Procedural Modelling of Urban Road Networks

Jan Beneš[1] Alexander Wilkie[1] Jaroslav Křivánek[1]
[1]Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic

Abstract

We present a model for growing procedural road networks in and close to cities. The main idea of our paper is that a city cannot be meaningfully simulated without taking its neighbourhood into account. A simple traffic simulation that considers this neighbourhood is then used to grow new major roads and to influence the locations of minor road growth. Waterways are introduced and used to help position the city nuclei on the map. The resulting cities are formed by allowing several smaller settlements to grow together and to form a rich road structure, much like in real world, and require only minimal per-city input, allowing for batch generation.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

Procedural modelling techniques have long been a point of interest in computer graphics. They allow for creation of realistic models of many types of natural environments, such as terrains [MKM89], water-bodies [ŠBBK08], plants [PL96], and clouds [MYDN01]. Typically, these techniques permit the generation of models of these phenomena on a level of detail that would be hard to achieve in terms of both quantity and quality if they were produced by hand. Relatively recently, an additional circumstance has arisen that further increases the use of procedural techniques in computer graphics. With the rising levels of quality expected from entertainment productions, the demand for high-quality graphical content of all kinds has sharply risen, and will continue to do so. As a consequence, to compensate for both the rising costs for graphical content production and for the lack of skilled artists, the video-game industry, movie studios, TV productions, and others are increasingly employing procedural modelling.

Furthermore, cities are, in a sense, at the core of modern societies. It is therefore not surprising that many movies and video-games involve various real or fictional cities in their story-lines, which creates an increasing demand for city-related graphical content.

We present a model for growing procedural road networks in urban areas that has two main goals. First, the current methods for cities require various maps [PM01, VABW09] or a growth centre distribution and are more interaction oriented [WMWG09, VABW09]. We, on the other hand, wish to develop a method that has minimal required *per-city* input and only has a small, well defined set of user parameters—a "*historical context*"—that can be reused by different cities that share the same traits. Unlike existing methods, such a method could then be used by end-user applications that require the generation of city assets at run-time. Second, we want to develop a method that can simulate the development of cities in an urban landscape, that is in the neighbourhood of other cities and smaller settlements, over time. This would allow for a simulation that is based more on first-principles in which smaller settlements grow together to form larger settlements and for a richer major road structure. As we show in the results section, our method allows for both these goals.

There are several topics that we consider to be outside of the scope of this paper. First, we only generate the road network of a city, but no buildings or other geometry. Those

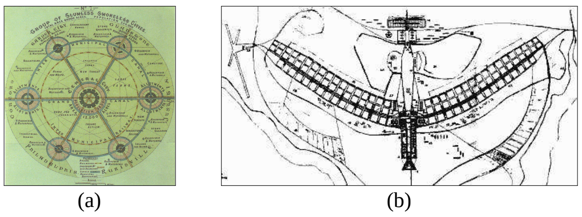

Figure 1: (a) The proposed structure of a garden city [Bee02]. (b) The plan of Brasilia, the capital of Brazil - inscribed in the outline of a bird with spread wings [Cos57].

can be generated by e.g. [MWH*06, GPMG10]. We also do not attempt to replicate any cultural influence on roads patterns, such as those seen in Figure 1, as we consider them to be a non-simulatable artifact that could perhaps be replicated on a per-culture basis, but is unlikely to emerge from any simulation.

2. Previous Work

The 2001 paper by Parish and Müller [PM01] is probably the first published effort to generate realistic cities for use in computer graphics. It uses a hierarchical, L-system-like method to generate both major and minor roads that follow one of the predefined road pattern styles (grid, radial, least-elevation) over a given population density and road pattern map. In Weber et al. [WMWG09], this approach was augmented with a custom traffic simulation and extended by a highly configurable land use simulation generating cities that evolve over time. In another work, Vanegas et al. [VABW09] proposed an interactive city design method based on geometrical and behavioural modelling that also generates a road network and supports grid and radial patterns. An early attempt at agent based modelling of cities is described in Lechner et al. [LWW03] and an interesting method where minor and major roads are traced in a sketched tensor field is given in Chen et al. [CEW*08]. Recently, Yang et al. [YWVW13] proposed a new hybrid streamline tracing and template matching method for generating high-quality, neighbourhood scale urban layouts by means of hierarchical domain splitting, aimed primarily at urban design. Also, methods for example-based synthesis of road networks [AVB08, YS12], and for interpolation of given historical urban road networks [KMK12] have been suggested. To the best of our knowledge, Groenewegen [GSdKB09] is the first author to point out the differences between various land use models, using two of them in determining land use for Western European and North American cities. In 2012, Emilien et al [EBP*12] described a method for generating various types of villages. Of all these methods, only [WMWG09,KMK12] and perhaps [EBP*12] model the *evolution* of a city over time, while the other generate cities and their respective road networks at a *fixed* point in time. Recently, Vanegas et al. [VGDA*12] proposed a method for "inverse design" of cities, an intuitive high-level method of controlling existing urban procedural models. Finally, two works by Galin et al. concerning the generation of roads over terrain have been published. The first one [GPMG10] proposes a method for generating single roads and their geometry over terrains using a discretization scheme, while the latter [GPGB11] extends it to allow for generation of plausible major road networks that connect (unsimulated) cities and other settlements. For completeness, a comparison of the main features of closely related methods is given in Table 1 of the supplementary materials.

Our contribution to previous work is: 1) Water transportation and its implications on settlement positioning, 2) Simulation that accounts for neighboring cities, 3) having settlements grow together to form an urban landscape, 4) changing minor road patterns with time, and 5) minimal per-city input. A thorough discussion of our contribution, its novelty, and its implications can be found in Sections 4 and 4.1.

3. Our Method

In accordance with previous work, we call the results of our algorithm *cities* and also understand the word city the way it is commonly used. However, the notion of a city is vague, as discussed by e.g. Frey [FZ01], and we therefore sometimes use the word *settlement* to describe a populated place, perhaps connected to other settlements by roads. When we want to emphasize that a city is the result of several settlements growing together, we might refer to it as a *conurbation*. We also use the phrase *urban landscape* to denote all the settlements that are the result of our method, stressing that there are possibly many of them and of various sizes. For convenience, a list of used symbols is provided in the supplementary materials.

3.1. Overview

At a coarse level of detail, as can be seen in Algorithm 1, our method consists of an initialization phase and a main loop. In the initialization phase, we use the positions of the neighbouring cities (as mentioned in Section 1) to generate an initial road network on which our city will grow. In the main loop, we then let the city develop by building new minor roads, simulating traffic, and building new major roads, in that order, repeating for as long as desired.

