Path Selection: A Novel Interaction Technique for Mapping Applications

Michael Ludwig, Reid Priedhorsky, Loren Terveen
GroupLens Research, Department of Computer Science and Engineering
University of Minnesota, Minneapolis, Minnesota, USA
{mludwig, reid, terveen}@cs.umn.edu

ABSTRACT

Many online mapping applications let users define *routes*, perhaps for sharing a favorite bicycle commuting route or rating several contiguous city blocks. At the UI level, defining a route amounts to selecting a fairly large number of objects – the individual segments of roads and trails that make up the route. We present a novel interaction technique for this task called *path selection*. We implemented the technique and evaluated it experimentally, finding that adding path selection to a state-of-the-art technique for selecting individual objects reduced route definition time by about a factor of 2, reduced errors, and improved user satisfaction. Detailed analysis of the results showed path selection is most advantageous (a) for routes with long straight segments and (b) when objects that are optimal click targets also are visually attractive.

Author keywords
Path selection, selection techniques, bubble targets, bubble cursors, routing.

ACM classification keywords
H5.2. [User Interfaces]: Graphical user interfaces, interaction styles.

INTRODUCTION

Over the past few years, mapping applications have become popular on the World Wide Web. Systems like MapQuest and Google Maps let users search and browse within geographic regions. Many also allow users to define routes and share them with other users. This is particularly popular in recreational applications like bicycling, running, and hiking, where individual users have useful personal experience and are eager to share it [17].

Typically, systems that support route definition do so with an interface that lets users indicate a route by drawing it on top of the map. However, existing route definition techniques have several problems. First, nearly all of them are built on top of Google Maps or one of its peers. As we have pointed out previously [18], these systems have a major limit: users cannot directly interact with the actual geographic data. This precludes important functions such as tagging and rating geographic objects, which enable powerful features like personalized route finding. For the current purpose, the relevant point is that users create routes by drawing lines in a completely separate layer overlaid atop the "real" road segments; i.e., the only link between the routes and the roads is visual co-location.

The Cyclopath system (*http://cyclopath.org*) does let users interact with the geographic data; a user defines a route by selecting the edges within the transportation graph (i.e., road or trail blocks) that comprise the route. However, this raises a second, different problem: as Figure 1 shows, even short routes consist of many individual edges. This turns route definition into a tedious multiple selection task.

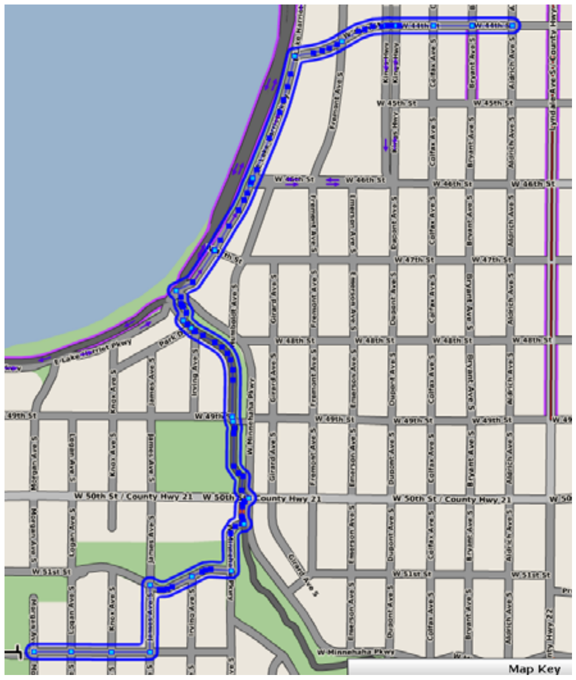

Figure 1: This short route – less than 2 miles – includes about 20 distinct edges of the transportation graph.

Our work addresses this problem. We designed and implemented *path selection*, a novel technique that dynamically computes the shortest path from the most recently selected node to the node nearest the cursor. Continuously updated visual feedback shows the shortest path, and when a user selects the next node, the entire path is added to the route. While developed for route selection, we believe this technique is useful for any application with graph data where users want to define paths through that graph. We empirically compared this technique to a state-of-the-art single selection technique, and found that adding path-finding results in faster route definition and greater user satisfaction. Detailed analysis identified structural properties of routes and nodes that determine how much path selection will help; for example, long straight segments within a route result in greater advantages.

The remainder of the paper is organized as follows. First, we survey related work. Next, we describe the context for our research and detail the selection tools we implemented. We then present the design of our experimental evaluation, show basic performance and user satisfaction results, and analyze the results in detail. Lastly, we identify areas for future research and conclude with a brief summary.

RELATED WORK

Mapping and routing applications

Many applications let users define and share routes through a road or trail map. A typical function is to let communities of users share knowledge that is valuable and hard to come by. For example, one could share a Google Maps tour of historical sites in Boston or a Bikely.com route of a safe bicycle commute into downtown. Counts & Smith created a research prototype aimed at sharing "recreational" routes, e.g., for runners, cyclists, or skiers[7]. Their system let users capture routes with GPS devices and upload route data to a map interface. As mentioned, in both the route-drawing and GPS-upload technique, the routes are weakly associated with the underlying geographic graph, linked only by visual co-location. We want routes to be defined directly in terms of the geographic data. This (a) requires exposing the geographic data to users for manipulation, and (b) transforms the process of route definition into a multiple selection task. We elaborate this point in our discussion of Cyclopath below.

