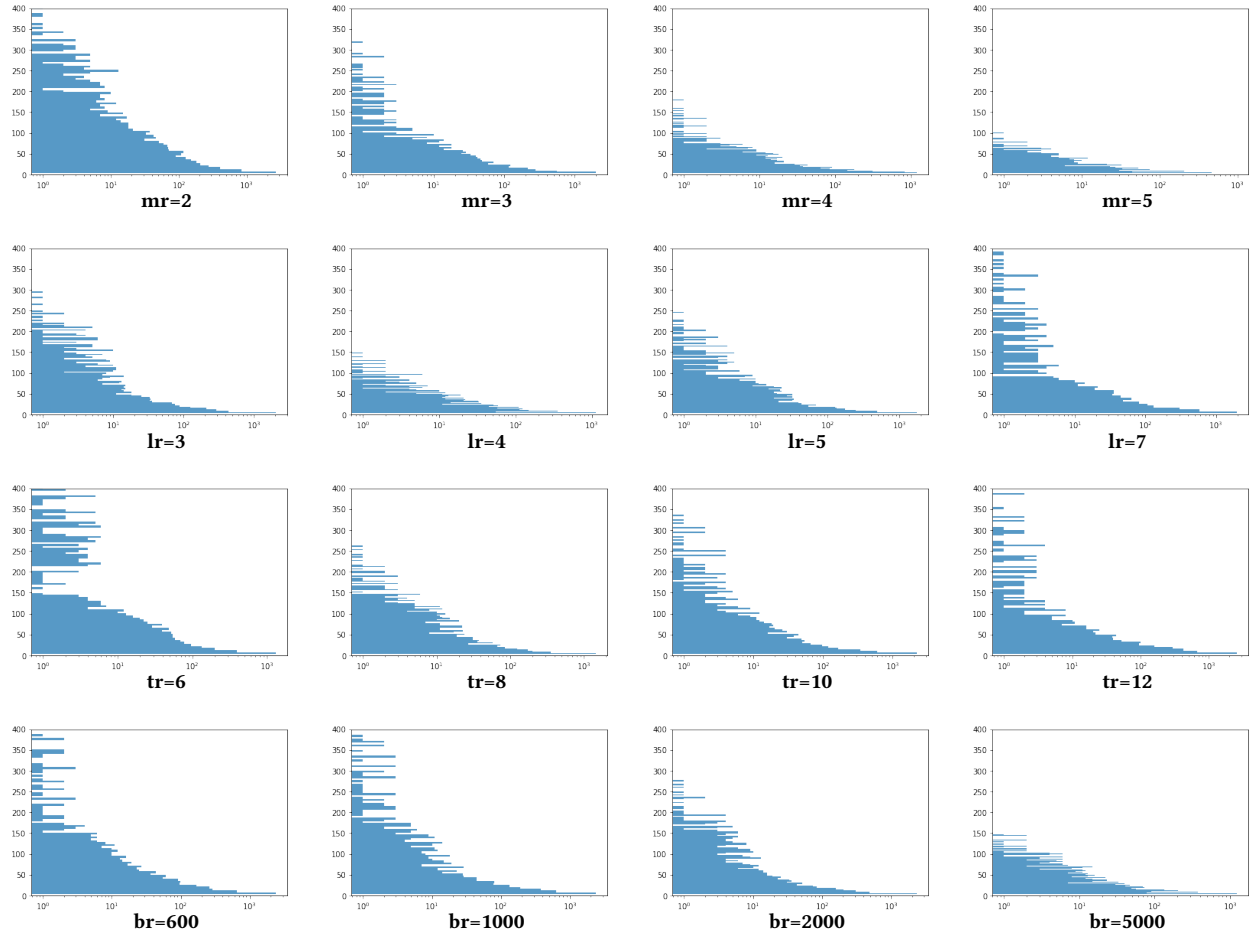

Figure 7: Global depth is on the y-axis and node count is on the x-axis (log scale). Each histogram contains information from 30 maps generated for that parameter setting.