Algorithm 1: High-level pseudo-code of our algorithm.

```
INITIALIZATION();                // see Section 3.2
while $t < t_{max}$ do
    GROWMINORROADS();            // see Section 3.3
    TRAFFICSIMULATION();         // see Section 3.4.2
    GROWMAJORROADS();            // see Section 3.4.2
    INCREASE($t$);
end
```

More precisely, our algorithm takes as its input 1) a terrain elevation map $\boldsymbol{m}$, classified into land, water, and forbidden areas, 2) a list of neighbouring cities $\boldsymbol{nc}$, which we do not simulate except for their contributions to the traffic simulation, 3) a set of control functions $\boldsymbol{t}(t), \boldsymbol{G}(t), \boldsymbol{R}(t)$ that determine the level of traffic, rate of city growth, and major road construction parameters with respect to simulation time t, 4) a time-dependent sequence of minor road pattern definitions $\boldsymbol{K}(t)$, and 5) time-independent coefficients $\boldsymbol{O}_1, \dots, \boldsymbol{O}_4$ that will be explained later. While the terrain $\boldsymbol{m}$, the traffic function $\boldsymbol{t}(t)$, and the list of neighbouring cities $\boldsymbol{nc}$ are required for each new city (lower-case bold font), we provide a way of generating them automatically, see Section 3.5. The growth and major road construction parameters functions $\boldsymbol{G}(t)$ and $\boldsymbol{R}(t)$, coefficients $\boldsymbol{O}_1, \dots, \boldsymbol{O}_4$, and the sequence of minor road patterns $\boldsymbol{K}(t)$ form a "*historical context*" (upper-case bold font)

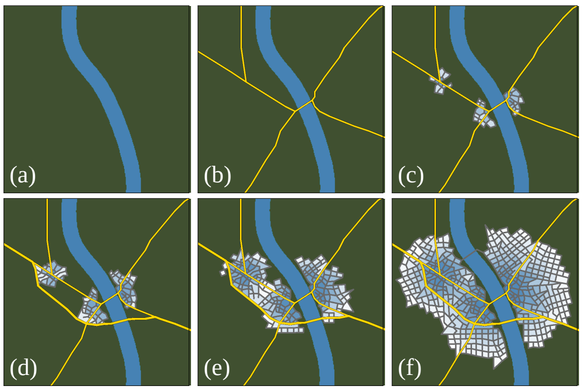

Figure 2: A typical progression of our algorithm. On an empty map (a), an initial road network (b) is grown and nuclei are identified. Initially, the city consists of several smaller settlements (c). As it grows and traffic increases, new major roads (yellow) are built (d) and a conurbation is formed (e). Eventually, even new major roads are absorbed (f).

which several different cities can share. With the historical context and per-city input, we proceed as follows:

1. **Initialization.** One by one, the algorithm builds major roads that connect pairs of neighbouring cities in $\boldsymbol{nc}$, resulting in an initial road network (see Figure 2b). In this initial road network, we find all intersections and use them as the initial nuclei of city growth.
2. **Growing minor roads.** With the positions of nuclei of city growth in mind, we select positions out of which new minor roads can grow. We then grow these new minor roads using the appropriate minor road pattern (“street pattern”) $\boldsymbol{K}(t)$.
3. **Traffic simulation.** For each pair of neighbouring cities, we keep track of the total traffic between them. This traffic can flow across one or more different trade routes, which are paths running entirely over major roads. If the available traffic capacity between a pair of neighbouring cities is greater or equal to the increase $\boldsymbol{t}(t) - \boldsymbol{t}(t-1)$ in traffic, no new major road is required and we only increase the traffic. Otherwise, a new major road might have to be created.
4. **Growing major roads.** If there is not enough available capacity to accommodate the increase in traffic, a new major road might have to be built. We therefore propose 1) a new major road that is subject to the major road construction parameters $\boldsymbol{R}(t)$. This road might consist entirely of new major roads, reuse existing minor and major roads, or both, and 2) as an alternative, we consider re-routing the surplus traffic over already existing major roads. Out of these two options, we choose the better one and add a new trade route that runs along it, thus accommodating for the increase in traffic. Finally, we increase the simulation time t and go back to step 2.

3.2. Initialization and the Initial Road Network

3.2.1. Rationale and Overview

According to Johnson [Joh67], cities can be classified into four generic types based on the reason for their existence: **1) Resource cities.** Generate a resource, for example by mining, crop growing, or fishing. Their position is determined by the location of the resource and they are likely to be directly connected to several other cities, as they need to trade their products. **2) Processing cities.** Handle a resource (including people) by providing services such as raw resource processing, assembly of goods from parts, changing mode of transportation, splitting trade routes, or providing housing for travellers. Thus, they grow on intersections of preexisting routes. **3) Market cities.** Provide local access to an (imported) resource. They are therefore cities with very high accessibility and can be thought of as being located at the intersection of roads connecting several neighbouring cities. **4) Other cities.** Such as fortresses or artificial capitals. Here, the existence of a city is not implied by the road network's structure. However, even these cities eventually develop connections to other settlements or cease to exist.

As explained, virtually all cities are to be thought of as being located on an intersection of trade routes and have a moderate or higher degree of connectivity. For these reasons, our simulation does not start on an empty map. It instead starts by constructing an initial road network. On this road network, roads connect the notional, unsimulated neighbouring cities (see Figure 3a), providing a road network structure on which to start our simulation, mimicking the situation at a very early point in the city's history, with the neighbouring cities already in existence and the road network connecting them in place. This road network contains a small number of intersections that will be the nuclei of the city's future growth.

3.2.2. Technique

We now discuss how the initial road network is constructed. Galin et al. [GPMG10] suggested the use of a pathfinding graph to generate roads over complex terrains. Below, we extend their algorithm so as to allow for paths over land (roads) and *also water*. Our pathfinding graph $\mathcal{G} = (\mathcal{L}, \mathcal{E})$ is defined over a regular lattice $\mathcal{L}$ spaced at $\ell = 20$m, which we classify into three mutually exclusive sets $\mathcal{L}_L$, $\mathcal{L}_W$, and $\mathcal{L}_F$ for lattice vertices over land, water, and forbidden areas respectively (see Figure 3b). First, we construct the land graph $\mathcal{G}_L = (\mathcal{L}_L, \mathcal{E}_L)$, where the edges connect vertices in $\mathcal{L}_L$ for which their distances along both axes of the lattice are less than or equal to a radius r (we use $r = 8$), see Figure 3c. For each (undirected) edge $e \in \mathcal{E}_L$, its weight $\textsc{Weight}(e)$ represents the difficulty of getting from one of its ends to the other. We calculate this weight by marching along the edge and keeping track of the presence of water, forbidden areas and maximum edge slope $E_{L,\max}$. If an edge runs over water, forbidden areas, or has $E_{L,\max}$ greater

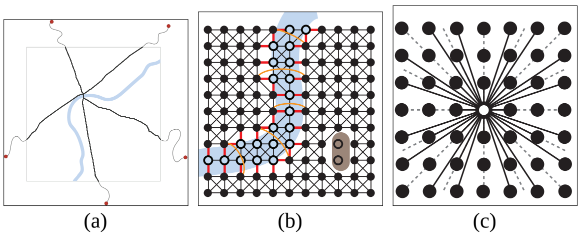

Figure 3: (a) The map $\boldsymbol{m}$ (inner border) with a river (blue), initial road network (thick black), neighbouring cities $\boldsymbol{nc}$ beyond the map's border (red); the gray roads are a schematic, non-simulated continuation of the map's road network. (b) Schematic depiction of the pathfinding graph $\mathcal{G}$, with land vertices $\mathcal{L}_L$ and edges $\mathcal{E}_L$ (black dots), water vertices $\mathcal{L}_W$ (black and blue) and edges $\mathcal{E}_W$ (between vertices $\in \mathcal{L}_W$), forbidden vertices (brown-black) in forbidden area (brown), bridge edges $\mathcal{E}_B$ (orange arcs), and harbour edges $\mathcal{E}_H$ (red). (c) The way edges (black) connect a vertex (white) on the lattice with the nearest other vertex for each direction (dashed line), here shown for a radius of $r = 3$.

than a cut-off constant $E_{L,\text{cut}}$, we discard it. For all non-discarded edges, their weight is computed as $\textsc{Weight}(e) = \textsc{Length}(e) \cdot \max(1, \boldsymbol{O}_1 \cdot E_{L,\max})$, where $\boldsymbol{O}_1$ is a scalar constant. This heuristic works well in practice. We then construct the water graph $\mathcal{G}_W$ in a similar fashion, requiring the edges not to pass over land or forbidden areas. In our implementation, water elevation is constant and no edge slope cut-off constant is used.