Recently, Google Maps enhanced its automated route finder to allow a user to modify a route by moving a point on the route, causing the route finder to compute a new route from the start to destination that includes the new intermediate point. We compare this interesting technique to ours in the Discussion section below.

Single target selection

The problem of selecting single targets – e.g., objects in a drawing tool or items in a menu – is one of the most studied in HCI. Work typically is organized around Fitts' Law [9, 14]. The intuition of Fitts' Law is simple: it says that the time to acquire a target increases as the distance to the target increases, and decreases as the size of the target increases. Many research projects have sought to create improved selection techniques, seeking ways to either increase target size or decrease distance to the target.

To reduce the distance required to move the cursor to a target, researchers have invented techniques such as analyzing cursor motion to bring virtual proxies of targets towards the cursor [3] or jumping the cursor across empty spaces directly to potential targets [11]. However, neither of these techniques works well in interfaces that are dense with many potential targets – which is precisely what a map interface is, with targets like roads and points of interest often within a few dozen pixels of each other.

Increasing target size does not require changing the visual display. Instead, effective selection size can be manipulated by creating a larger activation zone around either targets [6, 15, 16, 20] or the cursor [12, 19]. Some techniques [e.g., 4] combine distance decreasing and size increasing. Again, however, these techniques tend to suffer in target-dense environments: as a user moves toward a goal target, intervening targets slow down the cursor.

Bubble cursors [10] are another technique for increasing cursor activation area. As a user moves the cursor, the system computes and displays a "bubble" that is centered around the cursor and envelops the nearest target. The area for any target is computed using Voronoi regions [1], which take into account the distance of that target to other nearby targets. The dynamic area computation makes this method work better in target-dense environments. Experiments showed significant performance benefits; however, bubble cursors do not address the problem that route selection requires clicking on many individual targets.

RESEARCH CONTEXT AND DESIGN RATIONALE

Cyclopath [18] is the context for our research. A web-based mapping application with an interface that works similarly to Google Maps, Cyclopath is targeted to the route finding needs of bicyclists. The critical difference is that Cyclopath is a *geowiki*, meaning that all data can be edited by users. Users can annotate and rate edges of the transportation graph (i.e., segments of road or trail between two nodes), and can edit – modify, delete, and add – the geometry and topology of the graph and attributes of its edges.

In Cyclopath, users do not simply draw lines to indicate a route, they *select objects* (nodes or edges) in the map. Users define routes for two main purposes:

- Mass rating or annotation of a sequence of edges, e.g., rating the whole sequence as "good" or adding a note about heavy traffic during morning rush hour.
- Sharing routes with other users, e.g., creating and sharing a favorite ride by the river (a planned feature but one that is not yet implemented).

As noted above, the simple 2-mile route shown in Figure 1 consists of about 20 individual edges; some routes can be considerably longer, making route definition techniques that require users to select dozens or hundreds of individual objects tedious.

When analyzing design possibilities for route selection tools, we considered two main approaches. The first was making it easy to select individual nodes; we looked to state-of-the-art work on selecting individual targets, specifically bubble cursors [10] and bubble targets [6].

The second approach was to reduce the number of nodes users had to select to define a path. We were inspired by Cyclopath's route-finding feature. After a user selects one node (the *anchor*) and then moves the cursor, the system continuously computes the shortest path from the anchor to the node nearest the cursor. Then, if the user clicks, the entire shortest path is added to the route (see Figure 2).

Taking the cross product of these two approaches yields a 2×2 design space, as shown in Table 1. While we believed that a tool that used both bubble targets and path-finding would work best, implementing all four designs let us quantify and compare the benefits of each of the pure techniques and how much they improve over a baseline. We next present design details for each of the four tools.

Bubble targets?	**Path-finding?**	
	No	***Yes***
No	**CTRL-CLICK**	**PATH**
Yes	**BUBBLES**	**BUBBLE-PATH**

Table 1: Design space for path selection tools

CTRL-CLICK: Standard multiple selection

For a simple baseline, we chose the technique for selecting multiple objects that is implemented in widely used applications like file browsers and drawing programs: clicking on an edge selects it. (Selected edges are surrounded by a thick blue outline in all tools.) Clicking on an edge with the CTRL key down toggles its selection state: if it was not selected, it is added to the set of selected edges, and if it was selected, it is removed.

This tool does not enforce route semantics. That is, a user can select non-contiguous edges and can select the edges on a route in any order. We did not expect this tool to perform well. Rather, it served as a state-of-the-practice baseline for comparison to the advanced tools.

BUBBLES: Bubble targets / cursors

The next tool facilitates the selection of individual targets, and is based on the bubble cursors and bubble targets techniques. While one might think we could simply modify CTRL-CLICK to use bubble targets (cursors), the properties of our mapping interface forced several revisions.

First and most important, we decided it would be visually awkward to draw bubbles around edges. Because edges have irregular shapes and varying lengths, bubbles of different sizes and shapes would be continually appearing, growing, shrinking, and disappearing. Thus, we decided to make nodes the targets of selection. To select an edge, a user must select both of the edge's nodes in turn. (Implementing bubble targets with edges as targets would be an interesting alternative; creating bubbles that are visually appealing and readily comprehensible certainly poses a challenging visual design problem.)

