

Route Packing: Geospatially-Accurate Visualization of Route Networks

Jieqiong Zhao
Purdue University
zhao413@purdue.edu

Morteza Karimzadeh
University of Colorado Boulder
karimzadeh@colorado.edu

Hanye Xu
Purdue University
hanye.vera.xu@gmail.com

Abish Malik
Davista Technologies
amalik@davistatechnologies.com

Shehzad Afzal
King Abdullah University
of Science and Technology
shehzad.afzal@kaust.edu.sa

Guizhen Wang
Purdue University
wang1908@purdue.edu

Niklas Elmqvist
University of Maryland College Park
elm@umd.edu

David S. Ebert
Purdue University
ebertd@purdue.edu

Abstract

We present route packing, a novel (geo)visualization technique for displaying several routes simultaneously on a geographic map while preserving the geospatial layout, identity, directionality, and volume of individual routes. The technique collects variable-width route lines side by side while minimizing crossings, encodes them with categorical colors, and decorates them with glyphs to show their directions. Furthermore, nodes representing sources and sinks use glyphs to indicate whether routes stop at the node or merely pass through it. We conducted a crowd-sourced user study investigating route tracing performance with road networks visualized using our route packing technique. Our findings highlight the visual parameters under which the technique yields optimal performance.

1. Introduction

Visualizing several routes on a geographic map while preserving the geospatial layout of the routes is a challenging task. Routes may overlap or cross each other, merge and depart again, share the same edge but have opposite directions, and have different semantics, speeds or volumes. Many state-of-the-art techniques tend to use visual or data grouping techniques—such as edge bundles [1, 2], flow map layouts [3], and metro maps [4, 5]—that not only cluster entities into groups but also simplify their exact movement paths into schematic maps. Based on the same principles as Harry Beck’s tube map from 1931 [6], these techniques reduce complex paths into straight lines and organize haphazardly scattered nodes onto a regular grid to facilitate comprehension and legibility.

But what if we are truly interested in the exact geospatial position and layout of a particular path? Consider a supply-chain network of multiple trucks transporting goods from a distribution center to a set of receiving sites on a daily basis, such as for a

fast food restaurant chain. What if we need to be able to distinguish individual routes, even when they temporarily merge together on the same physical road, only to separate again? What if we need to be able to follow a route that passes several nodes (i.e., restaurants) without stopping before reaching its destination? And what if we also need to preserve the temporal order and volume of goods being transported on each individual route? What if there is a high risk of flooding (or other kinds of risks) in the covered area, and the planners need to route the trucks while overlaying maps with flood (or other kinds of risk) maps to mitigate the delivery risk? Distorting the nodes and routes into schematic maps in such scenarios at best slows the human decision-maker down, and may lead to confusion or poor planning. These additional constraints conflict with the visual and data aggregation techniques reviewed above, highlighting that current techniques are ill-equipped to handle situations in which the geospatial layout provides indispensable geographic context.

Compared to schematic maps (e.g., metro maps [4, 5]), geospatially-accurate maps (henceforth referred to as geo-accurate) preserve the exact geographic location of entity paths and are thus more understandable to humans. For instance, London Underground publishes a geo-accurate metro map [7], particularly after Guo [8] estimated that up to 30% of London travelers take wrong routes with the schematic metro map. Geo-accurate maps can be particularly important for tasks where location is critical to decision-making. For example, supply-chain companies monitor road conditions of specific locations when a hurricane may block their delivery routes [9]. Similarly, Google Maps provides traffic and accident locations on the map (for a single route) to help users select a fast and safe route. Although geo-accurate maps are important in decision-making, research on how such maps can be best visualized remains limited, especially on visualizing several routes.

In this paper, we present a novel technique for geo-accurate visualization of routes called *route packing*

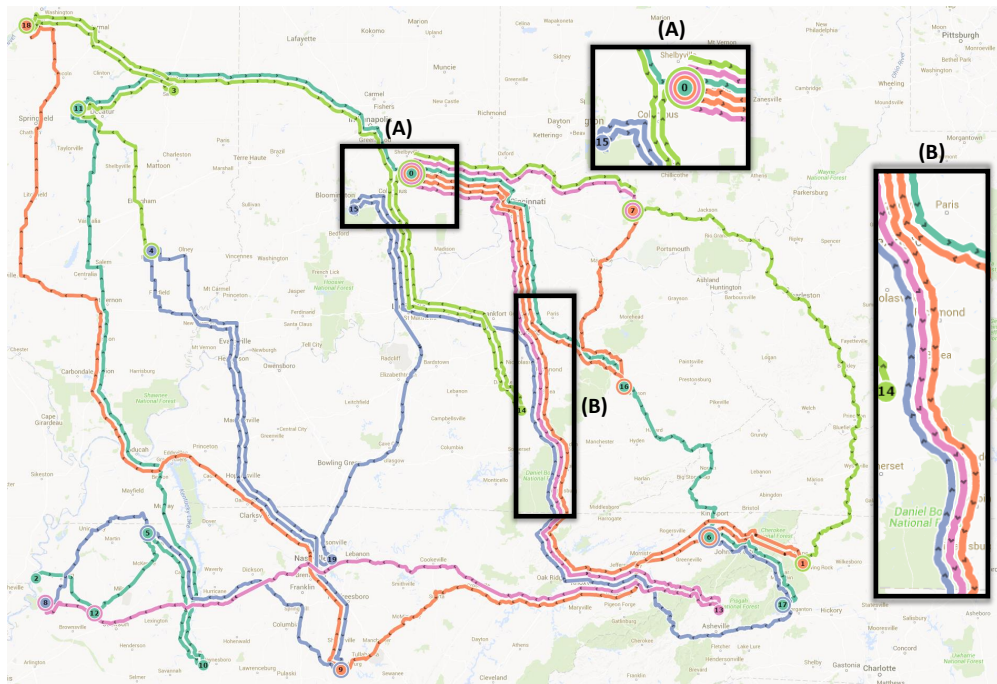


Figure 1. Route packing for a supply-chain network consisting of multiple individual routes (distinguished by categorical color encoding). Routes that share the same road for part or all of their stretch are packed rather than merged into a single line, preserving their identity. Glyphs on routes and on nodes indicate the direction of the route and whether they pass or stop at a specific node.

