Computational Network Design from Functional Specifications

Chi-Han Peng[1,2] Yong-Liang Yang[4] Fan Bao[1] Daniel Fink[5] Dong-Ming Yan[3,6] Peter Wonka[3,1] Niloy J. Mitra[2]

[1]ASU [2]UCL [3]KAUST [4]Bath Univ. [5]Urban Agency [6]NLPR-CASIA

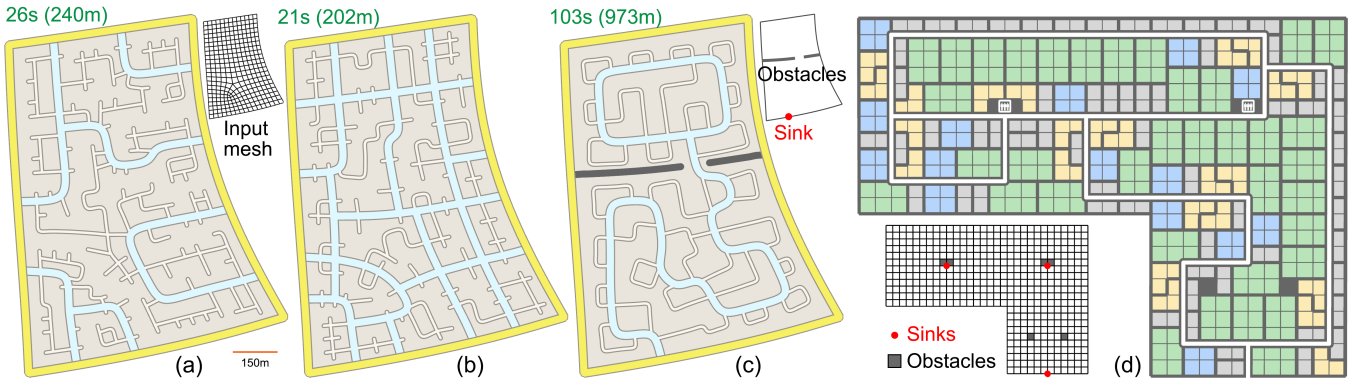


Figure 1: *We propose an algorithm that generates networks for design scenarios like mid-scale urban street layouts (a-c) and floorplans for office spaces (d). The user simply specifies an input mesh as the problem domain along with high-level specifications of the functions of the generated network. Examples include preference for interior-to-boundary traffic (a) or interior-to-interior traffic (b), networks with specified destinations (i.e.,* sinks*) on the boundary (c,d) and local feature control, such as reducing T-junctions (b) or forbidding dead-ends (c). For (a-c), the average travel time (distance) for interior-to-boundary traffic as estimated by the traffic simulator SUMO is indicated in green.*

Abstract

Connectivity and layout of underlying networks largely determine agent behavior and usage in many environments. For example, transportation networks determine the flow of traffic in a neighborhood, whereas building floorplans determine the flow of people in a workspace. Designing such networks from scratch is challenging as even local network changes can have large global effects. We investigate how to computationally create networks starting from *only* high-level functional specifications. Such specifications can be in the form of network density, travel time versus network length, traffic type, destination location, etc. We propose an integer programming-based approach that guarantees that the resultant networks are valid by fulfilling all the specified hard constraints and that they score favorably in terms of the objective function. We evaluate our algorithm in two different design settings, street layout and floorplans to demonstrate that diverse networks can emerge purely from high-level functional specifications.

Keywords: network layout, functional specifications, urban planning, optimization, computational design

Concepts: •**Computing methodologies** → *Shape analysis;*

1 Introduction

Behaviors in an urban neighborhood are largely dictated by the underlying networks, both at global and at local scales. For example, in the context of urban planning, the transportation network determines the traffic flow in a neighborhood; or, in the context of building interiors, the layout of corridors dictates the access times for internal foot-traffic and exit times.

While designing a complete city from scratch is less often required, urban planners are often asked to redesign or rejuvenate a street network in a local neighborhood. We interviewed several academics and professionals in traffic engineering, urban planning, and architecture to inquire about their current design practices. In such scenarios, urban planners often propose network connections drawing from their knowledge and prior experience. The designs are then tested by traffic engineers to provide performance feedback to the planners, who then refine the layouts. This slows down the iterative design process. Further, manually designing traffic networks is very difficult for a human as it is non-trivial to predict how a network will function only based on visual inspection.

In such a setting, a designer would instead want to create networks by simply describing *how* the target environments should function. We refer to such high-level descriptions as *functional specifications*. For example, functional specifications can come in the form of desired network density, transportation patterns, through-traffic, access times, local features, etc.

In this paper, we study how to design networks starting *only* from such functional specifications. We observe that in network planning the designer has to primarily balance between conflicting requirements: networks should be densely connected in order to obtain low average transportation times; at the same time, the total length of the network should be small to leave space for other assets (e.g., houses, shops). We produce a desirable network layout based on a novel integer programming (IP) based approach that takes as input a set of functional specifications and boundary description of the target domain. Technically, the proposed IP formulation guarantees that the designed networks are *valid* by ensuring that they are free of islands (see Figure 4 for examples) and that they offer sufficient coverage over the target domain, while having desirable *quality* as measured by the target functional specifications.

The proposed algorithm is evaluated in the context of urban street layouts and floorplans. We identify a set of commonly appearing functional specifications (see Section 3) and propose how to effectively model them (see Section 4). For example, Figure 1 shows different networks created by our algorithm using different functional specifications. In summary, our main contributions are:

- proposing the problem of network design directly from functional specifications;

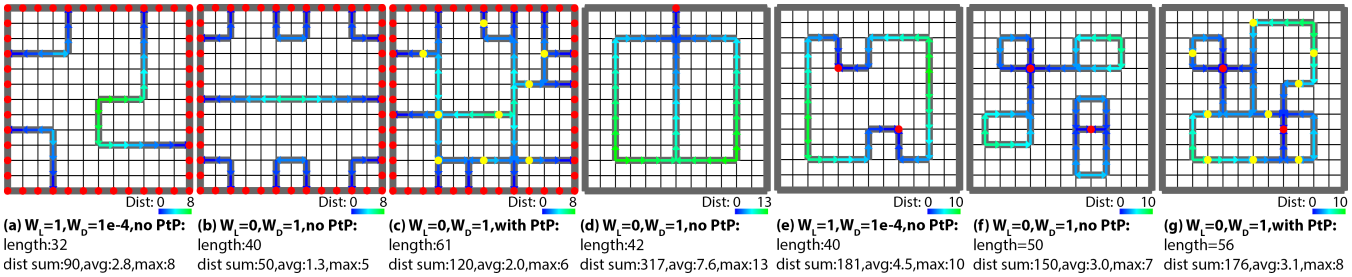


Figure 2: *Example results. Sink vertices are marked in red. The coverage range is two edges wide. The distance values of active half-edges are colored (see legends for color ranges). Boundary edges are excluded from the calculations. We forbid deadends and edges that are too close to each other. (a) to (c): A typical urban layout scenario such that all boundary vertices are sinks. The first two use different weights to optimize for (a) numbers of network edges and (b) sum of distance values. Note that we do not specifically constrain the maximum distance values. (c): A result that also optimizes for distance values but with the point-to-point constraint enabled (sampled vertices are marked in yellow). (d): We now constrain a boundary vertex (top middle) to be the sole sink. (e) to (g): A typical floorplan scenario such that a few inner vertices are sinks (e.g., elevators). Similarly, they differ by different relative weights and whether the point-to-point constraint is enabled.*

- formulating the task as an integer program that supports common functional specifications; and
- evaluating the method in different design contexts.

2 Related Work

Street modeling. Initial work on street network modeling focused on algorithms to synthesize street networks that resemble existing ones. One approach is to grow street segments greedily until the available space is filled [Parish and Müller 2001; Weber et al. 2009]. An alternative approach is to first sample points on the street network that are connected in a subsequent algorithm step [Aliaga et al. 2008]. Chen et al. [2008] proposed the use of tensor fields to guide the placement of street segments. One way to improve synthesis algorithms is to optimize the quality of street networks to include local geometric and quality metrics, such as street network descriptors [AlHalawani et al. 2014], sunlight for resulting buildings [Vanegas et al. 2012], shape of individual parcels [Yang et al. 2013; Peng et al. 2014b], or the shape of individual roads interacting with the environment [Maréchal et al. 2010]. There were some initial attempts to include global traffic considerations into the layout process. A first attempt was to compute a traffic demand model and use this model to modify street width or to guide expansion of the street network [Vanegas et al. 2009; Weber et al. 2009]. The connectivity of the road network is also a fundamental requirement for generating high-level roads connecting cities and villages [Galin et al. 2011]. A recent paper describes how to design traffic behavior in an urban environment [Garcia-Dorado et al. 2014]. This work touched on aspects of traffic design that complement our proposed system. While most of the proposed components are orthogonal to our paper, one important component of that system is an algorithm to modify an existing street network by making low-level random modifications. Instead, we focus on directly generating the topology and geometric layout of the initial coarse network *only* from functional specifications.