Second, making nodes the targets of selection requires using limited knowledge of the transportation graph to constrain selection. After selecting one node, the only valid targets are nodes directly reachable (via a single edge) from the just-selected node. Continually updated visual feedback (small orange circles) indicates valid targets.

Finally, we felt that warping the cursor would be visually confusing and unappealing in the target-dense environment of a mapping application. Therefore, we chose to draw bubble targets around the targeted node using the Voronoi method of Grossman [10] to compute the area of the bubble.

As the BUBBLES and CTRL-CLICK tools differ in several ways, the reason for any performance differences between the two will not be clear. However, testing both still furthers our research goals, because our primary aim is to compare the benefits derived from a state-of-the-research single target selection technique (BUBBLES) with our innovative path selection technique. CTRL-CLICK serves as a "sanity check" to verify that our advanced methods were improvements over a very simple technique.

PATH: Continuous path-finding and visual feedback

Like the BUBBLES tool, the PATH tool is node-based. However, it does not use bubbling; targets (nodes) have the same constant size in both visual and motor space. Instead, it uses continuous path-finding to let users short-circuit the process of selecting each of the many nodes in a typical route. Thus, the benefit of this approach is reducing the number of selections required to select an entire route.

At all times during the route selection process, the system computes the shortest path through the graph from the last selected node (the anchor) to the node nearest the cursor (when it is within 40 pixels). It does this using the A* search algorithm. When the user clicks on a node, the entire shortest path is added to the route.

The tool shows a preview of what selecting a particular node would do: the continuously-updated shortest path from the anchor to that node is indicated with a green highlight. We call this the *path extension*.

BUBBLE-PATH: Continuous path-finding + bubble targets

Our final tool simply combines the properties of the two previous tools: bubble targets plus path-finding.

Implementation details
Cyclopath is implemented as a Flex application, written in the ActionScript language and viewed using the Adobe Flash Player browser plugin. All the selection tools are implemented in ActionScript and run locally in the browser. Thus, even though Cyclopath is web-based, selection does not require fetching data from the server. This makes it possible for all the selection tools, even the ones that do path-finding, to operate without noticeable lag.

Preliminary analysis and terminology
As we have noted, the obvious advantage of path selection is that it can reduce the number of objects that must be selected to define a route. The route shown in Figure 2 below consists of 38 edges; however, with path-finding, only 7 selections are needed to select the entire route. We use the term *optimal* to denote the nodes that comprise the minimal set necessary to select a route. For a given route, we define the *optimal selection reduction ratio* (OSR ratio) as the total number of nodes in the route divided by the number of optimal nodes. Routes with many long straightaways, i.e., long sequences of contiguous edges that are more or less parallel to each other (see Figures 2 and 9 below), have high OSR ratios.

However, a high OSR ratio does not guarantee gains from using path selection. Path selection imposes a new perceptual/cognitive task: users must identify which nodes to select. They do this by moving the cursor to the vicinity of a candidate, then evaluating the path extension visual feedback to determine if (a) the path extension is on the desired route, and (b) if it advances the route "far enough" (an inherently subjective judgment). Thus, it is possible that identifying nodes to select is too much work; if so, path selection tools might actually be slower. Put another way: path selection suffers when optimal nodes are difficult to identify or visually "attractive" nodes are not optimal (we investigate below what makes a node attractive).

Therefore, we evaluate the four tools experimentally, using a number of routes with different characteristics.

EXPERIMENT
The subjects were students and staff at the University of Minnesota. We recruited subjects using relevant email lists, posters, and personal contacts. We ended up with 15 subjects, 8 women and 7 men ages 18 to 30. All subjects reported themselves to be daily computer users and occasional users of Web-based mapping applications. Subjects were given a $10 participation incentive.

The experiment was conducted on a 2.2Ghz MacBook Pro with a 15" built-in LCD display at 1440x900 resolution. Subjects used a mouse for input.

Design
The experiment was a within-subjects design. Each subject used all four of the selection tools. We defined 7 different routes for the experiment: 2 practice routes and 5 test routes for measuring performance[1]. The test routes vary in factors such as the number of long straightaways, number of corners, node density, etc. The overall experimental flow for each subject was:

- Introduction to the experiment.
- For each of the four selection tools:
  - Experimenter demonstrated the tool
  - Subject used the tool for 2 practice routes.
  - Subject asked clarification questions (if desired).
  - Subject used the tool for the 5 test routes.
  - Subject completed a user satisfaction survey for that tool. We selected (and adapted) a subset of questions from the QUIUS survey [5].

Each subject had a unique permutation of tool order and test route order. For example, one subject may have done the test routes in the order 3, 2, 5, 1, 4 (for all tools), and may have used the tools in the order BUBBLE, CTRL-CLICK, PATH, BUBBLE-PATH. Order of tools for subjects was controlled as follows: we generated all 24 possible permutations of the four tools, randomized the set, then assigned tool orderings to subjects according to this order.

Routes were indicated with a dark blue line through the relevant edges, with start and end nodes marked. Figure 2 is an annotated illustration of what a subject might see while using BUBBLE-PATH for Experimental Route 2. Route selection is complete when a subject selects every edge on the route, and no edge not on the route is selected.

All user interaction events were logged and timestamped. Panning and zooming were disabled to create a consistent environment for all subjects.