(Figure 1) that groups multiple routes sharing the same geographic space side-by-side without bundling them (while minimizing crossings), thereby preserving the identity of each individual route. While similar to metro maps [5], route packing does not simplify routes into straight or diagonal lines, but instead preserves their length, direction, and geospatial layout, making it useful for logistics planning and applications where geographic context is important. The route packing algorithm is sufficiently flexible to encode volume (such as amount of goods, traffic, or cost) into the width of the graphical line representing each route. To indicate whether a route stops at a specific node or merely passes it by, we use glyph decorations on nodes. Route packing is appropriate for visualizing 10-20 routes simultaneously (typical of supply-chain planning scenarios), enabling decision makers to compare the routes while preserving geographic context. Figure 1 presents an exemplar supply chain network from a fast food company, with five trucks delivering products to 20 stores, where each truck stops at 5–9 stores.

We also evaluate our route packing technique using a crowd-sourced user study involving more than 100 participants. Since no directly-comparable visualization technique exists in the literature, we opted to measure user performance using variations in route packing design. We applied the design choices proposed by

Holten et al. [10, 11] for visualizing directed node-link diagrams to route packing, comparing visual perception for arrows, tapered routes, and transparency. Our findings show that redundant encoding using both arrows and tapered lines resulted in the best performance for tracing routes in the network.

2. Related Work

Route networks can be visualized in three major ways [12]: When the geospatial position of vertices and edges is important, geo-accurate (or geo-referenced) networks are required, which can further be organized into traditional space-preserving vs. distortion-based representations. On the other hand, sometimes only the connectivity of the network is needed, in which general graph visualization is sufficient. Metro maps are a hybrid approach in which the network is laid out to maximize legibility, yet maintain some relation to the geographic position of the vertices and edges. Below we review these topics in more detail.

To show multiple routes simultaneously in **geo-accurate maps**, many techniques have utilized entity clustering algorithms with flowlines, including Phan et al.’s flow map layout [3], Andrienko and Andrienko’s silhouette graphs [13], Hadlak et al.’s attributed hierarchical structures that change over time [14], and Bouvier and Oates’s staining with flow

arrows [15]. Cornel et al. recently presented composite flow maps [16], which extract flows based on the structure of route shapes but then abstract and combine flows as ribbons. Our route packing technique stays faithful to the geospatial layout of the route network (helping users perceive the length of routes), preserving the individual identity and direction of routes while minimizing displacement and crossings.

Distortion-based methods distort the geometries in geographic space to optimize legibility or visual search. Cartograms, where area and distances in a map are distorted based on a variable such as population, are the origin such techniques [17], leading to work such as *Mélange* space-folding [18], and the generalized transmogrification space distortion framework [19] for keeping visibility by distorting space.

Our route packing technique does not distort the geographic space of the map itself, but it does distort the geometric stretch of the routes in order to avoid occlusion. This is essentially similar to a static version of the *EdgeLens* [20], which provides an interaction to visually separate multiple edges around a node to optimize their visibility.

Modeling a route network as a graph is possible if only the connectivity of the network is significant. Current **graph layout** algorithms can be used to find optimal positions for the vertices and edges [21]. Some graph visualization techniques have focused on preserving the overall topology (e.g., [22]) or intelligent routing of edges [23, 24]. However, while these efforts fulfill part of our needs in visualizing route networks, they are not suitable for geo-accurate maps.

Similar to geospatial visualization, scale is a key challenge for graph visualization. Early solutions focused on clustering vertices into groups [25], but recent work has instead used edge bundling [1, 2] where edges are grouped together to reduce occlusion. Interestingly, current edge bundling approaches are reminiscent of the flow map and flowline layouts discussed above. However, clustering and bundling for graphs have the same drawback, which is that the identity of individual entities is not preserved.

Nevertheless, we draw on prior graph visualization research. Holten and van Wijk in 2009 presented an in-depth user study on edge tracing in node-link diagrams [11], and in 2011, extended the study to a wider set of visual representations [10]. Our user study in this paper is based on these two studies, and we also derive our visual representations for directionality from their design space exploration.

Metro map layouts usually use vertical, horizontal, and diagonal edge directions, but maintain the geographic relations between nodes (or stations) [12].

Wu et al. [26, 5] studied the visualization aspects of automatically generating metro maps from a geographic map and connectivity graph. While this kind of layout has does not respect the geographic locations of edges, there are overarching design ideas here that are worth adopting. For example, Shen and Ma [4] use map layouts to visualize multiple parallel paths in an adjacency matrix representation of a graph. Similarly, Alper et al. [27] provide a linearized view of their *LineSets* visualization using the familiar visual language of metro maps. Our route packing technique builds on these ideas; we juxtapose incident routes side by side without bundling them together into a single unit, and use metro station visual encodings to communicate route crossings and shared stops.

Excessive crossings in metro maps can cause visual clutter and confusion for the end-user. The metro-line crossing minimization (MLCM) problem was proposed and formulated by Benkert et al. [28] in 2007. Bekos et al. [29] studied the MLCM problem on the graph structure of a path or tree. Later, Asquith et al. [30] applied integer linear programming to solve the problem for general graphs. Nöllenburg [31] and Mink et al. [32, 33] applied their line ordering methods on embedded graphs with an obvious relationship between vertices and edges. Extending these design principles from the MLCM research, we adapted the prior lines placement algorithms to keep the relative order along the edges and change the order in vertices.

3. The Route Packing Technique

Route packing juxtaposes overlapping routes side by side to keep the identity of each route distinguishable while staying faithful to its geographic footprint. In this section, we describe the details of our approach.