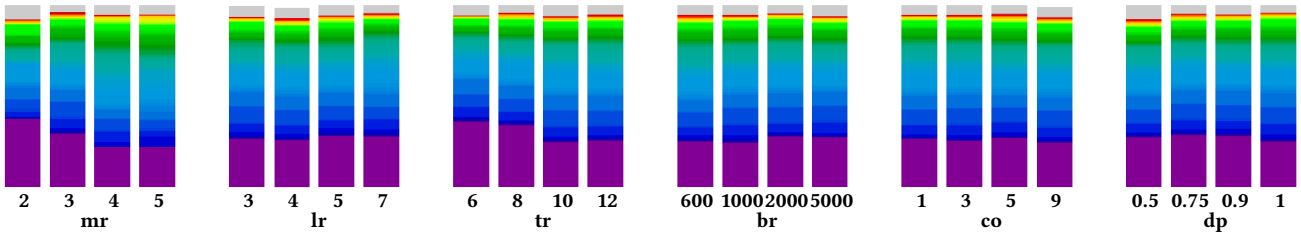

Figure 8: Stacked Gradient Charts. Each bar represents 30 maps generation for that parameter setting. Colors from violet to red represent ±2 standard deviations of mean curvature, where violet is the minimum and light gray represents outlier values.

traversal range *(tr)* has a clear positive correlation with minor roads in all directions. The summary bar graph shows positive correlation in both primary and secondary directions.

Figure 6 shows that a significant portion of major roads in secondary directions, but the distribution of directions is not as skewed toward primary/secondary directions as with minor roads. Minor road range *(mr)* has a clear positive correlation with major roads in all directions. Both primary and secondary direction major roads clearly trend upwards in the summary bar graph. *lr* again shows no pattern in the progression of radial plots, although the summary bar graph shows slight upward trends for both primary and secondary directions. *tr* also shows no apparent pattern in the radial plots, which is confirmed by the summary bar graph.

param	corr. median GD	corr. max GD
mr	-0.86 (< 0.01)	-0.84 (< 0.01)
lr	0.48 (< 0.01)	0.66 (< 0.01)
tr	-0.85 (< 0.01)	-0.59 (< 0.01)
br	-0.13 (0.16)	-0.70 (< 0.01)

Figure 9: Pearson correlation of parameters to median and max global depth on 30 data points each. p-values are in parentheses.

We expected some level of inverse correlation between the distribution of minor and major roads, since both types of roads occupy the same space. However, Figure 5 and Figure 6 did not reflect such an inverse correlation. The relative prevalence of the primary directions with minor roads adds some confusion; however, this is because snap-to-grid works more often for minor roads, which are generally shorter. Trends were not found with other parameters.

4.4 Evaluating node connectivity

Earlier, we defined global depth as a measurement of node connectivity. In Figure 7, the y-axis is the global depth value, and the x-axis is the corresponding count of nodes in log-scale. These global depth values were calculated from Equation 1, where a node connected deeper down is worth more in the sum, so that a high global depth indicates a deep connectivity graph. Figure 9 shows correlation values between parameters and median/maximum global depth.

Minor road range *(mr)* results in lower global depth values as it increases, and especially the instances of higher global depth in the hundreds disappeared in the higher settings. Together with radial plots, the histograms show that although the is only a small decline in the cumulative length of minor roads as *mr* increases, many short minor roads have been replaced with fewer, longer minor roads (it is harder to attribute the drop in global depth to major roads because there is a higher total length of them as *mr* increases). Now we have a better picture of a pattern visually identified in 4.1.

The histograms don't show a trend when varying *lr* (local prosperity/traffic range). For agent traversal range *(tr)*, interestingly, the low end has a relatively large number of nodes with higher global depth in the hundreds, which is unexpected because this is the setting where the road network's growth appeared trapped in the initial block and the road density was not strikingly different from the next higher parameter setting. This is new information that was not identified by simply looking at the map images. The next few histograms [tr=8, 10, 12] show an increasing spread of connectivity — less nodes with low global depth, but more nodes with high global depth.

Minimum local prosperity for a bridge *(br)* results in lower global depth values as it increases, losing the long tail of deep global depth in the hundreds at br=5000.

4.5 Evaluating distribution of curvature

Minor road range *(mr)* and agent traversal range *(tr)* result in evidently higher curvature as they increase, with the violet sections representing 0 curvature dropping around 25% and 15%, respectively, between the lowest and highest settings. Local prosperity/traffic range *(lr)* results in a slight decrease of curvature as it increases; a subtle but likely real effect since it's corroborated by the radial plots and bar graph in 4.3. Bridge min prosperity *(br)* and snap-to-grid correction *(co)*, also parameters of interest identified by the qualitative discussion in 4.1, don't show much variation of curvature across settings. An interesting discovery is that as the parameter *dp*, which is the decay multiplier for prosperity, is increased, the percentages of curvature at both extremes decrease — there are slightly fewer roads with 0 curvature and high curvature, and slightly more roads with middling curvature values.