As we want the land graph $(\mathcal{L}_L, \mathcal{E}_L)$ and the water graph $(\mathcal{L}_W, \mathcal{E}_W)$ to be connected, we "stitch" them together using harbour edges $\mathcal{E}_H$ and bridge edges $\mathcal{E}_B$. The harbour edges connect adjacent land and water vertices, allowing for a change of transportation mode, whereas the bridge edges (see [GPMG10] for details of bridge edge generation) connect not necessarily adjacent land vertices and have at least one point located over water. For vertices in forbidden areas, no edges are generated. The complete pathfinding graph $\mathcal{G}$ is therefore given as the union of the land component $(\mathcal{L}_L, \mathcal{E}_L)$, the water component $(\mathcal{L}_W, \mathcal{E}_W)$, the harbour edges $\mathcal{E}_H$ and the bridge edges $\mathcal{E}_B$, getting $\mathcal{G} = (\mathcal{L}, \mathcal{E})$, where $\mathcal{L} = \mathcal{L}_L \cup \mathcal{L}_W \cup \mathcal{L}_F$ and $\mathcal{E} = \mathcal{E}_L \cup \mathcal{E}_W \cup \mathcal{E}_H \cup \mathcal{E}_B$.

Next, we describe the construction of the roads that connect neighbouring cities. With each neighbouring city represented by a distinct vertex on the edge of either the water or the land lattice ($\mathcal{L}_L \cup \mathcal{L}_W$; see Figure 3b), we are given (that is we generate, see Section 3.5 for description) an ordered list $\boldsymbol{y'}$ of neighbouring city pairs p_i. Then, for each pair $p_i \in \boldsymbol{y'}$, we use Dijkstra's algorithm to find a minimum weight path $\mathcal{P}_i$ in $\mathcal{G}$ that connects the neighbouring cities in the pair p_i and multiply the edge weights along $\mathcal{P}_i$ by a reuse coefficient $\boldsymbol{u}$ (a constant in $\langle 0, 1 \rangle$ that can be randomly generated at the algorithm's start or optionally set by the user) to stimulate edge reuse by subsequent paths.

Each path $\mathcal{P}_i$ is a sequence of edges in the pathfinding graph $\mathcal{G}$. We, however, represent the built roads in a separate planar graph $G = (J, S)$, where each straight edge $s \in S$ is called a *segment* and the points where two or more segments meet are called *nodes* $j \in J$. Therefore, as a next step, we need to convert each found path $\mathcal{P}_i$ of edges in $\mathcal{G}$ into a sequence $P_{i,1}$ of land, water, bridge, and harbour segments—a *trade route*—where i denotes that the trade route runs between the cities of the neighbouring cities pair p_i and the index 1 means that it is the first of potentially many trade routes for that pair (see Section 3.4). During the conversion from $\mathcal{P}_i$ to $P_{i,1}$, we add new segments and nodes to G where necessary and *reuse* existing segments and nodes where possible, keeping the graph planar. Once the initial road network is built by adding roads between the pairs in the ordered list $\boldsymbol{y'}$ of neighbouring city pairs, we still have to find (as opposed to build) the trade routes $P_{i,1}$ in the segment graph G for all the remaining pairs, i.e., $\boldsymbol{y} \setminus \boldsymbol{y'}$, so that there is a trade route $P_{i,1}$ for every pair of distinct cities in the list of neighbouring cities $\boldsymbol{nc}$.

We continue the initialization step by finding all the nodes in the segment graph G with a degree ≥ 2 and all the nodes from which bridges and harbours emanate. We then add them to a set of *city growth nuclei* Q. The role of nuclei in our algorithm is discussed in more detail in Section 3.3. Finally, we set the current simulation time $t = 0$.

3.3. Growing Minor Roads

In the previous section, we have described the algorithm's initialization step, namely the construction of the pathfinding graph $\mathcal{G}$, the initial road network, and the set of city nuclei Q. We continue by describing how our algorithm grows minor roads, the first step of the simulation's main loop.

3.3.1. Rationale

As can be seen from looking at the maps of various old—especially European—cities (see Figure 4) there is a strong dependency between historical period and the exhibited minor road pattern, although other dependencies, both important (use of land) and less pronounced (majority culture), can be observed. We make the simplifying assumption that time is the only important quantity and, by means of the control function $\boldsymbol{K}(t)$, assign to each simulation time t a specific minor road pattern to be used when growing new minor roads.

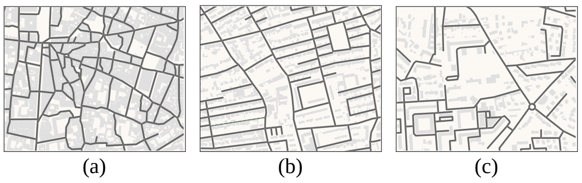

Figure 4: Various minor road patterns found in the historical centre (a) and in newer parts (b), (c) of Avignon, France. Images reproduced from OpenStreetMap and contributors.

3.3.2. Technique

The location of new minor roads is heavily influenced by the position of nuclei of city growth $q \in Q$. These represent places that attract city growth, such as intersections, workplaces, or public amenities (see Section 3.4.1). The stimulus that each nucleus q provides is expressed by $\text{STRENGTH}(q) \in \langle 0,1 \rangle$.

The nodes from which new minor roads may be grown form a set Z. This set consists of nodes $j \in J$ that have not spawned new streets yet. Additionally, for each segment $s \in S$ whose $\text{LENGTH}(s) \geq 2 \cdot \ell$, where ℓ is the spacing of vertices in the pathfinding graph $\mathcal{G}$, we add $\lfloor \text{LENGTH}(s)/(1.25 \cdot \ell) \rfloor$ equidistant temporary nodes along each such segment. To actually grow new minor streets, we order the nodes $z \in Z$ into a list Z' by the value of $\text{RANK}(z) = \text{EUCLIDEANDIST}(q_N, z) \cdot (1 - \text{STRENGTH}(q_N))$, where q_N is the nucleus nearest to the node z. We then select the first $\boldsymbol{G}(t)$ nodes from the ordered list Z' as the nodes from which new minor roads will be grown. In our implementation, we use the algorithm presented in [PM01] and extended in [WMWG09] to generate minor road patterns, but other algorithms, such as [AVB08, YS12], might conceivably be .

To prevent artifacts near map borders, we select the nucleus q_C corresponding to the initial road network intersection with the highest degree as the centre of the city that we are growing, let its $\text{STRENGTH}(q_C) = 1$, and attenuate the strength of other nuclei as they approach the map's borders.

As mentioned in 3.2.2, the first nuclei are added at the intersections of the initial road network. Later, when nuclei of city growth corresponding to intersections of the new major roads are built, additional nuclei are placed at positions randomly selected from Z to account for chance.

3.4. Traffic Simulation and Growing Major Roads

3.4.1. Rationale

For automated major road growing, if any, current city growth simulations, e.g. [WMWG09], usually depend on local traffic simulations. These simulations in turn depend on land use simulations. However, current land use simulations, such as [LRW*06], as well as other popular city models—such as the concentric, sector, and multiple nuclei models [Joh67]—are results of analyses of then-current city growths. In these time contexts, these can be readily applied, as demonstrated by Groenewegen et al. [GSdKB09]. The land use in the concentric model has, for example, been shown to be inverted in pre-industrial [Sjo60] and 20th century cities [Bur08]. We therefore argue that models involving land use simulations are not easily applicable to simulations over longer periods of time or to different cultures [KP06], motivating the development of our method. Further complexity is added by the changing understanding of cities [FZ01], with new urban structures such as megacities, edge cities, or outer cities appearing relatively recently [Soj00].

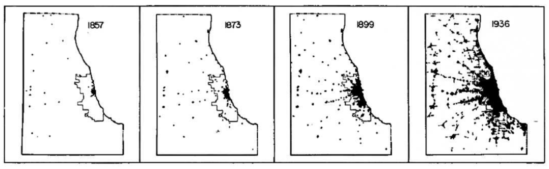

Figure 5: Time progression of settled areas within the Chicago metropolitan region between 1857 and 1936. A tentacle-like sprawl along traffic (railway) routes is clearly apparent. The dot-like pattern reflects the positions of train stations. Reproduced from [Hoy39].