3.1. Data Model

The data model for route network visualization consists of a georeferenced graph $G = \langle V, E \rangle$ modeling both the topology and the geographic position of the network as well as a set of routes $R = \{r_0, r_1, \dots, r_n\}$. A route $r_i = \langle V_i, E_i \rangle$ is a list of vertices $V_i = (v_{i,0}, v_{i,1}, \dots, v_{i,n})$ representing the stops along the way, as well as a list of edges $E_i = (e_{i,0}, e_{i,1}, \dots, e_{i,n-1})$ representing the path between them.

Unlike some road networks that model only the major cities in the network using vertices, the georeferenced graph uses vertices with specific geographic positions to model the geometry of the road. Figure 2 shows an example of a small route network with major nodes A, B, \dots, G , and many minor vertices modeling waypoints along the roads. Since the edge list in a route r_i contains sufficient information to determine

its geometry, the vertex list in the route is only used to determine which of the vertices along the route are stopping points. We call each consecutive pair of vertices in a route a *leg*. For example, the highlighted route in Figure 2 consists of the legs (A, B) and (B, F) .

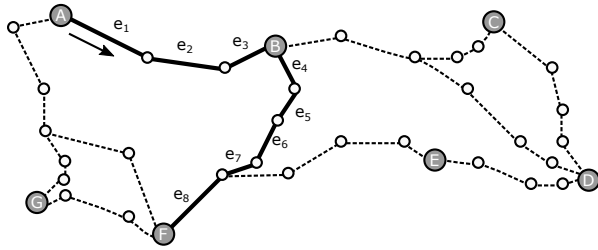


Figure 2. Example of a route in a route network. The highlighted route consists of two legs— A to B , and B to F —where the route stops temporarily, as well as the edges e_1, \dots, e_8 making up the actual path.

3.2. Design Rationale

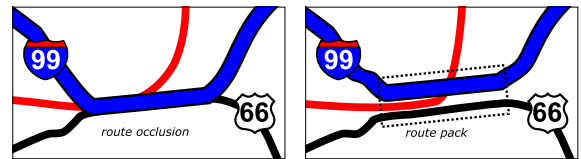
Given this data model, we formalize our design requirements for the route packing technique as follows:

- R1 **Preserve route edges:** All edges in a route network should be preserved so that they can be distinguished from other edges and matched to routes.
- R2 **Preserve route legs:** Legs along a route should be distinguishable so that endpoints can be identified.
- R3 **Convey direction:** The direction of a particular route should be conveyed across its entire path.
- R4 [Optional] **Convey volume:** Legs can optionally be associated with a weight representing volume, e.g. the amount of goods being transported.

The challenge for virtually all of these requirements is that a typical multi-route network contains a significant amount of overlap. For R1 (preservation of edges), the problem is that many routes will be sharing the same geographic stretch of a road for a partial (or full) leg. Sometimes the same stretch of road will be shared by two different routes heading in opposite directions, which challenges R3. For R3, routes may pass by certain vertices without stopping, whereas other routes do stop. Here, the challenge is to convey whether or not a particular route stops at a vertex.

3.3. Visual Design: Route Layout

Since different routes may share the same stretch of a road, directly visualizing different routes can generate severe visual confusion and hinder effective recognition of routes. For example, in Figure 3(a), the visual overlapping of the three routes makes it hard for a



(a) Occlusion of edges in route. (b) Route packing to preserve edges.

Figure 3. Example of three routes in (a) with significant overlap in a shared part of the road, causing occlusion. In (b), route packing has been applied to separate edges so that they are distinguishable and identifiable. Edge crossings are sometimes inevitable.

viewer to see detailed geographical information about the occluded edges. Route packing shifts and packs route edges to minimize visual overlap, resulting in the packed layout in Figure 3(b). The layout algorithm consists of several steps:

1. **Detecting overlap:** Due to the high precision of GPS coordinates, small data errors in GIS databases, and specific terrain conditions such as varying road width, route edges do not necessarily have to share the exact coordinates to exhibit overlap when rendered on the screen. To detect overlapping edges, we apply linear Kernel Density Estimation (KDE) to build a route network skeleton that groups route edges that are in close geographic proximity. Route edges are overlapped if they share some part of the skeleton.
2. **Shifting and packing:** Once overlapping edges have been detected, we *shift* them (displace them from their original position) to eliminate the overlap, and then *pack* them (bring them together along the same path while respecting edge width) one by one (Figure 3(b)). The displacement should be minimal and perpendicular to the edge direction for consistency. When displacing one edge, another edge that was not previously overlapped may become occluded. Therefore, these two stages—detection and shift/pack—are iterative and must be repeated until no more overlap is detected. When two edges overlap, only one of them must be shifted and packed. We apply route crossing minimization to decide which edge should have priority over the others (more details in Section 3.6).
3. **Rendering:** We use a categorical color assignment based on the topology of the route network skeleton (from Step 1) to render adjacent routes with different colors. Additionally, we use a white halo around each route line to visually separate packed routes.

3.4. Visual Design: Route Direction

Route Direction: Visualizing route direction (R3) is challenging due to limited space and potential for visual clutter. Inspired by Holten et al. [10, 11], we propose three alternative designs for conveying direction:

- **Arrow glyph:** An arrow is an intuitive representation to encode direction. We place multiple arrow glyphs at regular intervals along the entire route to fulfill R3 (Figure 4(a)). The size of the arrow is scaled by the width of the route line to save space and reduce clutter.
- **Transparency gradient:** This method uses a gradient of increasing transparency to convey the directionality (Figure 4(b)). The gradient can be applied to an entire route, or repeated for each leg.
- **Tapered line:** This design uses a variable-width route line that starts out at full thickness on the start node and tapers to smaller thickness at the end (Figure 4(c)). As for the transparency gradient, this design can be applied globally for an entire route, or repeated over and over again for each leg on the route.