Layout modeling. In the search for creating realistic virtual environments, layouts have been modeled in computer graphics for diverse settings. For certain target scenarios, transportation networks have no significant role, e.g., in the distribution of vegetation [Deussen et al. 1998], or the distribution of window configurations in urban facades [AlHalawani et al. 2013]. However, there are many examples of layouts that have at least some network aspect to them. In furniture layouts by Yu et al. [2011], the existence of obstacle-free walking paths was a consideration in modeling the objective function. Several biologically inspired simulations also model networks. For example, Runions et al. [2005] generate leaf venation patterns that result in fascinating graphs; Genevaux et al. [2013] model river networks using hydrological principles; Liu et al. [2013] investigate precast fabrication-friendly floorplan layouts, and Bao et al. [2013] explore shapes of good building layouts. Also important is the layout of building floorplans [Merrell et al. 2010], game levels [Ma et al. 2014], or warehouses [AlHalawani and Mitra 2015], as the rooms, hallways, and storage aisles also induce networks.

Traffic engineering. In the context of urban planning, the more technical aspects of designing road networks are studied in traffic engineering. A good review of traffic engineering can be found in the textbook by Ortuzar and Willumsen [2011]. A more theoretical branch of traffic engineering draws inspiration from network design [Yang and Bell 1998], which can be formulated as an abstract problem of assigning edge weights and capacities between nodes. Some existing methods also use IP in their problem formulation, e.g., [Koster et al. 2010; Luathep et al. 2011]. However, the approaches are different as we consider networks as graphs to be embedded in a 2D plane. Our results avoid intersections between edges, provide reasonable block shapes and avoid unreasonable road shapes (e.g., zig-zag roads). Further, the number and location of the crossings are treated as unknowns in our formulation.

Traffic simulation. Once a network has been designed, it can be analyzed using traffic simulation. There are several popular toolkits available, e.g., [MATSim 2015; VISSIM 2015]. In computer graphics, there are also multiple simulators that produce compelling visual output [Sewall et al. 2010; Sewall et al. 2011; Wilkie et al. 2013], but we focus on network performance (e.g., traffic throughput) rather than visual characteristics. After consultation with multiple urban planners and traffic engineers, we picked the state-of-the-art system SUMO [Krajzewicz et al. 2012] for our evaluation as it computes traffic-dependent dynamic route assignments.

3 Functional Specifications

Our goal is to allow users to create networks simply from a set of *functional specifications* describing how a synthesized network should behave. In an effort to understand current design practices, we interviewed multiple academics and processionals with extensive experience in urban design and planning. We also consulted the relevant literature in the field [Meyer and Miller 2000; Handy et al. 2003; Southworth and Ben-Joseph 2003; Board 2010]. In this section, we summarize our findings and identify a set of relevant functional specifications to support.

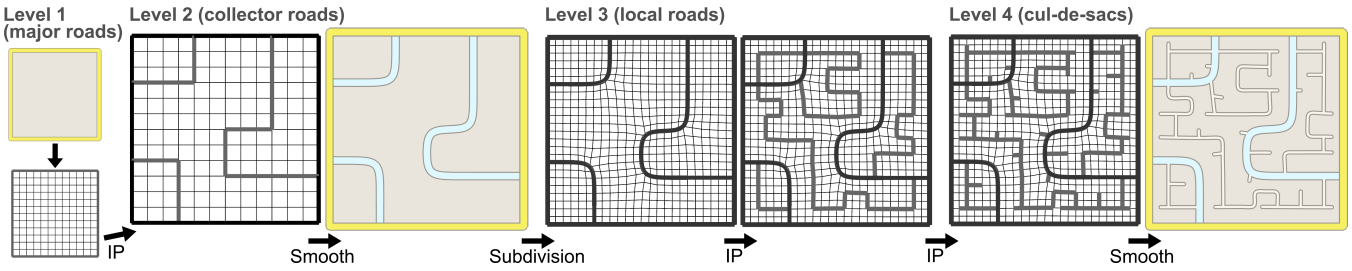


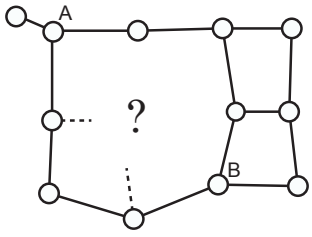
Figure 3: *Pipeline. For each sub-region, as determined by the input major roads, we generate layouts in three levels of decreasing coverage ranges. For each level, a rough street network is first generated by the IP-based approach (shown in gray). Afterwards, the geometry of the generated street network is refined by a smoothing process. Dead-ends are typically allowed only at the last level.*

Current design practice. The process for creating spatial compositions relying on foundational network topologies is necessarily complex and interdisciplinary. Traditionally, designers rely on a wide range of inputs, including intuition, experience, rules-of-thumb, analytic modeling and simulation, all combined in an iterative design-and-test process. As described by Marshall [2005], underlying frameworks for this design process are either prescriptive or preferred hierarchies for patterns of streets. The process of designing urban environments is a multi-disciplinary endeavor involving urban designers and planners, landscape architects, transport planners and traffic engineers, each with particular sets of knowledge to apply and optimal conditions to design.

Iteration occurs through gradual testing and refining design options to aim toward better performance and preferred traffic flows, through gradual adoption of inputs and modifications from design consultants. While some of these inputs are formulated as specific or quantifiable requirements, others are more complex and inscrutable, relying on intuition and perception. Further, these disciplines each have particular functional or feature requirements. For example, transport planners may have modal split targets; traffic engineers may deduce the need for X- or T- intersections; and urban designers may aim for vibrancy and activation of shopping streets. Much of the design process is spent in aligning such qualitative and quantitative requirements, which are often conflicting.

Currently, there is limited support for automatic generation of urban street layouts. Existing techniques for measuring the performance of a given network (e.g., space syntax considerations, traffic simulation, etc.) analyze only a *given* input network. The few tools that do automatically synthesize networks usually operate in a framework based on geometric resemblance to existing precedents (i.e., example-driven synthesis [Nishida et al. 2015]), do not directly support functional requirements, have difficulty responding to particular conditions of the sites context, and in general can be too brittle for use in the real design process. Hence, the majority of current urban designs and network layouts is composed, modeled and adjusted manually, while computational support is used only for (forward) analysis.

Transport modeling. A transportation network can be abstracted as a graph with nodes and edges (see below). Important components in transport modeling are supply, demand, and assignment. We explain these three concepts, namely supply, demand and assignment, using the example of road networks. *Supply* is defined by the existing road network and regulations, while *demand* is concerned with the desired movement of people or goods from one location to another. Supply is what we would like to model in our system. As input, we consider a partial road network where at least the nodes and edges on the boundary of a region are given. Our goal is to construct a network inside the region by computing the number and location of nodes (intersections) as well as the location and connectivity of the edges. Further, we assign a speed limit to each edge. Demand is specified for each pair of nodes in the network, e.g., 300 cars per hour want to travel from node A to node B in the network. Such demand typically depends on land use, e.g., where residential houses, jobs, and stores are located and in what density, and the available road network, e.g., people are more likely to make a trip to the store if the trip is short. Travel demand can be measured using sensors, simulated using large-scale urban simulation, or predicted by simple models. While supply and demand are connected, a reasonable assumption, especially in developing smaller areas, is that there is a fixed demand or a set of fixed demand scenarios (i.e., supply changes do not result in new demand requirements). Following this strategy, we assume a fixed travel demand for trips potentially passing through the developed region – an assumption that is typically valid for mid- and small-scale networks.



Another component in transport modeling is the *assignment* of demand (e.g., trips) to paths in the road network. Classical models for assignment follow *Wardrop's first principle* [Wardrop 1952], i.e., the equilibrium principle. These assignment models assume that individuals try to minimize their travel cost, evaluated as a combination of factors, such as distance, travel time, tolls paid, number of stops, and scenery. We choose a simple cost model that mainly depends on travel time, but other choices are also compatible with our system. We validate our results using a state-of-the-art traffic simulator, which implements more sophisticated assignment and travel cost models.

Design requirements. In this work, we focus on designing small and medium scale layouts purely from functional specifications. Incorporating functional requirements holistically into a design option is difficult, especially when the interactions between these requirements are numerous and complex. Both the connectivity of the networks and their geometric realizations (e.g., nicely shaped neighborhood blocks) are important in this setting.

We identified a set of recurring functional specifications as follows:

(i) *Density:* The desired density of a network encodes the average spacing between the network's edges (e.g., density of streets).

(ii) *Network lengths versus travel distances:* In networks with comparable densities, two extremes are possible. On one extreme, the total network length can be minimized. On the other extreme, networks can facilitate more efficient travel, usually toward certain

destination locations predefined on the domain. In our framework, the user can stipulate a relative preference for the two.

(iii) *Traffic types:* We support three types of traffic: interior-to-boundary, interior-to-interior, and boundary-to-boundary traffic. Networks arising from different types of traffic specifications tend to look quite different (e.g., Figures 6a, 6e, 6d). Users can indicate a preference among the three.