4.6 Limitations and Future Work

The simulation uses environment and agent behavior that is many times simplified from the real-life counterparts for computability. Although such a simulation would necessarily be a simplification, we hope to include more nuance in a future version of this project. Expressive range analysis is traditionally used to expose emergent properties of interacting parameters on more than one axis; this is lacking in our analysis due to space constraints. Moreover, the final evaluation of a generative system must be to observe its use by real users; this too was lacking in this report. We expect that some feedback will be gathered from autonomous vehicle testing engineers using our generator regarding controllability, qualitative similarity to/interchangeability with naturally-occurring road maps, and gaps in the expressive range.

5 CONCLUSION

In this paper we describe an agent-based city generation algorithm outputting evolved road networks, and explore approaches for expressive range analysis for such generated road networks. There are benefits and drawbacks for each of the approaches. Using qualitative analysis, we looked at each group of road networks generated using a particular parameter setting, and tried to visually parse for unifying characteristics that could be attributable to that parameter setting. This gave a strong sense of the range of road networks generated in response to each parameter and was a very useful point of departure from which to explore other analyses.

We looked at the directionality of generated roads using radial plots, which showed some relative trends between roads of different types and directions, but lost information about the length and number of individual road segments. The radial plots would reveal any impact the parameter settings have on the relative directionality of generated road networks; they showed us that there was limited impact. Network connectivity histograms show some trends of how well-connected typical generated road networks are under different parameter settings. This metric also acts as a proxy for showing the density of roads and magnifies the effect of changes in road density. Lastly, we used stacked gradient charts to look at the curvature of generated road networks, and they revealed a lack of expressivity in the simulation: a few parameters have a limited broad-stroke effect on curvature, but otherwise it is not possible to control the curvature distribution of generated road networks.

ACKNOWLEDGMENTS

This work was funded by a University Research Program gift from the Ford Corporation. We would like to thank Ashley Micks and Anjali Krishnamachar for their guidance on this project.

REFERENCES

[1] 2019. Cities: 7 Ways to Die Wiki. https://7daystodie.gamepedia.com/City. Online; accessed 28 March 2019.

[2] Matt Ball. 2017. A 3D Model Provides the Vision to Combat Blight. https://www.esri.com/about/newsroom/blog/procedural-modeling-provides-vision/. ESRI Blog; accessed 28 March 2019.

[3] Michael Batty. 2004. A new theory of space syntax. (2004).

[4] Michael Batty. 2007. *Cities and complexity: understanding cities with cellular automata, agent-based models, and fractals*. The MIT press.

[5] Carlos Campos, João Miguel Leitão, Joao Paulo Pereira, António Ribas, and António Fernando Coelho. 2015. Procedural generation of topologic road networks for driving simulation. In *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 1–6.

[6] Guoning Chen, Gregory Esch, Peter Wonka, Pascal Müller, and Eugene Zhang. 2008. Interactive procedural street modeling. In *ACM transactions on graphics (TOG)*, Vol. 27. ACM, 103.

[7] Matthew DeBord. 2018. Waymo just crossed 10 million self-driving miles – but the company has a secret weapon that gives it even more of an edge. https://www.businessinsider.com/waymo-self-driving-cars-secret-weapon-is-simulation-testing-2018-10. Web page, accessed 17 April 2019.

[8] Arnaud Emilien, Adrien Bernhardt, Adrien Peytavie, Marie-Paule Cani, and Eric Galin. 2012. Procedural generation of villages on arbitrary terrains. *The Visual Computer* 28, 6-8 (2012), 809–818.

[9] Eric Galin, Adrien Peytavie, Eric Guérin, and Bedřich Beneš. 2011. Authoring hierarchical road networks. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 2021–2030.

[10] Eric Galin, Adrien Peytavie, Nicolas Maréchal, and Eric Guérin. 2010. Procedural generation of roads. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 429–438.

[11] Stefan Hartmann, Michael Weinmann, Raoul Wessel, and Reinhard Klein. 2017. StreetGAN: towards road network synthesis with generative adversarial networks. (2017).

[12] Sophia Karagiorgou, Dieter Pfoser, and Dimitrios Skoutas. 2013. Segmentation-based road network construction. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 460–463.

[13] Sophia Karagiorgou, Dieter Pfoser, and Dimitrios Skoutas. 2017. A layered approach for more robust generation of road network maps from vehicle tracking data. *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 3, 1 (2017), 3.

[14] George Kelly and Hugh McCabe. 2006. A survey of procedural techniques for city generation. *The ITB Journal* 7, 2 (2006), 5.