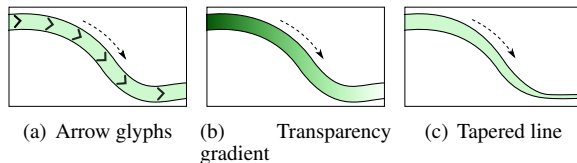


Figure 4. Three visual design alternatives for conveying route directionality.

While we considered using a color scale such as the red-to-green mapping proposed by Holten et al. [10, 11], we ultimately decided against this visual design because we already use categorical color to differentiate routes.

3.5. Visual Design: Nodes

Multiple routes may share the same locations as starting or stopping points; for example, in a supply-chain network, all routes tend to begin and end at a few distribution centers. This poses a challenge to effectively identifying and distinguishing those points of multiple routes (R2). For this reason, we propose three visual designs for communicating this information:

- **Concentric rings:** Inspired by metro maps [6], we overlay concentric rings with different colors on the node to indicate whether it is a waypoint for the corresponding color-coded route (Figure 5(a)).
- **Cookie bites:** Here, arrows pointing into the center of the node convey a stopping point; an arrow pointing out conveys the route leaving the node (Figure 5(b)).

- **Integrated arrows:** Similar to cookie bites, but the arrowheads are integrated into the route lines themselves to reduce visual clutter (Figure 5(c)).

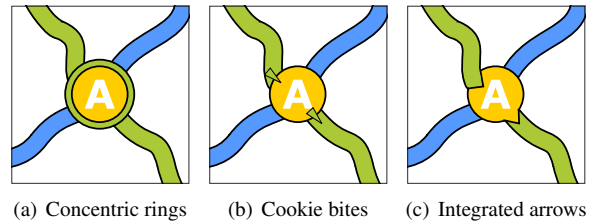


Figure 5. Design alternatives for nodes where the blue route (SW → NE) passes by a node, but where the green route (NW → SE) makes a stop.

Our current implementation uses concentric rings because of their ubiquity in metro and subway maps.

3.6. Implementation

Our route packing system is implemented using the web-based D3 [34] toolkit. Map and route network data are retrieved using the Bing Maps REST services¹. For each route, we query for the shortest driving distance and retrieve high-precision coordinates along the routes.

Bing Maps's outputs include thousands of sequential pairs of latitude and longitude. To simplify the raw graph data model, we first detect all overlapping routes (i.e., e4, e5, e6, e7 between leg (B, F) in Figure 2). For each shared route, we extracted its start point (i.e., B in Figure 2) and end point (i.e., vertex between e7 and e8 in Figure 2) as the crucial vertices to build a pruned graph. After that, we juxtapose shared routes side by side and minimize the number of crossings between routes (Algorithm 1). We preserve the relative shared routes displacement order within an edge between two crucial vertices and rearrange the displacement order for its consecutive edges if needed. In what follows, we elaborate on the details of our approach.

3.6.1. Detecting Shared Routes

We first detect shared sections of different routes, which may be occluded and hinder the effective recognition of routes (Figure 3). We use an approach based on linear KDE adopted from Lampe and Hauser [35] to extract the skeleton of the road network and identify groupings based on proximity. Figure 6(a) shows the result of applying linear KDE on the raw output from Bing Maps APIs. After converting to a binary image, we then apply the Zhang-Suen thinning algorithm to extract a skeleton of the route network [36] (Figure 6(b)). Finally, we apply a bifurcation detection

¹<https://msdn.microsoft.com/en-us/library/ff701713.aspx>

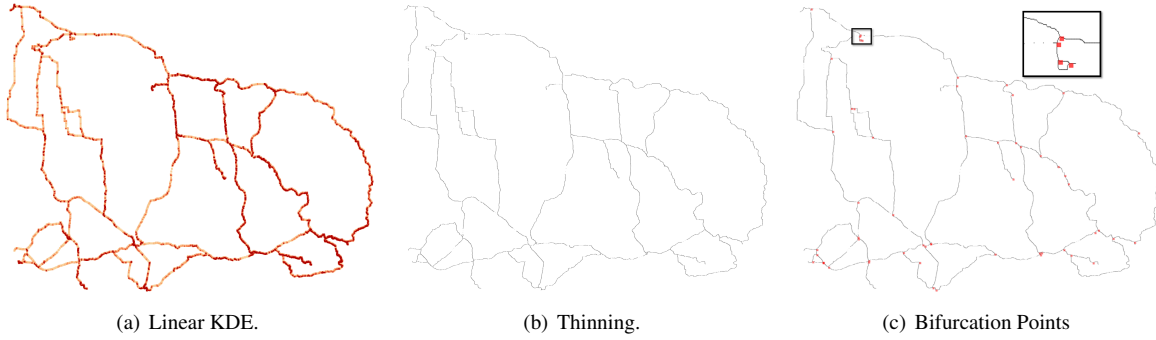


Figure 6. An illustration of applying linear KDE in (a) and then thinning algorithm to extract the skeleton of the road network in (b). In (c), red rectangles highlight the extracted results of bifurcation points.

algorithm [37] to identify intersections (Figure 6(c)) as crucial vertices to build a pruned graph.