(iv) *Sink locations:* A key function of networks is to facilitate access to certain predefined destination locations, i.e., sinks. We allow the shape of a network to be controlled by the distributions of such sinks. Intuitively, a network tends to look like a root-like structure grown from the destination locations. The user can select the sink location to be: (a) all of the boundary, (b) only at a subset of the boundary, or (c) at the interior of the target domain (e.g., shopping malls).

(v) *Local features:* There are certain local features that can have profound effects on the appearance of the target networks. Examples are deadends, branches, and T-junctions. We allow users to indicate if such features should be forbidden or should appear infrequently.

(vi) *User specifications:* We provide two ways to directly control the generated networks. The first is specifying certain locations as *obstacles* that the generated network must avoid. The second is enforcing certain routes to appear in the generated network. An example is a direct route between two boundary locations to boost through-traffic.

4 IP-based Network Design

4.1 Formulation

In this section, we introduce an integer-programming (IP) based optimization to design networks from the functional specifications (see Section 3). We assume that the input problem domain, given as a piecewise linear 2D polygon, is discretized into a polygonal mesh, M. Our goal is to select a subset of the edges in M as the final network. We consider a selected network to be **valid** if it satisfies two constraints: (C1) a *no-island constraint* that ensures access to specified destination locations (i.e., *sinks*) predefined on the domain; (C2) a *coverage constraint* that ensures that the network sufficiently covers the whole domain (i.e., all vertices in M are within a specified distance to the network).

We judge the set of such valid networks based on a set of *quality measures*: (Q1) we prefer a network with smaller total length as longer networks are more expensive to build and maintain, and they take up more usable space from the domain; (Q2) we prefer a network with shorter travel distances to the sinks (from every vertex in the network). Our objective function is a weighted sum of these two competing measures. We start with some definitions before explaining the constraints and the quality measures.

Definition 4.1 *A* network *is a subset of the edges in M. An edge is* active *if it is in the subset; otherwise, it is* inactive*. A vertex is* active *if any of its adjacent edges are active; otherwise, it is* inactive*. The* sinks *are a predefined subset of the vertices in M.*

We use Boolean indicator variables, E_m, to model the active/inactive states of every edge, e_m, for $m \in [0, N_E)$ with N_E being the number of edges in M.

(C1) No-island constraint. We require valid networks to be free of islands (i.e., unreachable parts), while still allowing loops in the networks. First, without changing the definition of a network, we distinguish each half-edge, $e_{i\to j}$ (i.e., goes form vertex v_i to v_j), as active or inactive, by a Boolean indicator variable $E_{i\to j}$, for $i, j \in [0, N_V)$ with N_V being the number of vertices in M. Our goal is to assign each active half-edge, $e_{i\to j}$, a *distance value* that roughly encodes the distance of v_j to the closest sink vertex along the active half-edges in the network. We model the distance value of half-edge $e_{i\to j}$ as a non-negative continuous variable, $D_{i\to j}$, for $D_{i\to j} \in [0, \Delta_{all}]$ where Δ_{all} is the sum of the lengths of all half-edges in M. This translates to the following requirement:

Proposition 4.2 *For a half-edge, $e_{i\to j}$, to be active, at least one of its succeeding half-edges (i.e., the half-edges that go from v_j but not go back to v_i) is also active and is assigned a distance value smaller than the distance value of $e_{i\to j}$, except if v_j is a sink vertex.*

We encode this requirement (see Appendix for proof) as: For every succeeding half-edge, $e_{j\to k}$, for $k \in \mathcal{N}_j \setminus \{i\}$, where $\mathcal{N}_j$ is the one-ring neighborhood of v_j, of every half-edge $e_{i\to j}$ in M such that v_j is not a sink vertex,

$$L_{i\to j;j\to k} \le E_{j\to k} \quad \text{and} \tag{1}$$

$$D_{j\to k} - D_{i\to j} + \Delta_{all} L_{i\to j;j\to k} \le \Delta_{all} - \Delta_{i\to j} \tag{2}$$

where, $L_{i\to j;j\to k}$ are auxiliary Boolean variables associated with every half-edge, $e_{i\to j}$, one for each of its succeeding half-edges, $e_{j\to k}$, for $i, j \in [0, N_V)$, $k \in [0, K)$. The length of half-edge $e_{i\to j}$ is $\Delta_{i\to j}$. Note that $L_{i\to j;j\to k}$ can be true only if (i) $e_{j\to k}$ is active (enforced by Inequality (1)), and (ii) $e_{j\to k}$ has a smaller distance value than the distance value of $e_{i\to j}$ by at least the length of $e_{i\to j}$ (enforced by Inequality (2)).

Next, the following inequality ensures that for $e_{i\to j}$ to be active, at least one of its auxiliary variables must be true: For every half-edge in M, $e_{i\to j}$, such that v_j is not a sink vertex,

$$E_{i\to j} - \sum_{0\le k<K} L_{i\to j;j\to k} \le 0. \tag{3}$$

The inset illustrates the no-island constraint using the top-middle part of Figure 4(d) with vertex indices. As all edges have uniform lengths, constants $\Delta_{i\to j}$, $i, j \in [0, N_V)$, all equal to one. Continuous variables $D_{0\to 1}$, $D_{1\to 2}$, $D_{1\to 3}$, and $D_{1\to 4}$ are assigned to be 2, 0 (arbitrary), 2, and 1, respectively. Therefore, by Inequality (2), $L_{0\to 1;1\to 2}$ and $L_{0\to 1;1\to 3}$ are false and $L_{0\to 1;1\to 4}$ is true. This allows $E_{0\to 1}$ to

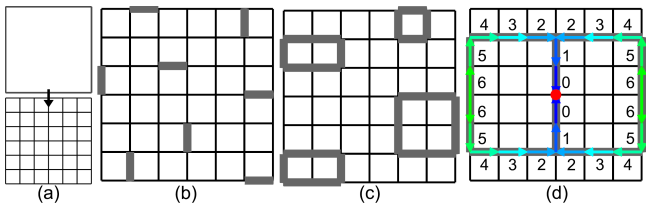


Figure 4: *The impact of no-island constraints. The coverage range is one edge wide. (a) The problem domain is first discretized into a mesh. (b) A network (i.e., a subset of the edges) that sufficiently covers the domain using the fewest possible number of edges. However, the solution may consist of many disconnected parts. (c) Simply forbidding degree-1 vertices is not enough to guarantee the network to be free of islands, as it is still possible to form disconnected loops. (d) Our no-island constraint guarantees that the network is entirely connected to the sink (red vertex). Here, we show the distance values assigned to each active half-edge.*

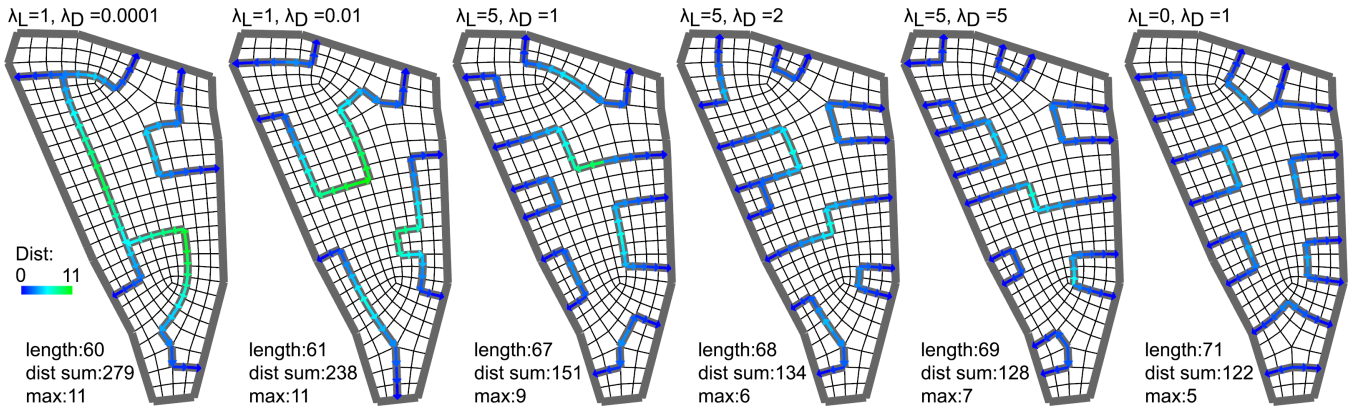


Figure 5: *Optimization results with changing λ_L (to minimize network lengths) versus λ_D (to minimize travel distances). A larger λ_L leads to shorter network lengths but longer travel distances, while a larger λ_D leads to the opposite.*

be true by Inequality (3). Note that the requirement does not forbid loops in networks, as shown in Figure 4.

Finally, we say that an edge is active if and only if at least one of its two half-edges is active: For every edge in M, e_x,

$$-1 \leq E_{i\to j} + E_{j\to i} - 2E_x \leq 0, \tag{4}$$

where $E_{i\to j}$ and $E_{j\to i}$ are the Boolean indicator variables of e_x's two half-edges. The Boolean indicator variable of e_x is E_x.