[15] George Kelly and Hugh McCabe. 2007. Citygen: An interactive system for procedural city generation. In *Fifth International Conference on Game Design and Technology*. 8–16.

[16] BaekGyu Kim, Akshay Jarandikar, Jonathan Shum, Shinichi Shiraishi, and Masahiro Yamaura. 2016. The SMT-based automatic road network generation in vehicle simulation environment. In *2016 International Conference on Embedded Software (EMSOFT)*. IEEE, 1–10.

[17] Joon-Seok Kim, Hamdi Kavak, and Andrew Crooks. 2018. Procedural city generation beyond game development. *SIGSPATIAL Special* 10, 2 (2018), 34–41.

[18] Devin Lavigne. 2016. GeoDesign, CityEngine, and Urban Planning. http://proceedings.esri.com/library/userconf/geodesign16/papers/geo_05.pdf. In *2016 Geodesign Summit*.

[19] Tom Lechner, Benjamin Watson, Uri Wilensky, Seth Tisue, Martin Felsen, Andy Moddrell, Pin Ren, and Craig Brozefsky. 2007. *Procedural modeling of urban land use*. Technical Report. North Carolina State University. Dept. of Computer Science.

[20] Markus Lipp, Daniel Scherzer, Peter Wonka, and Michael Wimmer. 2011. Interactive modeling of city layouts using layers of procedural content. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 345–354.

[21] Alexis C. Madrigal. [n. d.]. Inside Waymo's Secret World for Training Self-Driving Cars. https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/, note=Web page, accessed 17 April 2019, year = 2018.

[22] Gen Nishida, Ignacio Garcia-Dorado, and Daniel G Aliaga. 2016. Example-Driven Procedural Urban Roads. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 5–17.

[23] Yoav IH Parish and Pascal Müller. 2001. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 301–308.

[24] Otger Roglà Pujalt, Núria Pelechano Gómez, and Gustavo Ariel Patow. 2017. Procedural semantic cities. In *CEIG 2017: XXVII Spanish Computer Graphics Conference: Sevilla, Spain, June 28-30, 2017*. European Association for Computer Graphics (Eurographics), 113–120.

[25] David Santiago. 2015. Procedural and Automation Techniques for Design and Production of Sunset Overdrive. https://www.gdcvault.com/play/1022217/Procedural-and-Automation-Techniques-for. Online slide presentation from 2015 Game Developer's Conference; accessed 28 March 2019.

[26] Ruben M Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. 2014. A survey on procedural modelling for virtual worlds. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 31–50.

[27] Ruben Michaël Smelik, Tim Tutenel, Klaas Jan de Kraker, and Rafael Bidarra. 2011. A declarative approach to procedural modeling of virtual worlds. *Computers & Graphics* 35, 2 (2011), 352–363.

[28] Gillian Smith, Jim Whitehead, Michael Mateas, Mike Treanor, Jameka March, and Mee Cha. 2011. Launchpad: A rhythm-based level generator for 2-d platformers. *IEEE Transactions on computational intelligence and AI in games* 3, 1 (2011), 1–16.

[29] Jing Sun, Xiaobo Yu, George Baciu, and Mark Green. 2002. Template-based generation of road networks for virtual city modeling. In *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, 33–40.

[30] Carlos A Vanegas, Daniel G Aliaga, Bedrich Benes, and Paul A Waddell. 2009. Interactive design of urban spaces using geometrical and behavioral modeling. In *ACM transactions on graphics (TOG)*, Vol. 28. ACM, 111.

[31] Carlos A Vanegas, Tom Kelly, Basil Weber, Jan Halatsch, Daniel G Aliaga, and Pascal Müller. 2012. Procedural generation of parcels in urban modeling. In *Computer graphics forum*, Vol. 31. Wiley Online Library, 681–690.

[32] Paul Waddell. 2002. UrbanSim: Modeling urban development for land use, transportation, and environmental planning. *Journal of the American planning association* 68, 3 (2002), 297–314.

[33] Benjamin Williams and Christopher J Headleand. 2017. A Time-Line Approach for the Generation of Simulated Settlements. In *2017 International Conference on Cyberworlds (CW)*. IEEE, 134–141.

[34] Aleksey Yurkin and Max Rybalchenko. 2017. Discover How Frogwares' 'City Generator' is Saving Valuable Time During Development of 'The Sinking City'. https://www.unrealengine.com/en-US/developer-interviews/discover-how-frogwares-city-generator-is-saving-valuable-time-during-development-of-the-sinking-city. Online; accessed 28 March 2019.