Data: route network graph $G = \langle V, E \rangle$,
 routes: $R = \{r_0, r_1, \dots, r_n\}$, each route is composed by
 several route legs: $r_i = (l_{i,0}, l_{i,1}, l_{i,2}, \dots, l_{i,m})$, each route
 leg contains several vertices $l_{i,j} = (v_{i,k}, v_{i,k+1}, \dots, v_{i,p})$
Result: for each edge in a route, get its ranking of all route
 1 /*first round processing*/
 2 **foreach** two route legs l_i and l_j , $l_i \neq l_j$, computer their all
 longest common subpaths **do**
 3 **foreach** longest common subpath $lcs = \cap(l_i, l_j)$,
 (v_m, v_{m+1}, \dots, v_n) $\in lcs$ **do**
 4 $v_{cur} \leftarrow v_n$ last vertex in lcs ;
 5 $v_{pre} \leftarrow v_{n-1}$ last but one vertex;
 6 $v_{i,n+1} \leftarrow$ next vertex in l_i ;
 7 $v_{j,n+1} \leftarrow$ next vertex in l_j ;
 8 $angle_{base} \leftarrow$ angle of vector $(v_{pre} - v_{cur})$;
 9 $angle_i \leftarrow$ angle of vector $(v_{i,next} - v_{cur})$;
 10 $angle_j \leftarrow$ angle of vector $(v_{j,next} - v_{cur})$;
 11 $angleDiff_i \leftarrow angle_i - angle_{base}$;
 12 $angleDiff_j \leftarrow angle_j - angle_{base}$;
 13 **if** $angleDiff_i * angleDiff_j < 0$ **then**
 14 the route with negative $angleDiff$ win the
 comparison;
 15 **else**
 16 the route with larger $angleDiff$ win the
 comparison;
 17 **end**
 18 **foreach** edge $(e_m, e_{m+1}, \dots, e_n) \in lcs$ **do**
 19 record the comparison result;
 20 **end**
 21 **end**
 22 **end**
 23 /*second round processing*/
 24 **foreach** $e_i \in E$ **do**
 25 **foreach** route $\{r_0, r_1, \dots, r_n\} \in R$ **do**
 26 rank of r_i for $e_i \leftarrow$ how many times route i lost in
 comparison;
 27 **end**
 28 **end**

Algorithm 1. Minimize route crossings algorithm.

3.6.2. Shifting and Packing Overlapping Edges

After detecting shared routes, we obtain a graph of vertices (intersections and destinations) and intermediary connecting edges. Each vertex in the

graph is assigned a unique integer ID and each route leg as a sequence of vertices with these IDs. Two phases of processing are applied to obtain the proper ranking and the displacement of shared edges. To minimize crossings and reduce visual clutter, we adopted the MLCM algorithm in [31] and customized it for *Route Packing* to account for turns greater than 90 degrees.

We consider each vertex as an intersection of traffic either merging or diverging, and based on that, we determine the displacement order of different edges. This strategy has drawbacks when we draw an entire route along multiple edges, e.g. a large shift of displacement for two consecutive edges may generate extra curves. Our displacement algorithm can reduce such turbulence and minimizes crossings along the direction of a route: Algorithm 1 takes the reconstructed route network graph as input and generates the relative rank of each route. We generalize the comparison between two routes as diverging traffic.

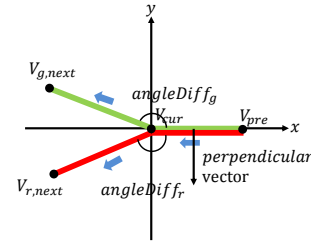


Figure 7. A generalized scenario of calculating the order of routes for a shared subpath.

Suppose two route legs share at least one edge. The displacement of routes on shared edges is decided by the angles where the routes leave the overlap subpath. Figure 7 demonstrates a visual example corresponding to pseudocode line 4–17 in Algorithm 1. First, we need to identify the longest common subpaths between all route legs. The displacement among the common subpath does not change, until two routes separate. As shown in Figure 7, we put the last vertex in shared

subpath on the center of the coordinate system. Then we align the second to last vertex in the shared subpath on the positive x Axis by subtracting the angle of vector $(V_{pre} - V_{cur})$. The range of an angles is $(-180, 180]$. The perpendicular vector is defined as counter-clockwise 90 degrees to the vector $(V_{cur} - V_{pre})$ along the moving direction. The red route tends to be top towards the direction of the perpendicular vector. After comparison of all shared subpaths, the rankings of all edges are determined. The ranking of a route for an edge towards its perpendicular vector direction is equal to how many time it lost in comparisons.

3.6.3. Transparency Gradient and Tapered Line

We use a gradient of increasing transparency to indicate route direction, where the transparency increases along the route from the starting point until the end point. For the current implementation, we set 35% and 100% as the minimum and maximum opacity. Similarly, for the tapered line, we utilize a variable-width route line that begins at maximum thickness at the start node, and tapers down to a minimum thickness by the end node. The width value is chosen to avoid unnecessary space usage while facilitating visual identification of the change in line width. In our implementation, we select 2 and 6 pixels as the minimum and maximum width.

4. User Study

We conducted a crowdsourced user study using Amazon Mechanical Turk to validate our design choices and determine the best directionality encoding for route packing (Table 1). Our participant pool were all crowdworkers. Route maps are common in daily life, and thus, it makes sense to engage a general population in the study. We recruited 132 (56 female) participants using Amazon Mechanical Turk with an age range between 20 to 68. Color blind users were excluded through self-reporting. Participants were paid \$0.80 and spent an average of 10 minutes on the experiment. Sessions were performed using a web browser on desktops, laptops, and a few tablets.

Table 1. Experiment setting.

Group ID	Representations	Participants
Group 1	Transparency Gradient, Arrow Glyph (TRA) Local Tapered Line, Arrow Glyph (LTA)	32
Group 2	Transparency Gradient (TR) Local Tapered Line (LT) Arrow Glyph (AG)	35
Group 3	Global Tapered Line, Arrow Glyph (GTA)	37
Group 4	Global Tapered Line (GT) Arrow Glyph, Concentric Ring (AGR)	28

4.1. Dataset and Task

Our experimental trials were designed by choosing different locations in the U.S. to generate random georeferenced graphs. To generate a georeferenced graph, we first randomly selected 10 nodes for a specified geospatial bounding box. To ensure similar task complexity, the 10 nodes were connected in an undirected graph. From these 10 nodes, we generated a total of 3 to 5 routes by randomly selecting 3 to 5 nodes (i.e., stops) to generate a connected route. We then retrieved a georeferenced subgraph for each route using the Bing Maps REST services API. Finally, we applied route packing on the resulting graph.