Evaluation metrics
To compare performance across the tools, we computed the following metrics:
- **Time** to select a route.
- **Errors**: clicks that selected either no node or a node not on the route.
- **Number of nodes** selected for a route; this is relevant only for the path selection tools.
- **Actual selection reduction ratio (ASR ratio)**: the number of nodes in a route divided by the (average) number of clicks subjects took to select that route. This is relevant only for path selection tools.

We derive several additional metrics from these basic ones, introducing them as appropriate below.

---

[1] In actual use, a desired route is known only to the user who is selecting it. However, for experimental purposes, we wanted all subjects to use the same routes.

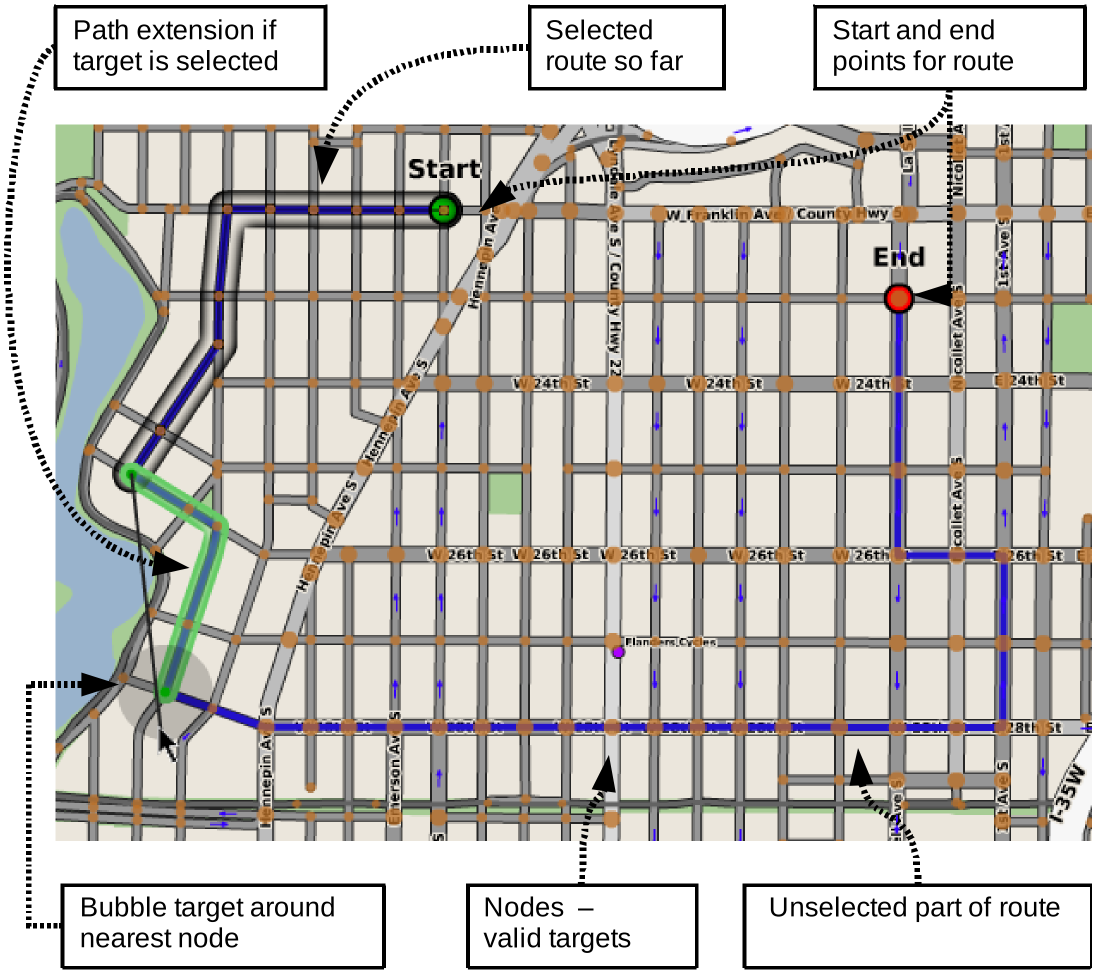
Path extension if target is selected
Selected route so far
Start and end points for route
Start
End
Bubble target around nearest node
Nodes – valid targets
Unselected part of route

Figure 2: The BUBBLE-PATH tool while in the process of selecting Test Route 2. Like BUBBLE-PATH, the BUBBLE tool uses bubble targets (but not path selection), and PATH uses path selection (but not bubble targets). (This figure is best viewed in color.)

RESULTS

We first present basic performance results, then describe findings of the user satisfaction survey. We follow with a detailed analysis that explains the performance results.

Performance: Time and errors

Figure 3 shows the average time subjects took to complete each task (test route) with each tool. There were significant differences among the tools for all routes (ANOVA; $p < .01$). Follow-up T-tests showed that all differences between pairs of tools for a route were significant, with three exceptions. For Routes 3 and 4, BUBBLES and PATH were indistinguishable, and for Route 5, BUBBLES and BUBBLE-PATH were indistinguishable.

The number of errors was low for all tools. However, ANOVA did show significant differences for each route ($p \leq 0.04$). As Figure 4 shows, CTRL-CLICK fared the worst. This is what we expected: selecting irregular and varying edge shapes is harder, and thus should be more error-prone. Follow-up T-tests showed that CTRL-CLICK was worse than all the other tools for routes 1, 3, and 5 ($p \leq 0.02$), and was worse than BUBBLE-PATH for all routes ($p < 0.01$). BUBBLE-PATH had the fewest errors for all routes except Route 5, but the differences between it and both BUBBLE and PATH were not significant, although there were strong trends for routes 2 and 3. BUBBLE and PATH were roughly comparable: except for Route 5, the average number of errors was similar. For Route 5, PATH was better than BUBBLE (T-test; $p = 0.01$).