Note: The above formulation was inspired by an IP formulation of the Traveling Salesman Problem [Miller et al. 1960]. However, unlike theirs, our formulation allows loops involving the starting nodes (i.e., sink vertices).

(C2) Coverage constraint. To ensure that the network sufficiently covers the whole domain, we require that an active vertex covers itself and its nearby vertices within a distance threshold. (Different coverage models can instead be used for different design scenarios.) A network sufficiently covers M if all the vertices in M are covered. We model this as described next.

We denote the active/inactive states of every vertex in M, v_y, as Boolean indicator variables V_y, $y \in [0, N_V)$. Since a vertex is active if and only if at least one of its adjacent edges is active, we have the following constraint: For every vertex in M, v_y,

$$1 - |\mathcal{E}_y| \leq \sum_{x\in\mathcal{E}_y} E_x - |\mathcal{E}_y| V_y \leq 0, \tag{5}$$

where $\mathcal{E}_y$ is the set of edges that are adjacent to v_y. We now express the coverage requirement as, for every vertex $v \in M$,

$$\sum_{x\in V_v^{cover}} V_x \geq 1, \tag{6}$$

where V_v^{cover} is the set of variables of the vertices that covers v.

Objective function. We balance between two quality measures: (Q1) the total length of the network, and (Q2) the total travel distances to the sinks. The first term can be expressed as the summation of all edge indicator variables multiplied by each edge's length. The second term can be expressed as the summation of all distance values of half-edges. Thus, the objective function takes the form:

$$\min_{E_{i\to j}, D_{i\to j}, L_{i\to j;j\to k}, E_x, V_y} \quad \lambda_L \sum_x \Delta_x E_x + \lambda_D \sum_{i,j} D_{i\to j},$$

where Δ_x is the length of edge e_x, λ_L is the weight of the total length term, and λ_D is the weight of the total travel distance term. Note that when λ_D is set to zero, the distance values may not be assigned correctly and may need to be calculated in a post-process. In Figure 5, we analyze how the assignments of λ_L versus λ_D affect the optimization results.

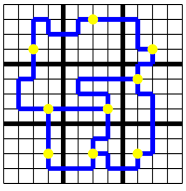
4.2 Extensions: Optional Functional Specifications

We also support two optional quality measures: (L1) ensures quick access between any two locations in the network by a *point-to-point constraint* (note that this is not guaranteed if we only optimize for quick access to the sinks); and (L2) introduces measures for controlling the local features (e.g., deadends, zigzags, and T-junctions) in a network.

(L1) Point-to-point constraint. Here, our goal is to constrain the travel distances between any two vertices in the network, not just the travel distances to the sinks. While explicitly modeling such constraints is possible, it would be prohibitively expensive since we need to model every possible path between every possible pair of vertices. Inspired by the construction of k-spanners for graphs [Baswana and Sen 2007], we describe a cost-effective way to approximate our goal.

We first partition M into a set of sub-meshes (i.e., a connected set of faces). We assume the partition is modeled by the user. We say that two sub-meshes are adjacent to each other if they share common edges. Next, we randomly sample one vertex in every sub-mesh. For sampling, we consider vertices that are not adjacent to any other sub-mesh, unless such vertices do not exist. For every pair of adjacent sub-meshes, we exhaustively enumerate the set of paths (i.e., a consecutive sequence of edges) connecting the two sampled vertices with topological lengths not greater than the length of a shortest path between the two vertices plus a tolerance value (we use 2). We can now set constraints to require at least one of the paths is selected (i.e., consisting of active edges). In summary, we require the network to connect every pair of adjacent sub-mesh by connecting their respective sampled vertices.

The inset shows an example for Figures 2c and 2g. We now describe the modeling. Let $P_{a\to b,x}$ be a Boolean indicator variable indicating the presence of the x-th path among the set of paths connecting two sampled vertices v_a (of sub-mesh M_a) and v_b (of sub-mesh M_b), for $a, b \in [0, N_S)$ and $x \in [0, X_2)$ with X_2



being the size of the set. Let $E_n^{a\to b,x}$, $n = 0, 1, ..., N-1$ denote the edges on path $P_{a\to b,x}$, where N is the size of the path. We have:

$$-N + 1 \leq NP_{a\to b,x} - \sum_n E_n^{a\to b,x} \leq 0. \quad (7)$$

For every set of paths connecting sampled vertices v_a and v_b,

$$\sum_x P_{a\to b,x} \geq 1, \quad (8)$$

indicating that at least one such path is selected. As shown in Figures 2c and 2g, enforcing such constraints leads to networks that are more tightly connected, which in turn have quicker access between any two vertices in the network. Users can control the strength of the point-to-point constraint by the density of the partitions. While this approach is computationally cheap, it can overconstrain the resulting network.

(L2) Local feature control. We now introduce quality measures controlling local features.

(i) Deadend avoidance: We may desire networks that have no deadends, i.e., an active vertex that is adjacent to just one active edge. To achieve this, we give every half-edge, $e_{i\to j}$ (from vertex v_i to v_j), a *non-emptiness* Boolean indicator variable, $\nu_{i\to j}$, which is true if any of the v_j's adjacent edges, excluding the edge of $e_{i\to j}$, is active. It is false otherwise. Thus, for every half-edge $e_{i\to j}$ in M,

$$-|\mathcal{E}_j \setminus \{i \to j\}| + 1 \leq \sum E_x - |\mathcal{E}_j \setminus \{i \to j\}|\nu_{i\to j} \leq 0 \quad (9)$$

with $x \in \mathcal{E}_j \setminus \{i \to j\}$, where $\mathcal{E}_j$ is the set of edges adjacent to v_j.

Deadends can then be avoided by the following constraints: for every half-edge $e_{i\to j}$ in M,

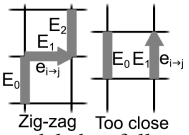
$$E_{i\to j} - \nu_{i\to j} \leq 0. \quad (10)$$

(ii) Branch avoidance: It is simple to avoid branches (i.e., a vertex with more than three adjacent active edges) in a network by the following constraint: For every vertex in M, v_y,

$$\sum E_x \leq 2, \quad (11)$$

where $x \in \mathcal{E}_y$ and $\mathcal{E}_y$ are the set of edges adjacent to v_y. Enabling the branch avoidance constraint leads to cycle-like networks.

(iii) Zig-zag avoidance: As shown in the inset, for each half-edge, $e_{i\to j}$, we identify two undesirable configurations. The presence of each configuration on $e_{i\to j}$ is denoted by a Boolean indicator variable, $Z_{i\to j}^k$, where k equals 0 for zig-zags and 1 for edges that are too close to each other. This is modeled as follows: For every $Z_{i\to j}^k$ that denotes the presence of the k-th undesirable configuration on half-edge $e_{i\to j}$,



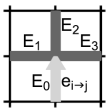
$$0 \leq \sum E_x - |E_x|Z_{i\to j}^k \leq |E_x| - 1 \quad (12)$$

with $x \in \mathcal{E}_{i\to j,k}$, where $\mathcal{E}_{i\to j,k}$ denotes the set of edges comprising the k-th undesirable configuration on $e_{i\to j}$. To forbid the presence of any of such configurations, we simply force all $Z_{i\to j}^k$ to be false. As this constraint can be too strict, we can instead minimize the occurrence of such configurations by adding the weighted summation of $Z_{i\to j}^k$ to the objective function.

(iv) T-junction avoidance: We can disallow or minimize the occurrence of T-junctions. For each half-edge pointing to a valence-4 vertex, $e_{i\to j}$, the presence of a T-junction on $e_{i\to j}$ is denoted by a Boolean indicator variable, $T_{i\to j}$, modeled as: For every half-edge $e_{i\to j} \in M$,

$$0 \leq E_1 + E_2 + E_3 + (1 - E_0) - 4T_{i\to j} \leq 3, \quad (13)$$

where E_0 is the edge indicator variable of $e_{i\to j}$'s edge, and E_1 to E_3 are the edge indicator variables of the other three edges adjacent to v_j (see inset). Again, we can forbid T-junctions by enforcing all $T_{i\to j}$ to be false, or by minimizing their occurrence by adding the weighted summation of $T_{i\to j}$ to the objective.



(v) User specifications: Users can explicitly specify certain combinations of vertices and/or edges to be inactive or active in a network. It is straightforward to impose these specifications by constraining the corresponding vertex or edge indicator variables to be true or false. Examples of such specifications can be seen in Section 5.

Extended objective function. The final IP takes the form

$$\min_{e_{i\to j}, D_{i\to j}, L_{i\to j;j\to k}, E_x, V_y, Z_{i\to j}^k, T_{i\to j}} \lambda_L \sum_x \Delta_x E_x + \lambda_D \sum_{i,j} D_{i\to j} + \sum_{k,i,j} \lambda_Z^k Z_{i\to j}^k + \lambda_T \sum_{i,j} T_{i,j} \quad (14)$$

subject to linear constraints in Equations 1 - 13 where Δ_x is the length of edge E_x, λ_L is the weight for the total length term, λ_D is the weight of the total travel distance term, λ_Z^k, $k = 0, 1$, are the weights of minimizing the occurrences of the two undesirable configurations, and λ_T is the weight of minimizing the occurrence of T-junctions. (Please note that some of the functional constraints are optional.) Figure 2 shows an example.