We attempt to overcome these problems by identifying trade and the resulting traffic as the most important, stable and simulatable factor for determining the shape and growth rate of a city across virtually all cultures. In addition to the discussion in Section 3.2.1, the importance of traffic and trading is also addressed by classic literature, e.g. [Chi50]. Next, the importance of transportation route configuration on city shape is treated by Hoyt [Hoy39], see Figure 5, who shows that city growth is related to accessibility and therefore, the presence of transportation routes. Similar observations in the context of highways have later been made for example in [TT95, Gar92]. Overall, accessibility attracts growth and increases near intersections. We therefore use a simple traffic simulation between the neighbouring cities to simulate the need for new major roads, which we use to drive the shape of our cities by placing additional growth nuclei at the newly created major road intersections and nearby (see Section 3.3.2).

3.4.2. Technique

When building new major roads at simulation time t, we optimize their length and cost with respect to the major road construction parameters $\boldsymbol{R}\colon (t,\alpha) \to \mathbb{R}^8$, an umbrella function mentioned above. For each (t,α), the user-defined function $\boldsymbol{R}(t) = (\boldsymbol{P}(t,\alpha), \boldsymbol{A}(t), \boldsymbol{C}_1(t), \ldots, \boldsymbol{C}_6(t))$ gives 1) a function $\boldsymbol{P}(t,\alpha)$, see Figure 7a, that relates angle α between consecutive segments to a cost penalty, 2) a tabulated function $\boldsymbol{A}(t)$

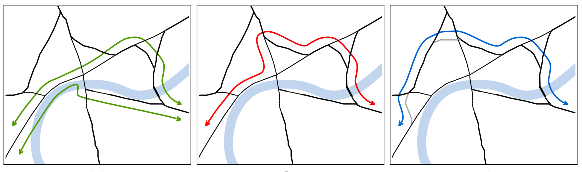

Figure 6: Traffic simulation overview. (a) There can be more than one trade route (green) between two neighbouring cities. When they cannot accommodate the increase in traffic between the cities, an alternative trade route (red) over existing major roads only (b) and a trade route (blue) that can build new roads (gray) are proposed (c) and the better one is built. Only major roads (black) are shown.

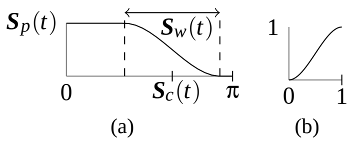

Figure 7: (a) The turn angle penalty function $\boldsymbol{P}(t,\alpha)$ at a fixed time t is a scaled version of the function SMOOTHSTEP, where $\boldsymbol{S}_c(t)$, $\boldsymbol{S}_w(t)$, and $\boldsymbol{S}_p(t)$ are the centre, width, and height of the step function. The angle α between two consecutive segments is on the x-axis, ranging from 0 (turn back) to π (no turn). (b) The function $\text{SMOOTHSTEP}(x) = 3x^2 - 2x^3$ [CR08].

that defines the recommended traffic capacity for newly built major roads, and 3) six cost coefficients $\boldsymbol{C}_1(t),\ldots,\boldsymbol{C}_6(t)$, which will be detailed later and which influence edge costs when new major roads are being proposed. Using $\boldsymbol{R}(t)$, we can control the shape of new major roads, allowing historical roads to be more curvy and to reuse existing minor roads more easily, while forcing newer roads to be smoother and stay clear of built-up areas.

For the purposes of the traffic simulation, we keep a list of trade routes $P_i = (P_{i,1},\ldots,P_{i,|P_i|})$ for each pair $p_i \in \boldsymbol{y}$ of neighbouring cities. The first trade route $P_{i,1}$ for each pair has been described and found in Section 3.2.2. We denote the traffic over a trade route $P_{i,j}$ as $T(P_{i,j})$ and always initially set it to 0. Since a trade route is basically a list of segments in the segment graph G, each segment can be shared by several trade routes (see Section 3.2.1, Figure 6a). We therefore also have to keep track of traffic on a per-segment basis, with the traffic $T(s)$ over a segment $s \in S$ being the sum of traffic over all trade routes the segment is a part of. The available traffic capacity $T_A(s)$ over a segment $s \in S$ is given as $T_A(s) = \boldsymbol{O}_2 \cdot \text{WIDTH}(s) - T(s)$, where $\boldsymbol{O}_2$ is a scalar constant. Using this, we can define the available traffic capacity over a trade route $P_{i,j}$ as $T_A(P_{i,j}) = \min_{s\in P_{i,j}} T_A(s)$. Finally, due to the possibility of several trade routes sharing a segment and therefore also available capacity, saturating (making $\boldsymbol{t}_A(P_{i,j}) = 0$) a trade route $P_{i,j}$ might change available capacity $\boldsymbol{t}_A$ of other trade routes. Therefore, the available capacity $\boldsymbol{t}_A(P_i)$ for a list P_i of trade routes is determined by saturating, one-by-one, the trade routes $P_{i,j} \in P_i$ and summing the traffic required to saturate all of them. The traffic values are then restored to their previous values. .

With the above definitions, we can proceed to describe the traffic simulation itself; see Algorithm 2 for pseudo-code. We use the historical context's function $\boldsymbol{t}(t)$ to determine the new amount of traffic flowing between a pair p_i of cities. We then reset the traffic for all trade routes in P_i to zero and check whether there is enough available capacity $T_A(P_i)$ along all trade routes in P_i to accommodate the new traffic $\boldsymbol{t}(t)$. If there is enough available traffic capacity, we greedily (with respect to trade route length) redistribute it among all trade routes in P_i and no new trade routes are created. If, however, there is not enough available traffic capacity, we need to find a completely new trade route. We do this by evaluating two possibilities: 1) a new trade route P_A (see below) over existing major roads, or 2) a trade route P_B that uses existing minor roads or requires new major roads (see below). We pick the better candidate by choosing the one with the smaller $\boldsymbol{O}_3 \cdot \text{LENGTH}(P_k) + \boldsymbol{O}_4 \cdot \text{COST}(P_k)$, $k \in \{A,B\}$ and add it to the list P_i of trade routes for the i-th neighbouring city pair, adding and upgrading segments in the segment graph G as necessary. Finally, we again redistribute the traffic $\boldsymbol{t}(t)$ between all trade routes in P_i greedily.

Algorithm 2: Traffic simulation and major road growing.

```
foreach $(p_i, P_i)$ do
    $T(P_i) := 0$;      // traffic for all trade routes $\in P_i$ to 0
    if $T_A(P_i) < \mathbf{t}(t)$ ;       // is the new traffic too high?
    then
        $P_A :=$ PROPOSEFROMMAJOR();   // see Sec. 3.4.2
        $P_B :=$ PROPOSEFROMNEW();     // see Sec. 3.4.2
        $P_W :=$ WINNER$(P_A, P_B)$;   // choose the winner
        $P_i := P_i \cup P_W$;                // add the winner
        TRADEROUTETOSEGMENTGRAPH$(P_W, G(S,J))$;
    end
    GREEDILYREDISTRIBUTE$(\boldsymbol{t}(t), P_i)$;
end
```

Trade route over existing major roads. Finding a new trade route P_A over existing roads is simple. We construct a temporary graph from all existing segments that represent major roads. To segments that cannot accommodate the increase in traffic, we assign infinite weight. To other segments, we assign a weight as described in Section 3.2.2. In this temporary graph, we then find the minimum weight path.