In the experiment, participants were shown a map visualization with two highlighted nodes. They were asked to determine whether the two nodes (i.e., stops) were connected through a route; that is, whether there existed a route that allowed them to travel from a given source to destination. We chose this task to ensure that our technique is able to visualize a route between two nodes effectively, even when multiple routes (sharing the same legs or crossing) are present. Jumping from one route to another route to reach the destination was explicitly forbidden, but there could be intermediate stops along the path between A and B, as shown in Figure 9(a). We emphasized the importance of directionality in the route network, but did not limit the number of intermediate nodes between stop A and B.

4.2. Experimental Factors and Design

The two experimental factors tested in the experiment are visual representations to convey directionality and levels of complexity. It is important to note that these two factors were tested in the context of multiple routes, and therefore, testing route packing in general. Firstly, We have three major designs for preserving route directionality: arrow glyph (Figure 8(a)), transparency gradient (Figure 8(b)), and tapered line (Figure 8(c)). The tapered line can be applied at two different scales: localized tapered line (stop to stop), and global tapered line (an entire route). Three different visual encoding strategies for indicating directionality were also evaluated: transparency gradient, local tapered line, and global tapered line. All these strategies can be applied alone or in combination with arrows that results in dual encoding of directionality. Beyond connectivity, we also studied the utility of the concentric ring design (Section 3.5). Table 1 shows the final set of representations we tested in the experiment. Secondly, the two complexity levels of questions tested in our experiment are: (a) stop A is directly connected with stop B, or (b) stop A is

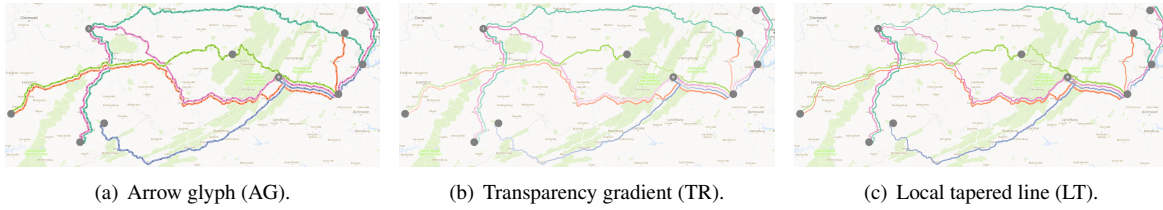


Figure 8. Three variants of route packing using different directionality methods.

connected with stop B through an intermediate node (Figure 9). This connectivity task is equivalent to that of Holten et al. [10, 11].

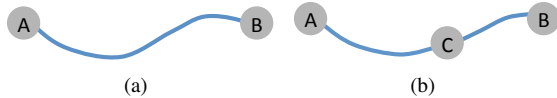


Figure 9. Complexity levels of user study questions.

To minimize the number of different visual representations for each participant and to keep each session under 10 minutes, we conducted the experiment in four groups. For each group, the participants were presented with two visual representations ($2 \times Vis$), and were given ten trials per representation: $5 \times two-stop$ (directly connected) routes, and $5 \times three-stop$ (one intermediate node) routes. This total of twenty trials in each group was generated using the same georeferenced graph data, but with different representation methods. We measured both accuracy and time spent on each trial.

After conducting the user study with 135 number of participants, we analyzed the correctness and completion times for different visual representations by averaging the results of all trials for the specific representation. We removed the data of 29 participants from our analysis as their performance was determined to be similar to chance (i.e., their results indicated that they had randomly clicked on any answer).

Table 2. Effects of factors on completion time.

Effect	Num OF	Den DF	F Value	Pr >F
Vis	7	2216	6.88	<.0001
Difficulty	1	2216	1.74	0.1867
Vis*Difficulty	7	2216	2.00	0.0519

4.3. Experiment Results

Figure 10(a) summarizes the correctness results for the eight different visual representations. The vertical axis shows the correctness ratio scaled from 0 to 1. We conducted a logistic regression with binomial distribution on the correctness ratings, and found no significant effect of visualization on correctness: $F(1,7) = 0.46, p = 0.8650$. This indicates that the different visual representations have similar correctness results (i.e., the different representations convey the information with similar accuracies).

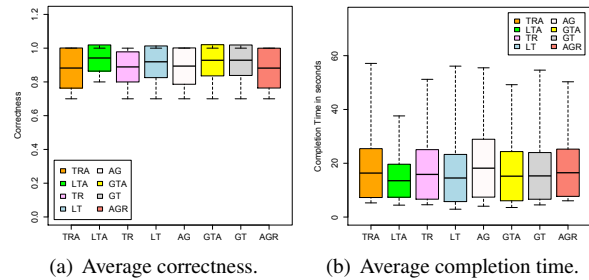


Figure 10. Correctness and completion time plots for eight route visual representations.

We measured the completion time for each representation by averaging across all trials. We analyzed the results using a repeated-measures analysis of variance (assumptions met). Visual representation had a significant effect on completion time: $F(1,7) = 6.88, p < .001$ (Table 2). Figure 10(b) shows average completion time as a function of the eight route visual representations. These two completion time results indicate that the different route visual representations have significant effect on completion time in our study.

The participants completed a post-test questionnaire for providing subjective feedback and overall preference on visualizations. 76 participants ($\approx 57.5\%$) preferred arrow glyphs (AG), followed by transparency (33 participants $\approx 25.0\%$), and tapered design (23 participants $\approx 17.5\%$). It is notable that participant preference is different from their performance. Participants achieved the highest speed with the tapered design (tapered < TR < AG). We compared the preference of individual designs to actual completion time. The participants who preferred AG spent 20.2 seconds on average, while the participants who preferred the tapered design spent 14.7 seconds on average. To sum up, AG might be easier to understand and more preferable due to familiarity, but the tapered design is a better choice in actual tasks for following directions of routes.