5 Applications

5.1 Urban street layouts

We aim to generate street layouts at small and medium scales. Based on the hierarchical nature of real-world road networks, we lay out the streets in four stages. First, we build a coarse network of *major roads* (e.g., freeways or arterial roads) that create a boundary of the target region and provides exterior traffic information. Second, for each region, we create a denser network of *collector roads* with the purpose of collecting traffic from the local roads. Third, grown from the collector roads, we develop an even denser network of *local roads* that roughly span the whole sub-region. Fourth, there may be *cul-de-sacs* grown around streets at the previous two levels.

We assume that the network of major roads is specified by the user. Afterwards, for each sub-region, street layouts are generated as follows (see Figure 3). First, we find a dense network of street segment candidates in the form of a semi-regular (i.e., most vertices are of valence 4) quad mesh, M, wherein the quads are roughly of uniform size and their shapes are close to a square. This assumption is based on the observation that real-world urban street layouts often favor 90-degree intersections. In practice, the input problem domain is quadrangulated by the patch-wise quadrangulation algorithm [Peng et al. 2014a]. Beginning at the sub-region level, we compute an initial street network by selecting a subset of segments of the dense network using the IP-based approach described in Section 4. The geometry of the street network (i.e., positions of the vertices along the street edges) is further improved by snake-based smoothing [Kass et al. 1988] (see Appendix). If the last level is

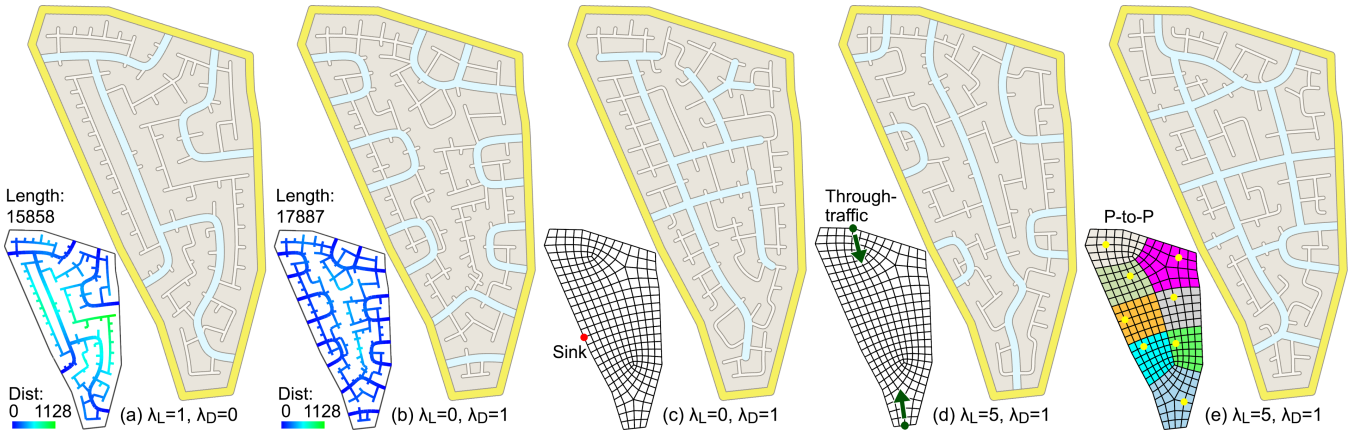


Figure 6: *Diverse street layouts resulting from different functional specifications. The first two layouts are optimized for (a) minimal network lengths and (b) minimal travel distances to the boundary, using different specifications of the optimization weights. The travel distances are shown in the bottom-left corners. (c) A layout with a single exit on the left. We also prefer a tree-like structure for this case, which is realized by allowing deadends on the second (collector roads) level. (d) A layout that encourages through-traffic in the vertical direction. This is realized by enforcing a shortest path connecting the two user-specified vertices (green) without inner branches on the second level. Note that through-traffic in other directions (e.g., horizontal) is implicitly discouraged. (e) A network that better supports interior-to-interior traffic, realized by the point-to-point constraint with a user-specified partition.*

not reached, we generate a denser network for the next lower level. In particular, we increase the resolution of the mesh by a Catmull-Clark subdivision scheme without vertex repositioning for greater degrees of freedom of the IP computation. We then generate the next level of roads working on the subdivided mesh. Figures 1, 6, and 7 show various street networks designed using our system.

In Figure 1a-c and Figure 6, we show how distinct street networks for the same sub-region can be created by different functional specifications. In Figure 7, we consider a practical scenario of designing a street layout for an empty section of land surrounded by existing streets.

We evaluated the functionality of the generated layouts using SUMO [Krajzewicz et al. 2012], as recommended by three urban planners and one of the authors [Marshall 2005] we consulted. The evaluation with SUMO is to demonstrate the correlation between design and practice, i.e., the predicted functional behavior closely matches the simulated one. The correlation indirectly justifies our choice of objective functions and functional constraints.

Design study. A professional urban planner used our tool to scope out a real-world scenario. The target was to redesign street networks to rejuvenate a very busy urban neighoborhood (see Figure 8). The exercise was a feasibility study to estimate the implications of converting a rail station into a residential area, possibly by moving the train network underground. A target density of the residential neighborhood was provided, which was used to set the density and coverage values for the functional specifications. Further, a traffic model of the dominant external traffic flow was created and had to be conformed to. Figure 8a shows the specified external traffic flow with arrows of the same color denoting source-destination pairs in the traffic demand model. Figure 8b shows the current street network. Our system generated different networks subject to different sets of specifications: 8c and 8d indicate scenarios by varying the balance between road length and travel distance to the sinks; 8e shows a setup with three boundary sinks (their locations were not explicitly specified); 8f encourages through-traffic by specifying a collector road from top to bottom; 8g shows the effect of two obstacles in the form of a park and man-made water body (see Figure 10 for parcelling result); and 8h shows a network favoring interior-to-interior traffic.

Feedback from our urban planner consultants was mostly positive. They commented that having such a tool would vastly improve the communication between urban planners and traffic engineers and that it would allow targeted evaluation of different functional scenarios (see Figure 8). They particularly liked the synthesized scenarios 8e, 8g, and 8h. A more systematic evaluation will be conducted in the future.

Evaluation. We measured the quality of the generated layouts by evaluating how well the specified functional specifications were met. The generated networks were all determined to be *valid*, i.e., free of islands and meeting the required coverage constraints. Certain quantities, e.g., length of the street network, were evaluated geometrically and directly measured from the network. The results are presented in black, in Figure 8.

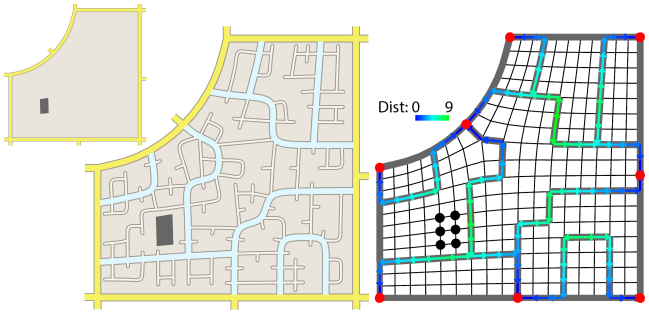


Figure 7: *Planning a street layout for an empty section of land surrounded by existing streets and with a historic site that should be preserved. To optimize travel times, instead of designating all boundary vertices as sinks, we place sinks only on the intersections of the existing streets (red vertices). The historic site is preserved by marking the corresponding vertices (black) as obstacles. On the right, we show the IP result of the level-2 roads. The distribution of the active half-edges indicate the shortest paths to the intersections (sinks) while the distance values encode the shortest distances.*

We made reasonable simplifying assumptions about the computation of travel time and trip assignment. We therefore can compare the average travel times and travel distances computed by our optimization framework with the output from the state-of-the-art traffic simulator, SUMO [Krajzewicz et al. 2012]. We tested the network only under low and medium level traffic, but not full congestion. In our settings, we used a traffic model of 250 cars/km^2 per hour with speed limits of 40 mph, 30 mph, 20 mph, 20 mph for the major, collector, local roads, and cul-de-sacs, respectively. We distributed the traffic uniformly across the interior region. In case of the design study (Figure 8), we additionally specified 1000 cars/hour uniformly distributed among the three main external sources/sinks. SUMO's measures are presented in green.

In all the cases, the functional specifications were satisfactorily met. For example, increasing total street lengths resulted in reduced average distance (to the boundary), or redistribution of through-traffic (see Figure 8b for original). Using SUMO simulation, we could also verify that Figure 8h has the shortest average travel distances among the six variations. Considering only interior-to-interior traffic, the second best scenario, 8d, has 9% and the worst scenario, 8b, has 44% longer travel distances. Overall, we observe that our optimization is very consistent with SUMO. Comparing the average travel times of the six different scenarios for trips from the interior to the boundary, we obtain the following results: (c) 20/17, (d) 18/16, (e) 50/43, (f) 20/17, (g) 19/16, and (h) 22/19 in the format SUMO/ours. While we systematically underestimated travel time, it was by an almost constant factor.