We offer a caveat in interpreting these results. As discussed below, subjects made many more selections with BUBBLE than with path selection tools. Thus, the *proportion* of erroneous clicks made with BUBBLE always was lower than PATH, and usually was lower than BUBBLE-PATH. There is no obvious right measure – absolute or proportional number of errors – the whole point of path selection is to reduce the number of clicks needed to select a route. However, the proportional analysis confirms an obvious hunch: bubble targets make each selection act easier, and path selection does not. Therefore, selection errors should decrease for the tools that use bubble targets, BUBBLE and BUBBLE-PATH.

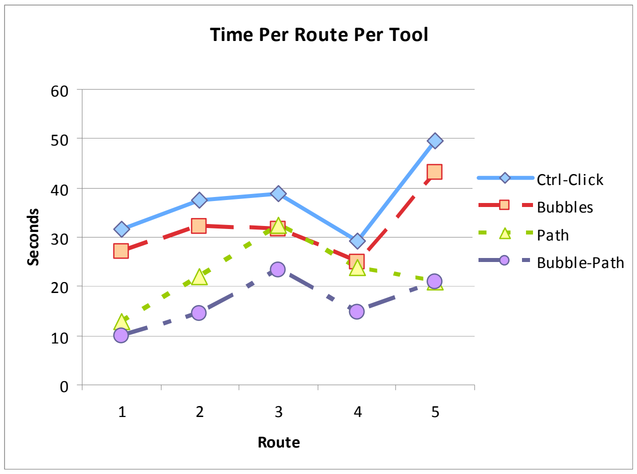

Figure 3: Task completion times for each route and tool

User satisfaction: Survey

Figure 5 summarizes the survey responses. There are several significant differences, as well as interesting trends. BUBBLE-PATH was always either significantly preferred over the other tools or statistically indistinguishable from them. One notable difference was seen for the "Overall: Terrible to Wonderful" question; an ANOVA showed significant differences ($p < 0.01$), and pairwise T-tests showed BUBBLE-PATH rated higher than all the other tools ($p < 0.01$). Another important difference was for the "Route Speed: Slow to Fast" question (ANOVA, $p < 0.01$), which measured how fast users perceived route construction to be. There was a trend favoring BUBBLE-PATH over PATH ($p = 0.057$), and BUBBLE-PATH was rated significantly higher than BUBBLE and CTRL-CLICK ($p < 0.01$). For the other Overall question, "Frustrating to Easy", there was a trend favoring BUBBLE-PATH over PATH ($p = 0.08$) and BUBBLE ($p = 0.09$), and all three of these tools were significantly higher than the baseline CTRL-CLICK ($p < 0.01$).

On the other hand, for questions that assessed ease of use, simplicity, and reliability, there were no significant differences, and BUBBLE-PATH and BUBBLE typically had nearly identical means, with PATH somewhat lower. We think

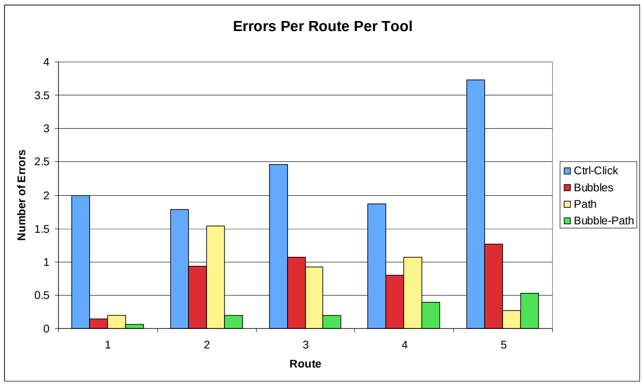

Figure 4: Errors for each route and tool

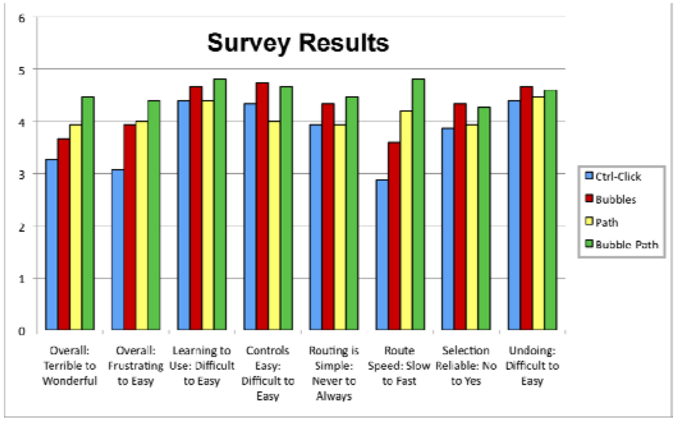

Figure 5: Survey results

there are two reasons for this. First, PATH presented the smallest selection targets to users, and thus an essential part of the task – moving the cursor into the target area – was difficult. Second, path selection adds an element of unpredictability to the selection task.