We summarize our findings from the user study as follows (Figure 11):

- Users spent less time solving trials using local tapered lines with arrows (LTA) than using transparency gradient with arrows (TRA) and arrow glyphs (AG).

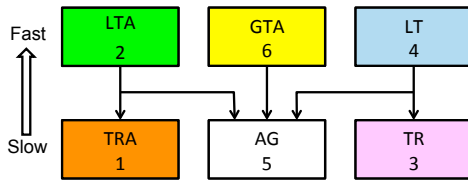


Figure 11. Pairwise relationships of visualization by completion time (Tukey HSD, $p < 0.05$).

- Users spent less time solving trials using local tapered lines (LT) than using transparency gradient (TR) and arrow glyphs (AG).
- Users spent less time solving trials using global tapered lines with arrows (GTA) than with using arrow glyphs (AG).
- There were no significant differences between arrow glyphs with concentric rings (AGR) and other visualizations.
- Overall, tapered representation performed the best.

4.4. Explaining the Results

The results from our study validate our design objectives that different route visual representations would garner different performances. We also find that design add-ons (arrows, concentric rings) do not have a major effect on completion time. These results provide guidance for the optimal design of route visualizations.

In Figure 11, we demonstrate the relationships between the 6 different route visual representations based on our posthoc analysis (significant to $p < 0.05$). We find that tapered line design is the best design compared to the other options because all of the techniques involving tapered designs—GRA, LTA, and LT—are superior to some other techniques even if there is no global order. More specifically, both LTA and LT have a shorter completion time compared to TRA, TR, and AG. In addition, GTA performed better than AG. We conclude that the tapered line design outperforms the other visualizations in conveying route directionality. This result is also consistent with the work by Holten et al. [10, 11], which found that a tapered design was optimal for visualizing directed edges in graphs. However, we find no significant difference in performance between the transparency design and routes with arrows. In addition, add-on arrows do not increase user performance with any of the types of visual design. Surprisingly, our analysis shows that there is no advantage to adding concentric rings at stops for conveying waypoints. A case study that shows the utilization of a tapered design and concentric rings for flight planning in the aviation domain is provided in the appendix².

²<https://docs.lib.purdue.edu/purvacsup/1>

4.5. Generalizing the Results

Our design is applicable to any closed route network for the visualization of 10-20 routes simultaneously. The number of routes is limited because the route packing technique consumes extra spaces to juxtapose routes while preserving their geographic information.

Furthermore, we conducted our study using Mechanical Turk. Our participants were diverse with different proficiency levels. This indicates that our participant pool is fairly representative of the general population. We also obtained consistent results from our participants, which is another indication that our findings generalize to other audiences. Though our subjects were non-experts, we expect that our results will hold with route planning experts.

Finally, the scalability of our techniques is difficult to assess given our evaluation results. The tasks were relatively small in scope; see Figures 8 for example images. To evaluate the effectiveness of conveying directionality, we only included 10 nodes (i.e., potential stops) in our route network trials. For larger and more dense route networks, route packing may cause too many routes to be packed in parallel, thereby consuming too much screen space. For instance, the packing strategy would fail if a dozen routes share the same tight U-turn on the map. Thus, additional filtering and zooming interactions need to be incorporated to support the analysis of larger networks.

5. Conclusion and Future Work

We presented a novel technique for visualizing routes on geographic maps called *route packing*, where the focus is to remain faithful to the identity, directionality, and weight of each individual route. To validate the utility of the technique, we also presented results from a controlled user study. The results point to the overall utility of the technique as well as the tradeoffs involved in the complexity and detail of the dataset to be visualized. In future research, we plan to apply route packing to a visual analytics tool for supply-chain logistics and simulation, as well as exploring its potential in other domains. Given that our current study in this paper was conducted with participants drawn from a general population, we are interested in determining how domain experts in supply-chain management may utilize our visualization. A comparative experiment against other techniques tackling route overlap is another possible direction that future research can undertake.

6. Acknowledgments

This work is funded in part by the U.S. Department of Homeland Security VACCINE Center under Award Number 2009-ST-061-CI0003.