Trade route over new major roads. Proposing a trade route P_B that requires an upgrade of existing minor roads or construction of new major roads is more complex. We start by defining the built-up area of a city and then use it construct a new, auxiliary path-finding graph $\mathcal{G}'$.

The segment graph G cuts up the map into pieces (faces of the *planar* segment graph G) that we call blocks. These are classified into 1) built-up blocks $B_{B,i}$, and 2) terrain blocks $B_{T,i}$. The first kind of blocks represents areas that do contain buildings, while the second represents larger undeveloped areas, such as fields or forests. We distinguish between them using a simple block-area classifier, which in our experiments has performed on par with more computationally demanding, e.g. shape-aware, classifiers. Next, we also define the city shell $\mathcal{S}$ to be the “outer block” (the outline of G; see Figure 8e). Using these, we give the built-up area B as

$$B = \left(\text{FILL}(\mathcal{S}) \setminus \textstyle\bigcup_i \text{FILL}(B_{T,i})\right) \cup \text{NB}(\mathcal{S}) \cup \left(\textstyle\bigcup_i \text{NB}(B_{T,i})\right) \quad (1)$$

where the FILL operator represents the inside of a block or the shell and the NB operator the neighbourhood of a block's or the shell's outline. The reasoning is detailed in Figure 8.

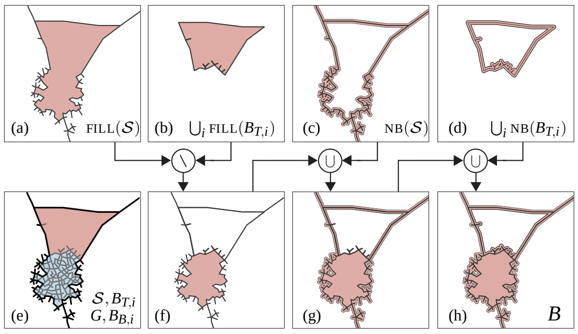
(a) FILL($\mathcal{S}$)
(b) $\bigcup_i$ FILL($B_{T,i}$)
(c) NB($\mathcal{S}$)
(d) $\bigcup_i$ NB($B_{T,i}$)
$\setminus$
$\cup$
$\cup$
(e) $\mathcal{S}, B_{T,i}$ $G, B_{B,i}$
(f)
(g)
(h) B

Figure 8: Illustration of Eq. 1. First, we realize that the inside of the city shell $\mathcal{S}$ (a) contains all of the city's blocks, both built-up and terrain, as shown in (e). Since we are only interested in the built-up areas, we subtract the inside of the terrain blocks (b), yielding the built-up blocks (f). Next, we want to add the neighbourhood of all roads. The neighbourhood of most is already contained in the intermediate result (f). Those still partially missing are the ones on the city shell (c) and the ones on terrain blocks (d). Adding these first yields (g) and then (h), which is our definition of the built-up area B. In (e), city shell $\mathcal{S}$ (thick line), built-up blocks $B_{B,i}$ (blue), terrain block(s) $B_{T,i}$ (red), and segment network G (gray) are shown. Shell and outlines are shown for clarity.

With a definition of the built-up area B in place, we proceed to construct, as illustrated in Figure 9, a pathfinding graph $\mathcal{G}'$ that will allow us to find paths over both segments and terrain. Since the built-up area B can only expand and can never shrink, we first take the pathfinding graph $\mathcal{G}$ and permanently remove from it all land edges $e \in \mathcal{E}_L$ that traverse the built-up area B. Next, we create a temporary copy $\mathcal{G}'$ of the pathfinding graph $\mathcal{G}$ and add to it all segments and nodes from the segment graph G. We continue by "stitching" these two components together by taking nodes along the city shell $\mathcal{S}$ together with all nodes along all terrain blocks $B_{T,i}$. We then connect them to nearby vertices in the set $\mathcal{L}_L$ of land vertices, provided that the new edge does not intersect or get too close to existing segments in G. Additionally, "stitches" connecting points the shell $\mathcal{S}$ alone are generated to allow for smoother roads in some cases (not shown in Figure 8). Using these "stitches", we connect the disconnected components (copies of the pathfinding and segment graphs $\mathcal{G}$, G) in $\mathcal{G}'$.

In order to control the characteristics of new major roads, such as the tendency to reuse existing major and minor roads, we multiply weights of edges in $\mathcal{G}'$ by coefficients $\boldsymbol{C}_1(t),\ldots,\boldsymbol{C}_6(t)$. In addition to edges representing major roads ($\boldsymbol{C}_1(t)$), minor roads ($\boldsymbol{C}_2(t)$), unbuilt harbours ($\boldsymbol{C}_3(t)$), built harbours ($\boldsymbol{C}_4(t)$), and (unbuilt) bridges ($\boldsymbol{C}_5(t)$), we also define an additional edge type—*extendible edges* ($\boldsymbol{C}_6(t)$)—which are not surrounded by built up blocks and can therefore be easily widened to allow for more traffic. This way, one can for example ask that a new major road reuses existing major roads heavily (by setting $\boldsymbol{C}_1(t)$ to a small value) and minor roads be avoided if possible ($\boldsymbol{C}_2(t)$ to a high value).

We propose a new trade route by using the resulting pathfinding graph $\mathcal{G}'$ as input to a modified Dijkstra's algorithm [Vol08] which penalizes turns in paths using $\boldsymbol{P}(t,\alpha)$. After the new major road is constructed, its extendible pieces are widened to the recommended width $\boldsymbol{A}(t)$ of a new major road.

3.5. User Input and Automation

So far, the ways the user can intervene in our simulation and its input have been left largely unmentioned. As has been described in Section 3.1, our algorithm's input can be divided into two sets: a historical context that is common to a larger group of cities, and per-city input that needs to be generated to keep our method automatic within the scope of a historical context. In contrast to previous methods that concentrated mostly on user-aided generation [VABW09] or required per-city datasets [WMWG09, PM01], this allows us to generate batches of cities with similar traits. Below, we detail automatic generation of per-city data, that is 1) the traffic function $\boldsymbol{t}(t)$, 2) the neighbouring city list $\boldsymbol{nc}$, and 3) the neighbouring city set $\boldsymbol{y}$ and its sublist $\boldsymbol{y'}$. As all of these are input for the algorithm itself, they need to be generated before the algorithm begins. Lastly, we discuss how the user can manipulate the segment graph G and list of nuclei Q.

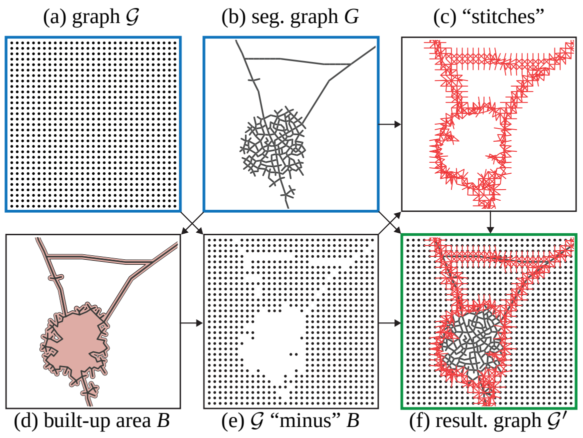

Figure 9: We start by generating the built-up area B (d) from the segment graph G (b), as described in Section 3.4.2 and Figure 8. We take the pathfinding graph $\mathcal{G}$ (a)—for clarity, only the vertices of $\mathcal{G}$ are shown—and update it by removing all edges and vertices that intersect the built-up area B, resulting in (e). Then we generate the edges—"stitches" (b); shell to shell "stitches" not shown—that connect the updated pathfinding graph $\mathcal{G}$ (e) and the segment graph G (b). Finally, we combine the "stitches" with the segment graph G (b) and the updated pathfinding graph $\mathcal{G}$ (e) to yield the resulting-graph $\mathcal{G}'$. Input has a blue border, result a green one.