ANALYSIS: WHEN PATH SELECTION DOES (NOT) HELP

We wanted to understand in more detail the properties of routes that led to different levels of benefit for path selection: large advantages for PATH over BUBBLE on three routes, but no advantage on the other two routes. Our preliminary analysis led to three conjectures:

- The greater the actual selection reduction ratio, the greater the advantages of path selection.
- More long straightaways in a route means greater advantages for path selection.
- When the optimal nodes for a route are also the visually attractive nodes for that route, path selection will have greater advantages.

Path selection wins for routes that require fewer clicks

Data shown in Table 2 support the first conjecture. Table 2 shows the number of nodes in each route, the optimal (minimal) number of nodes that had to be selected by the path selection tools, and the actual average number of nodes selected during the experiment for both path selection tools.

Consider first the *potential* selection reduction ratio. Observe that Route 5 has the highest OSR ratio, and Route 4 has the lowest. As we would expect, Figure 3 shows that PATH had the biggest advantage over BUBBLE for Route 5, and no advantage for Route 4.

However, the potential reduction in selections doesn't tell the whole story. Most dramatically, Routes 1 and 3 had the same OSR ratio, yet Figure 3 shows that while PATH had a large advantage over BUBBLE for Route 1, the two tools were equivalent for Route 3. However, the *actual* selection ratios for the two routes differ dramatically: the ASR ratio for Route 1 was 4.7, and the ASR ratio for Route 3 was 2.3.

Task	Num Nodes	Optimal		Actual			
				PATH		*BUBBLE-PATH*	
		Sels.	Ratio	Sels.	Ratio	Sels.	Ratio
1	34	7	4.9	7.2	4.7	7.1	4.8
2	39	7	5.6	11.5	3.4	10.1	3.9
3	39	8	4.9	17.1	2.3	16.1	2.4
4	31	9	3.4	11.4	2.7	10.7	2.9
5	49	6	8.2	9.9	5.0	10.7	4.6

Table 2: Optimal and actual reductions in selection.

Long straightaways favor path selection

We next investigate the second conjecture: long straightaways increase the advantage of path selection. There are two reasons for this. First, the potential reduction in clicks is high. Second, we thought that this would be apparent to users: i.e., they would find it easy to identify the beginning and end nodes of the straightaway as appropriate targets to select. This is because these nodes are, by definition, "corners", and we believed that users will be likely to click on corner nodes.

We analyzed the relative time benefit of path selection as segment length (number of edges selected by a single click) increases. The intuition is that path selection should have greater benefits when users select a segment of length 8, for example, than one of length 3.

To compute this, we wanted to compare the time it took users to go "the same distance" using BUBBLE and BUBBLE-PATH, and see how the time varied with the distance. The following procedure formalizes this intuition:

```
o for every subject u and route r:
    ▪ for every selection made by u in route r with BUBBLE-PATH:
        ▪ get the start and end nodes, s and e, the segment length l, and the time $T_{BP}$
        ▪ compute the time it took subject u to go from s to e using BUBBLE; call this time $T_B$
        ▪ store the tuple (l, $T_B$-$T_{BP}$)
o for all stored tuples:
    ▪ for all values of l:
        ▪ compute the average of $T_B$-$T_{BP}$ for segments of that length
```

Figure 6 shows the results, graphing segment length against the time advantage of BUBBLE-PATH over BUBBLE (**$T_B - T_{BP}$**). The graph shows a linear relationship that supports our conjecture: the longer the segment, the greater the advantage of path selection. Further, the data point for segments of length 1 illustrates the overhead involved in path selection: it took users slightly more time to select a given segment of length 1 using BUBBLE-PATH than using BUBBLE. We believe that this is due to the perceptual and cognitive cost of identifying nodes to click on.

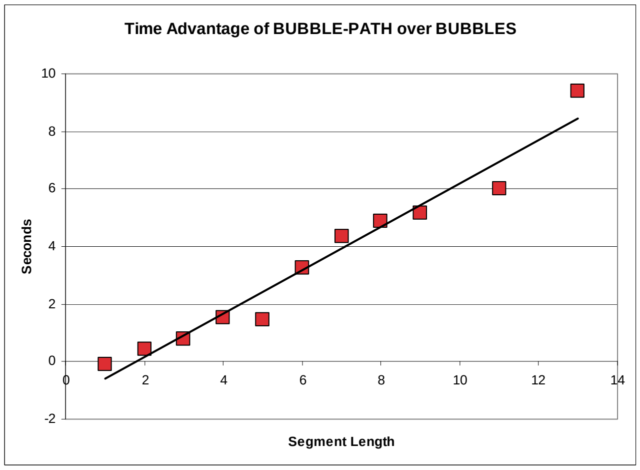

Figure 6: Longer segments increase benefits of path selection

A simple application of these results is that path selection should yield little or no advantage for routes with many short segments. Figure 7 supports this. For each route, it shows the proportion of segments (aggregated across all users) of different lengths. Routes 3 and 4 had the largest proportion of short segments, and the smallest proportion of long segments, and BUBBLE-PATH had the smallest time advantage over BUBBLE for these routes.[2] Routes 1, 2, and 5 had the most longer segments, and the fewest segments of length 1, and BUBBLE-PATH had the largest time advantage over BUBBLE for these routes.

What makes a node an attractive target?