References

- [1] D. Holten, "Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 741–748, 2006.
- [2] D. Holten and J. J. van Wijk, "Force-directed edge bundling for graph visualization," *Computer Graphics Forum*, vol. 28, no. 3, pp. 983–990, 2009.
- [3] D. Phan, L. Xiao, R. B. Yeh, P. Hanrahan, and T. Winograd, "Flow map layout," in *Proceedings of the IEEE Symposium on Information Visualization*, pp. 219–224, 2005.
- [4] Z. Shen and K.-L. Ma, "Path visualization for adjacency matrices," in *Proceedings of the EuroGraphics/IEEE VGTC European Symposium on Visualization*, pp. 83–90, 2007.
- [5] H.-Y. Wu, S. Takahashi, C.-C. Lin, and H.-C. Yen, "Travel-route-centered metro map layout and annotation," *Computer Graphics Forum*, vol. 31, no. 3, pp. 925–934, 2012.
- [6] A. Degani, "A tale of two maps: Analysis of the London underground "diagram"," *Ergonomics in Design*, vol. 21, no. 3, pp. 7–16, 2013.
- [7] J. Brownlee, "The London Underground finally publishes a geographically accurate Tube Map," <http://www.fastcodesign.com/3051293/the-london-underground-finally-publishes-a-geographically-accurate-tube-map>, 2015. Online; accessed 15 June 2019.
- [8] Z. Guo, "Mind the map! the impact of transit maps on path choice in public transit," *Transportation Research Part A: Policy and Practice*, vol. 45, no. 7, pp. 625–639, 2011.
- [9] S. Ko, J. Zhao, J. Xia, S. Afzal, X. Wang, G. Abram, N. Elmquist, L. Kne, D. V. Riper, K. P. Gaither, S. Kennedy, W. J. Tolone, W. Ribarsky, and D. S. Ebert, "VASA: Interactive computational steering of large asynchronous simulation pipelines for societal infrastructure," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1853–1862, 2014.
- [10] D. Holten, P. Isenberg, J. J. van Wijk, and J.-D. Fekete, "An extended evaluation of the readability of tapered, animated, and textured directed-edge representations in node-link graphs," in *Proceedings of the IEEE Pacific Symposium on Visualization*, pp. 195–202, 2011.
- [11] D. Holten and J. J. van Wijk, "A user study on visualizing directed edges in graphs," in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pp. 2299–2308, 2009.
- [12] M. Steiger, J. Bernard, T. May, and J. Kohlhammer, "A survey of direction-preserving layout strategies," in *Proceedings of the ACM Spring Conference on Computer Graphics*, pp. 21–28, 2014.
- [13] N. V. Andrienko and G. L. Andrienko, "Interactive visual tools to explore spatio-temporal variation," in *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pp. 417–420, 2004.
- [14] S. Hadlak, C. Tominski, H.-J. Schulz, and H. Schumann, "Visualization of attributed hierarchical structures in a spatiotemporal context," *International Journal of Geographical Information Science*, vol. 24, no. 10, pp. 1497–1513, 2010.
- [15] D. J. Bouvier and B. Oates, "Evacuation traces mini challenge award: Innovative trace visualization staining for information discovery," in *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pp. 219–220, 2008.
- [16] D. Cornel, A. Konev, B. Sadransky, Z. Horváth, A. Brambilla, I. Viola, and J. Waser, "Composite flow maps," *Computer Graphics Forum*, vol. 35, no. 3, pp. 461–470, 2016.
- [17] W. Tobler, "Cartograms and cartosplines," in *Proceedings of the Workshop on Automated Cartography and Epidemiology*, pp. 53–58, 1976.
- [18] N. Elmquist, Y. Riche, N. Henry, and J.-D. Fekete, "Mélange: Space folding for visual exploration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 3, pp. 468–483, 2010.
- [19] J. Brosz, M. A. Nacenta, R. Pusch, S. Carpendale, and C. Hurter, "Transmogrification: causal manipulation of visualizations," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, pp. 97–106, 2013.
- [20] N. Wong, M. S. T. Carpendale, and S. Greenberg, "EdgeLens: An interactive method for managing edge congestion in graphs," in *Proceedings of the IEEE Symposium on Information Visualization*, pp. 51–58, 2003.
- [21] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.
- [22] T. Dwyer, K. Marriott, and M. Wybrow, "Topology preserving constrained graph layout," in *Proceedings of the International Symposium on Graph Drawing*, vol. 5417 of *Lecture Notes in Computer Science*, pp. 230–241, 2008.
- [23] D. P. Dobkin, E. R. Gansner, E. Koutsofios, and S. C. North, "Implementing a general-purpose edge router," in *Proceedings of the International Symposium on Graph Drawing*, vol. 1353 of *Lecture Notes in Computer Science*, pp. 262–271, 1997.
- [24] T. Dwyer and L. Nachmanson, "Fast edge-routing for large graphs," in *Proceedings of the International Symposium on Graph Drawing*, vol. 5849 of *Lecture Notes in Computer Science*, pp. 147–158, 2009.
- [25] P. Eades and Q.-W. Feng, "Multilevel visualization of clustered graphs," in *Proceedings of the International Symposium on Graph Drawing*, vol. 1190 of *Lecture Notes in Computer Science*, pp. 101–112, 1996.
- [26] H.-Y. Wu, S. Takahashi, D. Hirono, M. Arikawa, C.-C. Lin, and H.-C. Yen, "Spatially efficient design of annotated metro maps," *Computer Graphics Forum*, vol. 32, no. 3, pp. 261–270, 2013.
- [27] B. Alper, N. H. Riche, G. Ramos, and M. Czerwinski, "Design study of LineSets, a novel set visualization technique," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2259–2267, 2011.
- [28] M. Benkert, M. Nöllenburg, T. Uno, and A. Wolff, "Minimizing intra-edge crossings in wiring diagrams and public transportation maps," in *Proceedings of Graph Drawing 2007 (LNCS 4372)*, pp. 270–281, Springer, 2007.
- [29] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis, "Line crossing minimization on metro maps," in *Proceedings of Graph Drawing 2008 (LNCS 4875)*, pp. 231–242, Springer, 2008.
- [30] M. Asquith, J. Gudmundsson, and D. Merrick, "An ILP for the metro-line crossing problem," in *Proceedings of the fourteenth symposium on Computing: the Australasian theory (CATS '08)*, vol. 77, pp. 49–56, 2008.
- [31] M. Nöllenburg, "An improved algorithm for the metro-line crossing minimization problem," in *Proceedings of Graph Drawing 2010 (LNCS 5849)*, pp. 381–392, Springer, 2010.
- [32] M. Fink and S. Pupyrev, "Metro-line crossing minimization: Hardness, approximations, and tractable cases," in *Proceedings of Graph Drawing 2013 (LNCS 8242)*, pp. 328–339, Springer, 2013.
- [33] M. Fink, S. Pupyrev, and A. Wolff, "Ordering metro lines by block crossings," *Journal of Graph Algorithms and Applications*, vol. 19, no. 1, pp. 111–153, 2015.
- [34] M. Bostock, V. Ogievetsky, and J. Heer, "D³: Data-driven documents," *IEEE Transactions on Visualization & Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.
- [35] O. D. Lampe and H. Hauser, "Interactive visualization of streaming data with kernel density estimation," in *Proceedings of the IEEE Pacific Visualization Symposium*, pp. 171–178, 2011.
- [36] T. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.
- [37] B. M. Mehre, "Fingerprint image analysis for automatic identification," *Machine Vision and Applications*, vol. 6, no. 2-3, pp. 124–139, 1993.