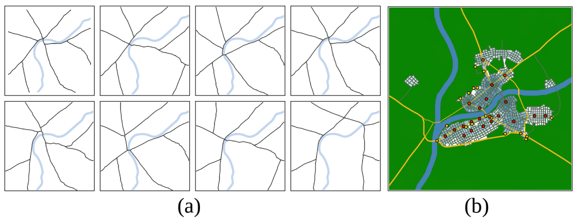

Figure 10: (a) Initial road networks corresponding to various automatically generated sublists ***y'*** of the set of all neighbouring city pairs ***y***. (b) Demonstration of the intervention capabilities of our method. Nuclei (coloured dots) and their strengths were manipulated to influence the locations of growth and the city's shape, and minor roads were added.

Traffic function. There are many sensible choices for generating the traffic function $\boldsymbol{t}(t)$. Since growth and trade are intertwined, as discussed in Section 3.4.1, we choose the function to be a slightly randomized copy of $\boldsymbol{t}(t) = \boldsymbol{b} \cdot \boldsymbol{G}(t)$, where $\boldsymbol{b}$ is uniformly selected from $\langle 0.7, 1.3 \rangle$.

Neighbouring cities. We generate the set of neighbouring cities $\boldsymbol{nc}$ using a heuristic. First, we generate the number $\boldsymbol{c} = |\boldsymbol{nc}|$ of neighbouring cities. With $\boldsymbol{c}_{min} = 4 < \mu < \boldsymbol{c}_{max} = 10$, we choose the number $\boldsymbol{c}$ of neighbouring cities from a binomial distribution with $n = \boldsymbol{c}_{max} - \boldsymbol{c}_{min}$ tries and a given expected value $\mu = 7 - \boldsymbol{c}_{min}$. Next, we generate the vertices on the edges of the lattice $\mathcal{L}$ that represent the neighbouring cities. We do this by generating $\boldsymbol{c}$ equally spaced rays emanating from the centre of the map X at angles $i\frac{2\pi}{\boldsymbol{c}} + \phi + \theta_i$ where $\phi \in \langle 0, 2\pi)$ and $\theta_i \in \langle -\frac{\pi}{2\boldsymbol{c}}, \frac{\pi}{2\boldsymbol{c}} \rangle$ are uniformly distributed global angular offset and per-ray perturbations.

List of pairs of neighbouring cities. Using a fixed $\boldsymbol{nc}$, we sort all pairs $\boldsymbol{y} = \{(C_u, C_v) \mid u < v \text{ and } C_u, C_v \in \boldsymbol{nc}\}$ of neighbouring cities by the angle $C_u X C_v$, where X is the centre of the map, removing pairs (smallest angles first) until the removal of a further pair would result in a disconnected initial road network. We then permute the remaining pairs, yielding a list of neighbouring city pairs ***y'***. As a result, we get a variety of initial road networks, as shown in Figure 10a. Further variation is achieved by adding or substituting the removed pairs into the list ***y'***.

Historical Context. Before automatic generation can start, the user is required to provide a set of values representing a historical context. In our experience, the system is stable and minor changes in parameter values cause only minor changes to the result's generic appearance. The result itself, however, might change significantly due to e.g., a major road being built in a different place and influencing the position of growth centers Q. Most of the values constituting the historical context have a well defined meaning. For example, the built harbour edge coefficient $\boldsymbol{C}_4(t)$ expresses how expensive changing the mode of transportation between land and sea would be. For some, such as the turn angle penalty function $\boldsymbol{P}(t, \alpha)$, a reasonable amount of experimentation is required.

Direct manipulation. We optionally let the user intervene in the algorithm by letting her/him add or remove nuclei of city growth during the simulation and also by allowing her/him to change their strengths. This, combined with the possibility of manipulating the growth and traffic functions $\boldsymbol{G}(t)$, $\boldsymbol{t}(t)$, and the segment graph G, allows the user to modulate the locality and strength of city growth. The user is also allowed to paint new roads directly (see Figure 10b).

Summary. We described methods for fully automatic generation of all per-city input, with the exception of terrain $\boldsymbol{m}$, which can be generated by published methods, e.g. [MKM89]. The definition of minor road patterns $\boldsymbol{K}(t)$ depends on the method used. In our case, the method of [PM01, WMWG09], a L-system equivalent, is used.

4. Results and Discussion

The algorithm was implemented in C# and uses CUDA to compute the *initial* pathfinding graph $\mathcal{G}$. The generation times for 8×8km maps for a lattice $\mathcal{L}$ with a spacing of $\ell = 20$m were between 2.5 and 4.5 minutes running on a single core of an Intel i7-2600K 3.4GHz CPU, with the construction of the pathfinding graph $\mathcal{G}$ run on a GeForce 480 GTX card.

For reproducibility, the parameter values for all presented results, together with images of all simulation steps, are available in the supplementary materials. Below, we discuss the results in the context of our contribution (in bold).

Positioning of settlements and water transportation. Unlike previous work, we put a strong emphasis on the placement of both initial and later settlements on the provided map. We place those settlements ("nuclei of city growth") at the intersections of roads and back this approach by available literature; see Section 3.2.1. The results of this strategy can be seen in the positioning of the initial settlements in Figure 12a,(i) and in the isolated settlements generated later, as seen in Figures 11a, 11c, and 11e. The positioning of nuclei of city growth is an important aspect of city development that is strongly influenced by trade *between* cities, something that has not been addressed by previous work. We further extend this approach by being the first to consider a *new* mode of transportation: *water*. By treating places where waterways turn into roads as intersections [Joh67], we allow the simulation to automatically create and position ports (e.g. New York, Hong Kong). This is shown in Figure 12a,(i) by generating a settlement on the coast and in Figure 12d, where a further port is generated to ease the pressure on the first port. Additional ports are also generated in Figures 11d and 11e.

Neighboring cities and traffic simulation. Our approach is the first to consider the influence of trade between neighboring cities. Here, our contribution is not in using a traffic simulation, but in using it to simulate *trade between cities*