The final conjecture is that path selection would not confer advantages when optimal nodes and attractive nodes do not align well. This situation would occur in routes with high *potential* selection reduction ratios, but low *actual* ratios. We identified several characteristics of a node that we thought might change the likelihood of being selected:

- **Angle** – the angle between the two edges on a route that impinge on a node.
- **Segment position** – this is a proxy for how far along in a straight segment a node is, formalized as the number of previous nodes with angle < 5º.
- **Bubble target area** – since the size of the bubble target for a node depends on how close other nodes are, this is a proxy for visual density. Intuitively, higher density makes it harder to distinguish an individual node and thus may decrease its selection probability.

[2] PATH had no advantage for these routes. In additional analysis, we found that subjects were very consistent in the nodes they selected with PATH and BUBBLE-PATH. Thus, we expect to see the same results if we do the analysis of Figure 7 with PATH instead of BUBBLE-PATH.

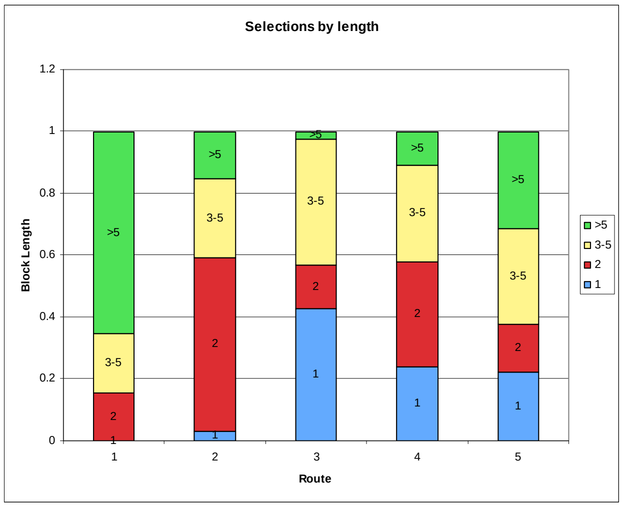

Figure 7: Segment lengths for each experimental route

We experimented with machine learning models that used these factors to predict whether a node would be selected. We divided all the nodes in all the routes – a total of 192 – into 3 categories based on how many subjects selected them using PATH.

- **Rare**: nodes selected by fewer than 5 (out of 15) subjects.
- **Sometimes**: nodes selected by at least 5 and no more than 8 subjects.
- **Frequent**: nodes selected by at least 9 subjects.

We tried out several of the learning algorithms included in the Weka software (*http://www.cs.waikato.ac.nz/ml/weka/*), frequently achieving over 90% classification accuracy. We then manually examined several of the best decision-tree rule sets. Surprisingly, these rule sets achieved such high accuracy using only the *angle* factor. After manually removing some redundant clauses, we ended up with a simple classification rule: category is **Rare** if angle $< 6^\circ$, category is **Sometimes** if $6^\circ \leq$ angle $\leq 33^\circ$, and category is **Frequent** otherwise.

This rule says that the more a node is "like a corner", the higher the probability of user acquisition. Classification accuracy for these rules is 98% (188/192); the confusion matrix is shown in Table 3.

	Classified as		
Actual category	Rare	Sometimes	Frequent
Rare	135	-	-
Sometimes	1	4	3
Frequent	-	-	49

Table 3: Confusion matrix for selection prediction rules

The relationship between angle and probability of acquisition is further illustrated in Figure 8.

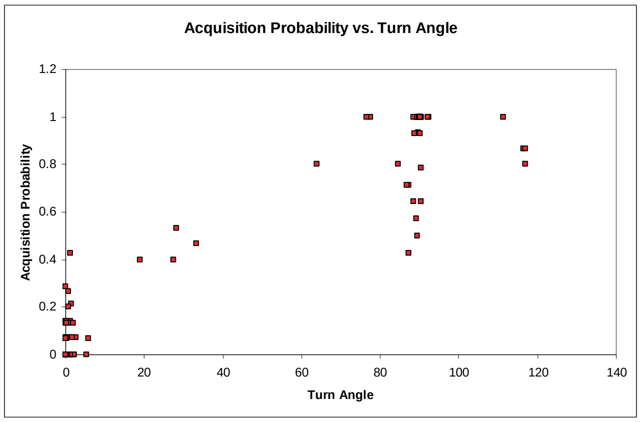

Figure 8: Effect of turn angle on probability of acquisition

The rules are exemplified by a visualization of the selection data from the PATH tool for Route 1 (Figure 9) and Route 3 (Figure 10). The key to the visualization is:

- A square is displayed for any node that any subject selected (with PATH).
- A thick black border indicates optimal nodes; a gray border indicates non-optimal nodes.
- The number in each square shows the proportion of subjects who clicked on that node.