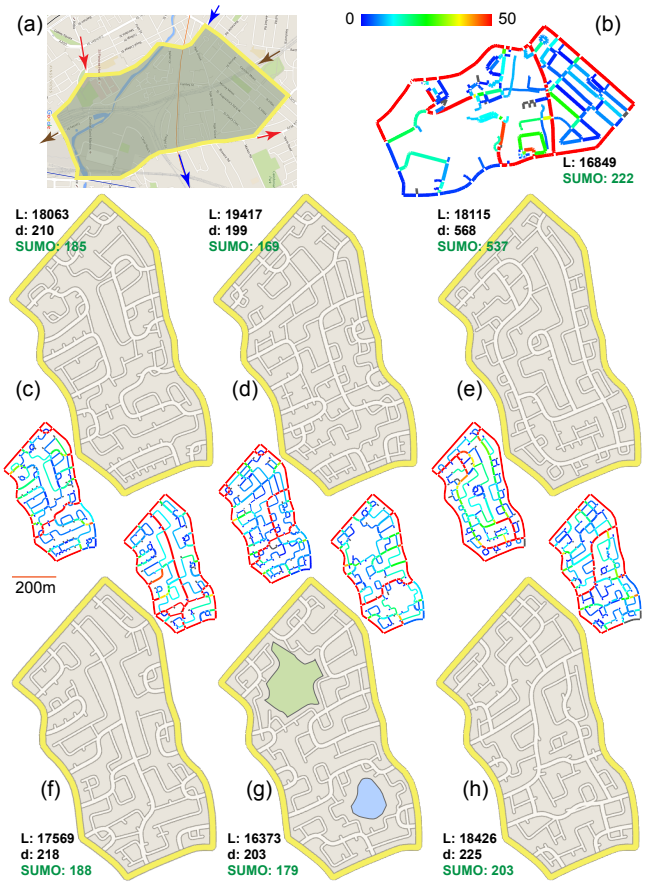


Figure 8: *Case study: Various scenarios created using functional specifications (see text for details). L denotes the total street length in meters, d the average travel distance for specified traffic demand, and SUMO indicates the average travel distance per the SUMO traffic simulator. The color-coded street network shows the traffic distribution over different road segments (gray indicates no traffic).*

5.2 Floorplans

We used our method to model *corridors* (i.e., the passage areas connecting rooms) in a building floorplan, as described next. We assume that the given building footprint is discretized into a quad mesh, M. A computed network represents the corridors for the building. The user first defines a set of room templates [Peng et al. 2014b] describing the admissible room shapes as combinations of squares, such as a 2x2 square room, a 3x2 long room, and an L-shaped room, with the possibility of having four non-valence vertices within (see Figure 9a top). We then enumerate the potential placements of the rooms on M, each placement being a connected set of faces on M (see Figure 9a bottom).

Our goal is to find a subset of all possible potential room placements such that no two overlap and together they fully cover M's faces. We denote the presence of each potential room placement in the subset as Boolean indicator variables, R_x, for $x \in [0, N_r)$ with N_r being the number of all potential placements. The coverage constraint is simply adapted as, for every face on M, f,

$$\sum_i R_i^{cover} = 1, \tag{15}$$

where R_i^{cover}, $i \in [0, X_6)$, denotes the set of indicator variables of the potential room placements that cover f. For faces denoted as *obstacles*, we change the right-hand side of the equation to zero.

In addition, a room has to be connected to the corridors (i.e., active edges of the network) and cannot have corridors in its interior, modeled as follows: for every potential room placement, R_x,

$$X_7 R_x - \sum_{0 \le i < X_7} (1 - E_i) \le 0, \tag{16}$$

where E_i, $i \in [0, X_7)$, denotes the set of indicator variables of the inner edges of R_x, and

$$R_x - \sum_{0 \le j < X_8} E_j \le 0, \tag{17}$$

where E_j, $j \in [0, X_8)$, denotes the set of indicator variables of the boundary edges of R_x. In summary, the IP formulation for floorplans comprises the original network IP formulation (see end of Section 4) with the coverage constraints (Equations 4 - 6) replaced by the room tiling constraints (Equations 15 - 17).

Our floorplans approach inherits all the functional specifications for modeling networks/corridors (see Section 5.1) except that the density aspect is now determined by the user-specified admissible room shapes. In addition, as the presence of rooms is explicitly expressed as a Boolean variable, users can precisely control the occurrences of each room type using linear constraints. In Figure 9b, we show several floor plans for an office building with distinct functions. In Figure 1d, we show a floor plan for a large facility.

5.3 Timing and Analysis

We implemented our algorithms using C++. We report timings on a desktop computer with a 2.4 GHz eight-core CPU and 8 GB memory. We used Gurobi [2014] to solve the IP problems. The timing statistics are shown in Table 1. In practice, as it is difficult to find a global optimum, we also accept sub-optimal solutions (fulfilling all hard constraints) computed within reasonable time limits.

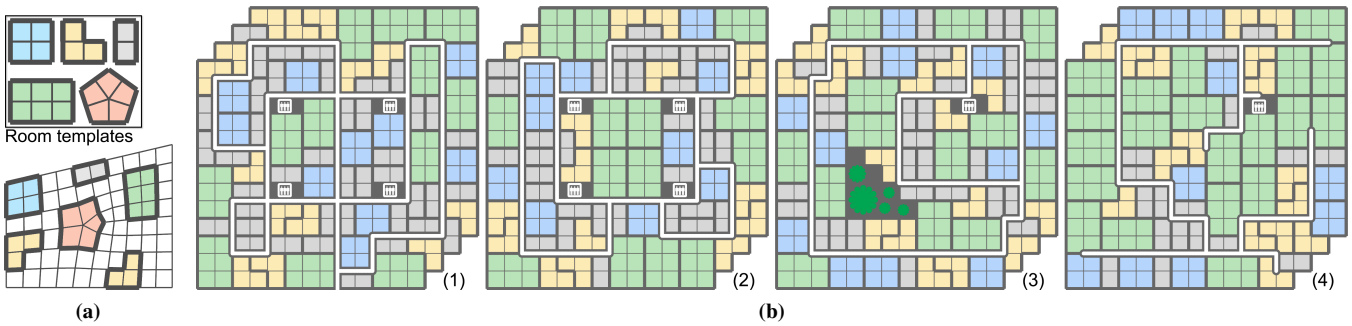


Figure 9: *(a) Top: a set of room templates. Bottom: several potential room placements on a mesh. (b) Floor plans for a four-story office building. Note that they consist of both the corridor network and the tiling with the room templates. (1) A floor plan with a single sink predefined as the building's entrance (bottom middle). The network is constrained to pass through the four elevator locations. Some faces are denoted as obstacles to be occupied by the elevators. (2) A floor plan sharing the same locations of the elevators (as sinks). (3) A floor plan that has a single sink and a large obstacle area for a roof garden. (4) Another floor plan with a single sink. Deadends are allowed in the network.*

The IP can become infeasible due to hard constraints. For example, it becomes infeasible if the coverage constraint requires that all vertices need to be covered and that too-close edges are forbidden. However, the Gurobi solver flags this as infeasible.

In practice, a real mid-sized layout plan easily takes several months due to the manual efforts of layout generation and the inevitable trial-and-error iterations caused by function test failures. Reducing the time to the order of minutes can significantly increase the number of design iterations.

Although the algorithm can be made to finish in seconds by removing the global constraints (like the L1 point-to-point constraint), such an approach is not attractive. Our main contribution is handling such global constraints that dictate behavior on the mid-scale.

Gurobi uses LP relaxation and branch-and-bound to solve the IP problem and can achieve optimality if the gap between the current objective value and the valid lower bound (from all the leaf nodes of the search tree) is negligible (0.01% in our implementation).

Limitations. The main limitation is the lack of interactivity. As it usually takes a few minutes to solve a medium-sized urban layout problem, our system is not interactive. Also, while we ensure validity of the networks, we may only achieve a local minimum with respect to the quality measures. A restriction of the IP formulation is that new additions/modifications should be linear; otherwise, the problem becomes too expensive to solve. Finally, we only model bidirectional roads and do not support one-way traffic in our current formulation.