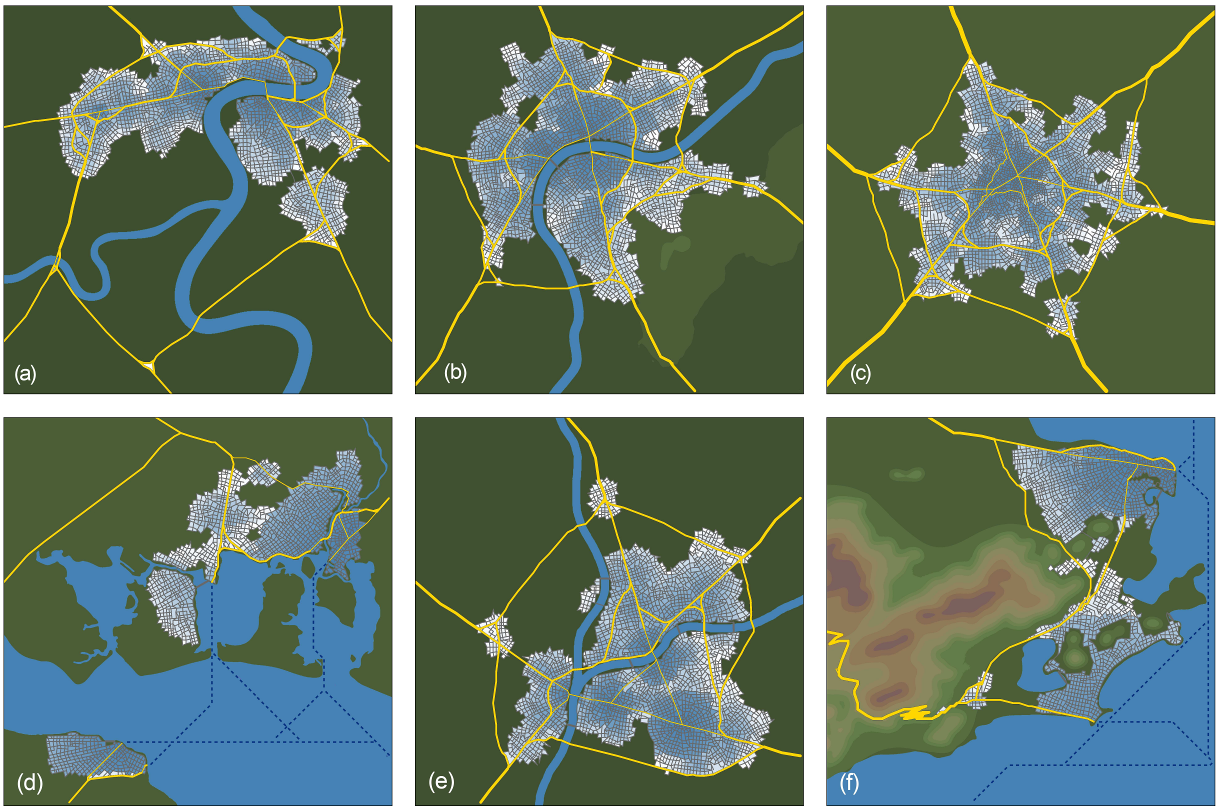

Figure 11: An eccentric city (a), a normal city (b), a city with a more classic major road structure (c), a coastal city spanning several pieces of land (d), a city with two isolated satellite suburbs (e), and a city in a difficult, mountainous region (f). Each quantization level represents a 60m vertical difference. Block colour indicates age. Major and minor roads are yellow and grey, respectively. For a discussion of the results, see Section 4. Minor roads that do not enclose a block are not shown in the visualization.

and identifying its impact on major road and city shape; see Section 3.4.1 for a thorough discussion. While previous work either requires the user to sketch major roads [VABW09] or determines their shape using *predefined* patterns [WMWG09, PM01], our trade simulation grows major roads *automatically*, without patterns, and *in a meaningful way*. Depending on historical era, major roads either need to navigate (Figure 11b) or circum-navigate (Figures 11 and 12) the built-up areas over large portions of terrain. This influences major road shape. In turn, with each new major road (waterway), new intersections, and with them, nuclei of city growth, are generated, influencing the city shape (kidney like shape in Figure 11a, rugged city edges in Figure 11c, position close to water in Figures 12, 11d, and 11f). This results in a relatively non-deterministic, but plausible major road structure with ring-roads (Figures 11b, 12d, 12e) and by-passes (Figure 11a). In the inset of Figure 12e, the development of major roads close to the second harbour is shown. Insets in Fig. 12f show the progression of a suburb, itself originally a few isolated blocks (*i*), to a compact settlement (*ii*), to when it's about to grow together with the city itself (*iii*).

Settlements growing together and urban landscape. Due to our way of positioning settlements (see above), we grow cities over time in the same way (see Figure 12) *real* cities are formed, that is by having several settlements (cities) grow together (e.g. LA or mostly any city); see Section 3.4. This is an original contribution, not found in previous work. The spectrum of urban forms (smaller and larger isolated suburbs, see Figure 11e, larger gaps within the cities, see Figure 11b) increases realism for applications where the city is seen as a whole (flight simulators, tycoon games, some RPGs) or where the user can navigate through the generated city (driving simulations, open world games), transitioning from outside of the city, through sparser areas and gaps, up into the inner city. The way the urban landscape transforms (sparser areas become dense, suburbs form conurbations) over time adds further realism to games that take place over long periods of time (open world games, tycoon games). Previous methods were unable to generates such cities without chance or user intervention.

Changing minor road patterns with time. We observe that the minor road patterns have a tendency to change with

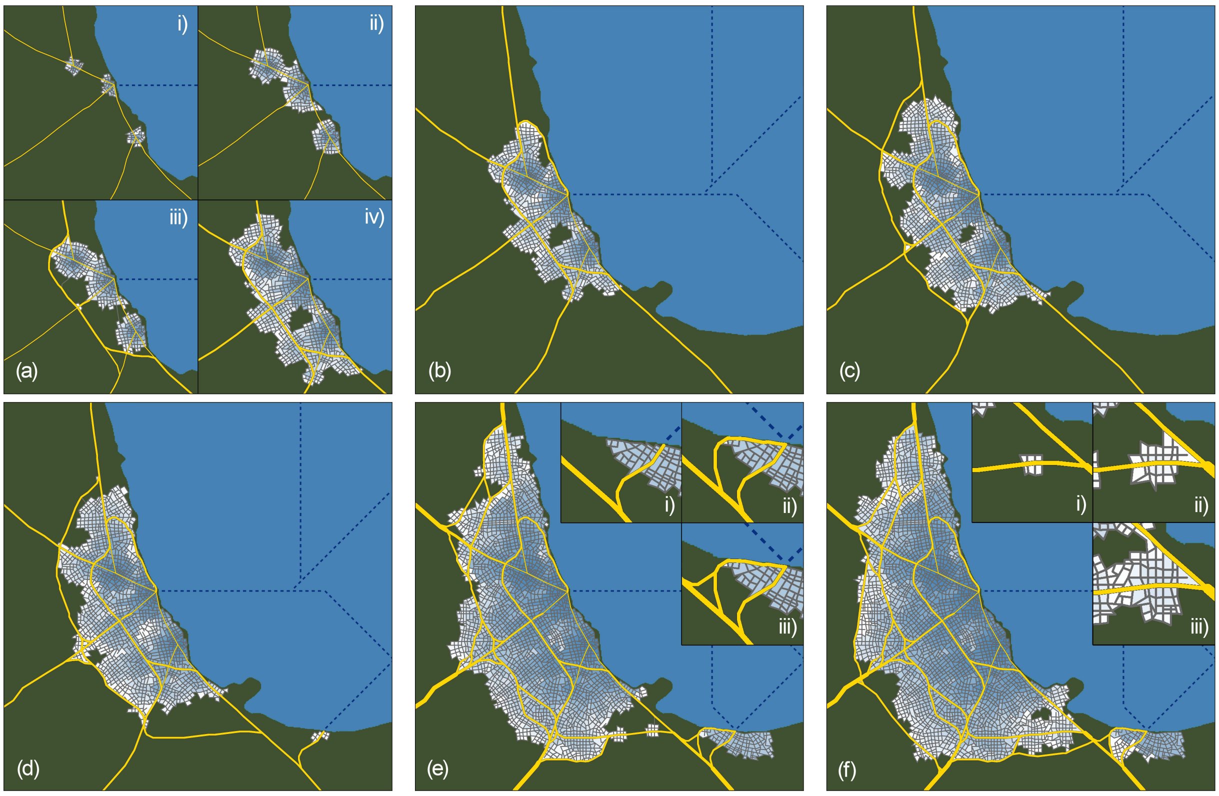

Figure 12: Evolution of a lake-side city with a harbour. Water trade-routes are dashed and in blue. See Section 4 for a discussion.

time and use this observation to grow our minor roads. The change is visible in both Figure 11 and Figure 12, with the older parts of the city being darker and the newer parts being lighter. This approach is a new alternative to the existing methods where the spatial distribution of minor road patterns was either user-defined [PM01] or where a land-use like simulation was used [VABW09, WMWG09]. Our results show that even non-land-use based minor road growth is possible.

Minimal per-city input. Whereas recent previous work [VABW09, WMWG09] is geared towards user-aided generation, our method is designed to require minimal input. This means it is maximally automatic, allowing for unsupervised generation of cities with similar traits. As a result, replay values in increased in games where content regeneration (tycoon games, MMORPG, open world games) is desirable.