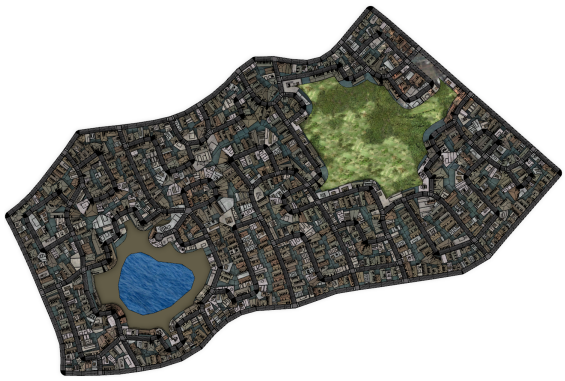


Figure 10: *Parcel and building layout using CityEngine based on street layout in Figure 8g. Note that since our algorithm ensures that the streets are appropriately spaced parcelling for the buildings remains a simple task without requiring any street modifications.*

Table 1: *For every example shown in the paper, we show the number of edges in the mesh, the parameters for the IP, and the times to obtain the shown solutions. inf (i.e., infinite) means the corresponding features is forbidden. Y means the feature is allowed and N means forbidden. For urban street layouts, the times to calculate the results of level-2, level-3, and level-4 are shown.*

	Mesh		$\lambda_L, \lambda_D,$	Dead			Time
	edge	vars	$\lambda_Z^0, \lambda_Z^1, \lambda_T$	end	Branch	PtP	(seconds)
Fig 1a	657	6092†	5 , 1 , 5 , inf , 0	N	Y	N	379/366/49
Fig 1b	657	7470†	0 , 1 , 5 , inf , 10‡	N	Y	Y	168/274/34
Fig 1c	657	6231†	1 , 0 , 5 , inf , 0	N	Y	N	456/467/-
Fig 1d	1012	11338	1 , 0 , 5 , inf , 0	N	Y	N	920
Fig 2a	312	2241	1 , 0* , 5 , inf , 0	N	Y	N	132
Fig 2b	312	2241	0 , 1 , 5 , inf , 0	N	Y	N	35
Fig 2c	312	2281	0 , 1 , 5 , inf , 0	N	Y	Y	11
Fig 2d	312	2766	0 , 1 , 5 , inf , 0	N	Y	N	24
Fig 2e	312	3865	1 , 0* , 5 , inf , 0	N	Y	N	74
Fig 2f	312	3865	0 , 1 , 5 , inf , 0	N	Y	N	56
Fig 2g	312	4751	0 , 1 , 5 , inf , 0	N	Y	Y	26
Fig 3L3	1200	6960	1 , 0 , 5 , inf , 0	N	Y	N	86
Fig 3L4	1200	1930	1 , 0 , 5 , inf , 0	Y	Y	N	2
Fig 6a	535	4432	1 , 0* , inf , inf , 0	N	Y	N	121/218/23
Fig 6b	535	4432	0 , 1 , 5 , inf , 0	N	Y	N	174/318/76
Fig 6c	535	5304	0 , 1 , 5 , inf , 0	Y	Y	N	70/294/46
Fig 6d	535	5036	5 , 1 , 5 , inf , 0	N	Y	N	12/223/43
Fig 6e	535	5651	5 , 1 , 5 , inf , 0	N	Y	Y	73/195/114
Fig 7	543	4415	0 , 1 , 5 , inf , 0	N	Y	N	229/500/1
Fig 8c	662	6508†	5 , 1 , 5 , 100 , 0	N	Y	N	390/336/15
Fig 8d	662	6508†	1 , 5 , 5 , 100 , 0	N	Y	N	328/392/10
Fig 8e	662	6508†	5 , 1 , 5 , 100 , 0	N	Y	N	370/377/12
Fig 8f	662	4346†	5 , 1 , 5 , 100 , 0	N	Y	N	229/303/4
Fig 8g	662	4338†	5 , 1 , 5 , 10 , 0	N	Y	N	128/121/5
Fig 8h	662	7280†	5 , 1 , 5 , 100 , 0	N	Y	Y	359/372/8
Fig 9b1	520	7986	1 , 0 , 5 , inf , 0	N	Y	N	1018
Fig 9b2	520	7876	1 , 0 , 5 , inf , 0	N	Y	Y	188
Fig 9b3	520	7944	1 , 0 , 5 , inf , 0	N	Y	N	201
Fig 9b4	520	7944	1 , 0 , 5 , inf , 0	Y	Y	N	1507

*: 1e-4. †:L1. ‡:L4.

5.4 Comparisons with Other Approaches

In this section, we show the advantages in terms of performance of formulating network problems into IP form and solving them with a specialized IP solver (e.g., Gurobi).

Manual solution. We first compare our solutions to some trivial solutions created manually. In Figure 11, we manually create solutions to achieve the same optimization goals as in Figure 2a. Such solutions use more edges than our IP-based solution. We also attempted to create floorplan results by hand. We found that it is very challenging to create full room tilings manually, let alone simultaneously finding a valid network that satisfies the given constraints.

Stochastic search. For comparison, we implemented a stochastic search-based approach to solve the network problems. Beginning at a trivial solution (e.g., every edge is active), the approach iteratively performs the following types of operations to alter the current solution: (i) deleting a single edge, (ii) deleting a pair of adjacent edges, (iii) deleting a triple of consecutive edges, and (iv) adding a single edge. At each iteration, we enumerate all possible feasible operations, rank them according to the new objective values (the lower the better), and pick one to perform. We pick operations in a simulated annealing sense (i.e., the higher ranked, the greater chance to be picked, and such tendency becomes more absolute at each iteration). The approach stops when there is no feasible solution or when a time limit is reached. As shown in Figure 12, we find that such a stochastic approach cannot compete with the IP-based approach in terms of speed and result quality.

6 Conclusions and Future Work

We proposed an algorithm for the computational design of networks for layout computation, such as street networks and building floorplans. The user provides high-level functional specifications for the target problem domain, while our algorithm jointly realizes the connectivity and the detailed geometry of the network. While there is a considerable amount of work on using functional specifications for evaluating networks, to the best of our knowledge, this is the first attempt to synthesize these layouts *purely* based on functional specifications.

In future work, it would be interesting to consider multi-modal transportation networks (e.g., public transportations) for a richer variety of urban street layouts. We would also like to tackle other network design problems by our IP-based approach, such as the layouts of residential houses, automated warehouses, and electrical layouts. In addition, while the meshes used in this paper are all quadrilateral because of our target applications, new design problems may necessitate the need for more kinds of mesh tessellations, such as a hybrid of quad and triangle meshes.

Finally, it would be interesting to extend our framework for network synthesis to game layouts, layouts for virtual worlds, and large-scale urban planning.

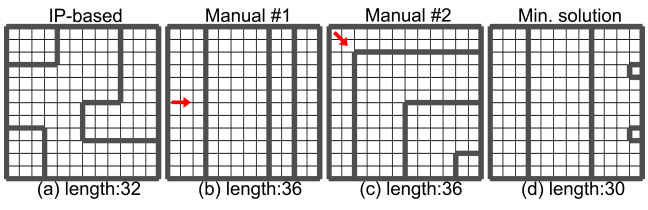


Figure 11: *(a) Our IP-based solution to find a network that cover the mesh (coverage range is two edges wide) with the fewest possible number of edges while avoiding zig-zags and edges that are too close. (b) and (c) Two manual results for comparisons. Our strategy is to start at one side of the boundary or a corner and grow edges as far as possible. Zig-zags and edges that are too close are avoided. (d) An optimal solution that allows zig-zags and edges that are too close, found by our IP approach.*

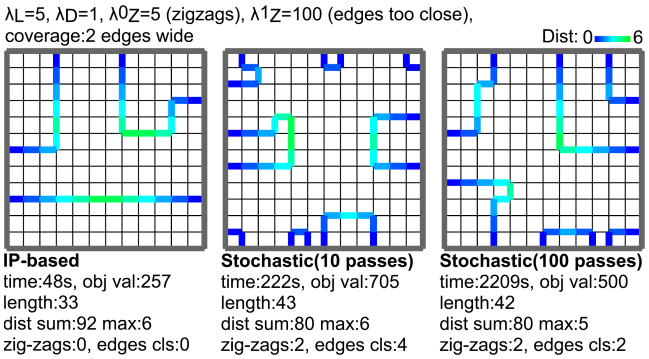


Figure 12: *Comparing our method to a stochastic search-based approach. We run multiple passes and pick the best solution. Even with a much longer time, the stochastic approach cannot find solutions of comparable qualities.*

Acknowledgement

We thank the reviewers for their comments and suggestions for improving the paper. Special thanks to Stephen Marshall and Benjamin Heydecker for sharing their knowledge, expertise, and experience regarding urban street network design and traffic planning, and to Carlos Molinero, Clmentine Cottineau and Elsa Arcaute for initial discussions. This work was supported in part by the ERC Starting Grant SmartGeometry (StG-2013-335373), Marie Curie CIG 303541, the National Science Foundation, Office of Sponsored Research (OSR) under Award No. OCRF-2014-CGR3-62140401, research funding from King Abdullah University of Science and Technology (KAUST), EPSRC Grant EP/K02339X/1 and EP/M023281/1, and National Natural Science Foundation of China (Nos. 61372168 and 61572502).

References

ALHALAWANI, S., AND MITRA, N. J. 2015. *Advances in Visual Computing: ISVC 2015, Part II*. ch. Congestion-Aware Warehouse Flow Analysis and Optimization, 702–711.

ALHALAWANI, S., YANG, Y.-L., LIU, H., AND MITRA, N. J. 2013. Interactive facades: Analysis and synthesis of semi-regular facades. *Computer Graphics Forum (Proc. EUROGRAPHICS) 32*, 2pt2, 215–224.