Conclusions. We present a novel, maximally automatic method with several original contributions that generates cities with an rich urban structure (suburbs, gaps) and automatically generated and meaningful major roads over time by means of a trade simulation between neighboring cities.

4.1. Limitations and Future Work

While we hope to have provided an impulse to the field of procedural city modelling, we admit to the limitations of our method. First, our prototype implementation is not real-time. This is less of a problem where cities can be generated in the background as game-time progresses or where multiple cities are generated at once on several cores. For games that require developed cities quickly, a more production ready implementation would be necessary. Second, since our method does not simulate land-use (see Section 3.4.1), it has to be approximated using one of the existing methods before different types of buildings and other geometry can be generated. Also, if differing minor road patterns were to be generated for different land uses as in [VABW09], a land use simulation would have to be incorporated as a sub-step in each step of our algorithm. Next, the traffic $\boldsymbol{t}(t)$ is currently equal for all city pairs without significantly impairing the results. Different $\boldsymbol{t}(t)$ for each city pair could provide further improvements, but is outside of the scope of our paper. We also believe our method could benefit from also considering an intra-city traffic simulation such as the one in [WMWG09] while generating major roads. Furthermore, no current city simulation that develops cities over time has investigated the possibility of redevelopment within cities, including the change in road structure over time. Such redevelopments, as demonstrated by the Hausmann's renovation of Paris, are a non-trivial matter that remains largely unexplored. Lastly, including further modes of transportation and their "intersections", e.g. subway, train, and bus stops, as well as airports, could further enhance future results.

This work was supported by the Charles University in Prague, project GA UK No. 581412.

References

[AVB08] ALIAGA D. G., VANEGAS C. A., BENEŠ B.: Interactive example-based urban layout synthesis. *ACM Trans. Graph. 27*, 5 (2008). 2, 5

[Bee02] BEEVERS R.: *The Garden City Utopia: A Critical Biography of Ebenezer Howard*. Olivia Press, 2002. 1

[Bur08] BURGESS E. W.: The growth of a city: An introduction to a research project. In *Urban Ecology: An International Perspective on the Interaction Between Humans and Nature*, Marzluff J. M., (Ed.). Springer, 2008. 5

[CEW*08] CHEN G., ESCH G., WONKA P., MÜLLER P., ZHANG E.: Interactive procedural street modeling. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 103:1–103:10. 2

[Chi50] CHILDE V. G.: The urban revolution. *The Town Planning Review 21*, 1 (4 1950). 5

[Cos57] COSTA L.: Report of the pilot plan of Brasilia. Competition Entry, 1957. 1

[CR08] CORTES R., RAGHAVACHARY S.: *The RenderMan Shading Language Guide*. Thomson Course Technology, 2008. 6

[EBP*12] EMILIEN A., BERNHARDT A., PEYTAVIE A., CANI M.-P., GALIN E.: Procedural Generation of Villages on Arbitrary Terrains. *The Visual Computer* (2012). 2

[FZ01] FREY W. H., , ZIMMER Z.: Defining the city and urbanization. In *Handbook of Urban Studies*, Paddison R., Lever W., (Eds.). Sage Publications, 2001. 2, 5

[Gar92] GARREAU J.: *Edge City: Life on the New Frontier*. Anchor, 1992. 5

[GPGB11] GALIN E., PEYTAVIE A., GUÉRIN E., BENEŠ B.: Authoring Hierarchical Road Networks. *CG Forum 30*, 7 (2011). 2

[GPMG10] GALIN E., PEYTAVIE A., MARÉCHAL N., GUÉRIN E.: Procedural Generation of Roads. *CG Forum 29*, 2 (2010). 2, 3, 4

[GSdKB09] GROENEWEGEN S. A., SMELIK R. M., DE KRAKER K. J., BIDARRA R.: Procedural City Layout Generation Based on Urban Land Use Models. In *EG Short Papers* (2009). 2, 5

[Hoy39] HOYT H.: *The structure and growth of residential neighborhoods in American cities. Washington, DC.* US Gov't Printing Office, Washington, D.C., 1939. 5

[Joh67] JOHNSON J. H.: *Urban Geography, An Introductory Analysis*. Pergamon Press, London, UK, 1967. 3, 5, 8

[KMK12] KRECKLAU L., MANTHEI C., KOBBELT L.: Procedural interpolation of historical city maps. In *EG 2012* (2012), Eurographics Association. 2

[KP06] KNOX P., PINCH S.: *Urban Social Geography, An Introduction*. Pearson Education, United Kingdom, 2006. 5

[LRW*06] LECHNER T., REN P., WATSON B., BROZEFSKI C., WILENSKI U.: Procedural modeling of urban land use. In *SIGGRAPH '06 posters* (2006). 5

[LWW03] LECHNER T., WATSON B., WILENSKY U.: Procedural city modeling. In *1st Midwestern Graphics Conference* (2003). 2

[MKM89] MUSGRAVE F. K., KOLB C. E., MACE R. S.: The synthesis and rendering of eroded fractal terrains. *SIGGRAPH '89* (1989). 1, 8

[MWH*06] MUELLER P., WONKA P., HAEGLER S., ULMER A., GOOL L. V.: Procedural modeling of buildings. *ACM Trans. Graph. 25*, 3 (2006). 2

[MYDN01] MIYAZAKI R., YOSHIDA S., DOBASHI Y., NISHITA T.: A method for modeling clouds based on atmospheric fluid dynamics. In *Proc. Pacific Graphics 2001* (2001), pp. 363–372. 1

[PL96] PRUSINKIEWICZ P., LINDENMAYER A.: *The algorithmic beauty of plants*. Springer, 1996. 1

[PM01] PARISH Y. I. H., MÜLLER P.: Procedural modeling of cities. In *ACM SIGGRAPH 2001* (2001), ACM Press. 1, 2, 5, 7, 8, 9, 10

[ŠBBK08] ŠŤAVA O., BENEŠ B., BRISBIN M., KŘIVÁNEK J.: Interactive terrain modeling using hydraulic erosion. In *Eurographics Symposium on Computer Animation* (2008). 1

[Sjo60] SJOBERG G.: *The Preindustrial City: Past and Present*. The Free Press, New York, 1960. 5

[Soj00] SOJA E. W.: *Postmetropolis, Critical Studies of Cities and Regions*. Blackwell Publishers, Malden, MA, USA, 2000. 5

[TT95] TOLLEY R., TURTON B.: *Transport Systems, Policy and Planning: A Geographical Approach*. Longman, 1995. 5

[VABW09] VANEGAS C. A., ALIAGA D. G., BENEŠ B., WADDELL P. A.: Interactive design of urban spaces using geometrical and behavioral modeling. In *SIGGRAPH Asia '09* (2009). 1, 2, 7, 9, 10

[VGDA*12] VANEGAS C. A., GARCIA-DORADO I., ALIAGA D., BENES B., WADDELL P.: Inverse design of urban procedural models. *ACM Trans. Graph.* (2012). 2

[Vol08] VOLKER L.: *Route Planning in Road Networks with Turn Costs*. Tech. rep., Universität Karlsruhe, 2008. 7

[WMWG09] WEBER B., MÜLLER P., WONKA P., GROSS M.: Interactive geometric simulation of 4D cities. *CG Forum 28*, 1 (2009). 1, 2, 5, 7, 8, 9, 10

[YS12] YU Q., STEED A.: Example-based road network synthesis. In *EG 2012 - Short Papers* (2012), Eurographics Association. 2, 5

[YWVW13] YANG Y.-L., WANG J., VOUGA E., WONKA P.: Urban pattern: Layout design by hierarchical domain splitting. *ACM Trans. Graph. 32* (2013). 2