ALHALAWANI, S., YANG, Y.-L., WONKA, P., AND MITRA, N. J. 2014. What makes London work like London. *Computer Graphics Forum (Proc. SGP) 33*.

ALIAGA, D., VANEGAS, C., AND BENES, B. 2008. Interactive Example-Based Urban Layout Synthesis. *ACM TOG (Siggraph Asia) 27*, 5.

BAO, F., YAN, D.-M., MITRA, N. J., AND WONKA, P. 2013. Generating and exploring good building layouts. *ACM TOG (Siggraph) 32*, 4.

BASWANA, S., AND SEN, S. 2007. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Struct. Algorithms 30*, 4, 532–563.

BOARD, T. R. 2010. *Highway Capacity Manual*. Transportation Research Board.

CHEN, G., ESCH, G., WONKA, P., MÜLLER, P., AND ZHANG, E. 2008. Interactive procedural street modeling. *ACM TOG (Siggraph) 27*, 3, 103:1–9.

DEUSSEN, O., HANRAHAN, P., LINTERMANN, B., MĚCH, R., PHARR, M., AND PRUSINKIEWICZ, P. 1998. Realistic modeling and rendering of plant ecosystems. In *Proc. ACM SIGGRAPH*, 275–286.

GALIN, E., PEYTAVIE, A., GUÉRIN, E., AND BENES, B. 2011. Authoring hierarchical road networks. *Computer Graphics Forum 29*, 7, 2021–2030.

GARCIA-DORADO, I., ALIAGA, D. G., AND UKKUSURI, S. V. 2014. Designing large-scale interactive traffic animations for urban modeling. *Computer Graphics Forum (Proc. EUROGRAPHICS) 33*, 2, 411–420.

GÉNEVAUX, J.-D., GALIN, E., GUÉRIN, E., PEYTAVIE, A., AND BENEŠ, B. 2013. Terrain generation using procedural models based on hydrology. *ACM TOG (Siggraph) 32*, 4, 143:1–143:13.

GUROBI OPTIMIZATION, INC., 2014. Gurobi optimizer reference manual.

HANDY, S., PATERSON, R., AND BUTLER, K. 2003. Planning for street connectivity: Getting from here to there. In *Planning Advisory Service Report*.

KASS, M., WITKIN, A., AND TERZOPOULOS, D. 1988. Snakes: Active contour models. *Int. Journal of Computer Vision 1*, 4, 321–331.

KOSTER, A., KUTSCHKA, M., AND RAACK, C. 2010. Towards robust network design using integer linear programming techniques. In *Next Generation Internet (NGI), 2010 6th EURO-NF Conference on*, 1–8.

KRAJZEWICZ, D., ERDMANN, J., BEHRISCH, M., AND BIEKER, L. 2012. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements 5*, 3&4, 128–138.

LIU, H., YANG, Y.-L., ALHALAWANI, S., AND MITRA, N. J. 2013. Constraint-aware interior layout exploration for precast concrete-based buildings. *The Visual Computer (CGI Special Issue)*.

LUATHEP, P., SUMALEE, A., LAM, W. H., LI, Z.-C., AND LO, H. K. 2011. Global optimization method for mixed transportation network design problem: A mixed-integer linear programming approach. *Transportation Research Part B: Methodological 45*, 5, 808 – 827.

MA, C., VINING, N., LEFEBVRE, S., AND SHEFFER, A. 2014. Game level layout from design specification. *Computer Graphics Forum (Proc. EUROGRAPHICS) 33*, 2, 95–104.

MARÉCHAL, N., GUÉRIN, E., GALIN, E., MERILLOU, S., AND MRILLOU, N. 2010. Procedural generation of roads. *Computer Graphics Forum 29*, 2, 429–438.

MARSHALL, S. 2005. *Streets & Pattern*. Spon press, New York.

MATSIM, 2015. Matsim, http://www.matsim.org/.

MERRELL, P., SCHKUFZA, E., AND KOLTUN, V. 2010. Computer-generated residential building layouts. *ACM TOG (Siggraph Asia) 29*, 6, 181:1–181:12.

MEYER, M., AND MILLER, E. 2000. *Urban Transportation Planning*. McGraw-Hill.

MILLER, C. E., TUCKER, A. W., AND ZEMLIN, R. A. 1960. Integer programming formulation of traveling salesman problems. *J. ACM 7*, 4, 326–329.

NISHIDA, G., GARCIA-DORADO, I., AND ALIAGA, D. G. 2015. Example-driven procedural urban roads. *Computer Graphics Forum*, 1–14.

ORTZAR, J. D. D., AND WILLUMSEN, L. 2011. *Modelling Transport*. Wiley.

PARISH, Y. I. H., AND MÜLLER, P. 2001. Procedural modeling of cities. In *Proc. ACM SIGGRAPH*, 301–308.

PENG, C.-H., BARTON, M., JIANG, C., AND WONKA, P. 2014. Exploring quadrangulations. *ACM TOG 33*, 1, 12:1–12:13.

PENG, C.-H., YANG, Y.-L., AND WONKA, P. 2014. Computing layouts with deformable templates. *ACM TOG (Siggraph) 33*, 4, 99:1–99:11.

RUNIONS, A., FUHRER, M., LANE, B., FEDERL, P., ROLLAND-LAGAN, A.-G., AND PRUSINKIEWICZ, P. 2005. Modeling and visualization of leaf venation patterns. *ACM TOG (Siggraph) 24*, 3, 702–711.

SEWALL, J., WILKIE, D., MERRELL, P., AND LIN, M. 2010. Continuum Traffic Simulation. *Computer Graphics Forum (Proc. EUROGRAPHICS) 29*, 2, 439–448.

SEWALL, J., WILKIE, D., AND LIN, M. C. 2011. Interactive hybrid simulation of large-scale traffic. *ACM TOG (Siggraph Asia) 30*, 6.

SOUTHWORTH, M., AND BEN-JOSEPH, E. 2003. *Streets and the Shaping of Towns and Cities*. Island Press, Wasington DC.

VANEGAS, C. A., ALIAGA, D. G., BENEŠ, B., AND WADDELL, P. A. 2009. Interactive design of urban spaces using geometrical and behavioral modeling. *ACM TOG (Siggraph Asia) 28*, 5.

VANEGAS, C. A., GARCIA-DORADO, I., ALIAGA, D. G., BENES, B., AND WADDELL, P. 2012. Inverse design of urban procedural models. *ACM TOG (Siggraph Asia) 31*, 6, 168:1–168:11.

VISSIM, 2015. http://vision-traffic.ptvgroup.com/.

WARDROP, J. 1952. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers 1*, 2, 325–378.

WEBER, B., MÜLLER, P., WONKA, P., AND GROSS, M. H. 2009. Interactive geometric simulation of 4D cities. *Computer Graphics Forum (Proc. EUROGRAPHICS) 28*, 2, 481–492.

WILKIE, D., SEWALL, J., , AND LIN, M. C. 2013. Flow reconstruction for data-driven traffic animation. *ACM TOG (Siggraph) 32*, 4.

YANG, H., AND BELL, M. 1998. Models and algorithms for road network design: a review and some new developments. *Transport Reviews 18*, 3.

YANG, Y.-L., WANG, J., VOUGA, E., AND WONKA, P. 2013. Urban pattern: Layout design by hierarchical domain splitting. *ACM TOG (Siggraph Asia)*.

YU, L.-F., YEUNG, S.-K., TANG, C.-K., TERZOPOULOS, D., CHAN, T. F., AND OSHER, S. 2011. Make it home: Automatic optimization of furniture arrangement. *ACM TOG (Siggraph) 30*, 4, 86:1–86:11.

Appendix

Proof for Proposition 4.2. We now prove the requirement that forbids islands in networks. Assume a network contains one or more islands and the requirement is still met. Then, there exists at least one weakly connected component that is not connected to the sink vertices (i.e., an island). Within this island, there cannot exist any active half-edge that has no succeeding active half-edges; otherwise the requirement is immediately violated. Therefore, there must exist at least one loop of active half-edges in the island, denoted as $e_{0\to1}$, $e_{1\to2}$, ... $e_{n-1\to0}$, where n is the number of vertices in the loop. Since none of these vertices is a sink, $D_{0\to1} > D_{1\to2} > ... > D_{n-1\to0} > D_{0\to1}$, a contradiction.

Snake-based smoothing. We use the active contour model (snakes) [Kass et al. 1988] to smooth the coarse street networks generated by the IP approach. These networks tend to contain many sharp turns (e.g., stair-shaped) due to the nature of quad meshes. We give a summary of the algorithm as follows.

A snake is a distinct non-empty sequence of active edges that connects a distinct sequence of vertices (i.e., no branches or loops). Moreover, the valence of the first and last vertices of a snake cannot be 2; that is, a snake must end at non valence-2 vertices. We first decompose a street network into snakes. It is straightforward to see that a network can be decomposed into non-overlapping snakes that together fully cover the network. Snakes can include intersection vertices of the street network, i.e., an active vertex that connects to more than two active edges, from its interior. After the snakes are extracted, we subdivide each snake so that the smoothing algorithm has a higher degree of freedom.