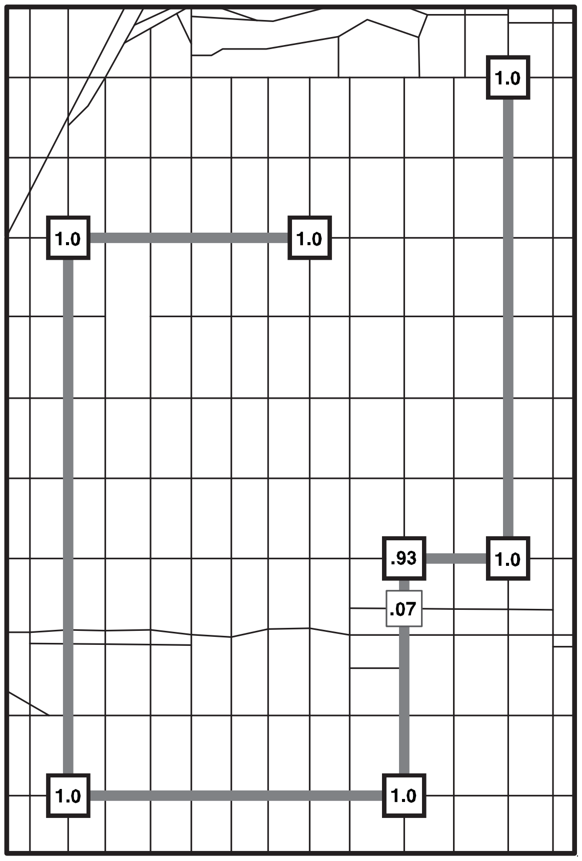

Figure 9: Selection data for Route 1

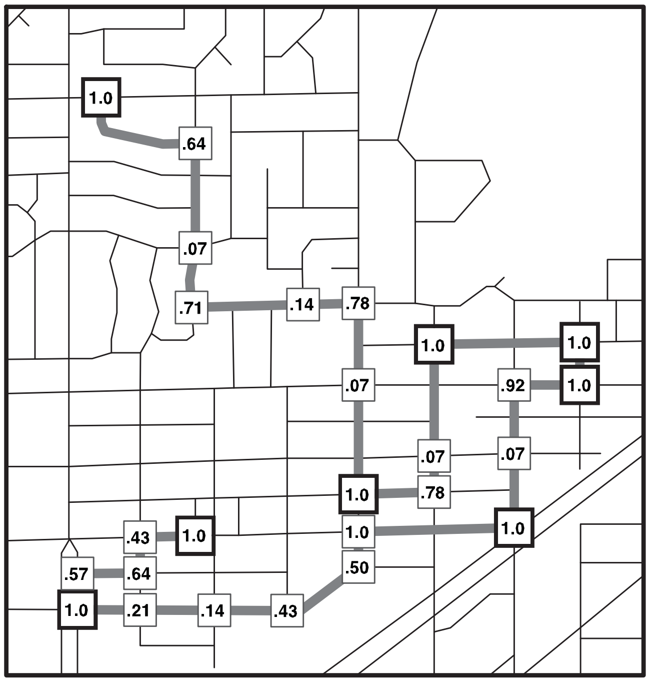

Figure 10: Selection data for Route 3

Route 1. Path selection had a large time advantage for this route. It has high OSR and ASR ratios, and it consists almost entirely of long straightaway segments. The optimal nodes are all corners, and all the corners are optimal.

Route 3. Path selection had no advantage for this route. It has a high OSR ratio, but a low ASR ratios. It contains many short segments. Even worse, while all the optimal nodes are corners, there are many corners that are not optimal, and the data confirm that these non-optimal corner nodes are frequently selected.

Discussion: Applications and future directions

Path selection can be applied directly in any mapping application that lets users define routes. Even if it applied only to mapping applications, it still would be a significant contribution, since such applications are important and increasingly common. However, we believe it is useful for any application where the data being manipulated form a graph, and users want to define paths through that graph. Such applications include biological networks (e.g., gene expression relationships), anatomical systems, architectural diagrams and floor plans, and flow charts.

An important point is that the A* algorithm we use to compute best path extensions gives us flexibility when moving to different domains. A parameter to the algorithm, the cost function, controls what path the algorithm computes; for our purposes, we use distance (shorter is better). However, if the best path in (say) a gene expression network is not based on distance but on the probability of a gene being realized, the only modification needed would be to define a cost function based on probability (or whatever would make the path extension best match the user's expectation).

Second, since in many applications a user's desired route is not known a priori, a promising approach is to dynamically predict what nodes a user is most likely to click on at any given time. We can use three factors to make such predictions: (a) history of selections by this and other users, (b) properties of edges other than distance that may make them likely to be added to a route (e.g., generic bikeability scores or user preferences), and (3) the direction of the user's mouse motion (for example, as done by [3]). Once likely nodes are identified, they can be made more visually attractive (e.g., by displaying an orange circle like we did in the BUBBLE tool) and easier to select (perhaps by letting their bubble target areas grow larger by stealing visual space from nearby, less-promising nodes, or just by making them larger in control space [4]).

Third, our analysis of the conditions when path selection works best suggests another experiment. We could test path selection with a set of routes that vary precisely along the factors we identified, e.g., turn angle of optimal and non-optimal nodes, length of straightaways, and visual density.

Fourth, we would like to compare path selection to Google Maps' "route-then-refine" technique. Google Maps lets users modify an automatically generated route from point A to point C by allowing a new point B to be added in the middle of the route. This creates a new route from A to C that is the composition of a route from A to B and a route from B to C. It is an open question how well this approach works in practice for domains such as bicycling and how it compares to path selection. Several factors may influence the relative success of the two techniques, and thus should be controlled in an experimental evaluation: (1) *Path length*. We think path selection is likely to work better for shorter paths. For longer paths, the picture is less clear, and is likely to depend on (2) *number of modifications required*. The work of refining a route in Google Maps is non-trivial: a user must select a part of the route to move, identify where to move it to, and then evaluate the new route. Further, these tasks may require panning and zooming, with multiple visits to particular map regions.

Finally, another potential technique for defining a path through a graph is simply to let users "draw" the path, i.e., move the cursor along the desired path without clicking objects. The system then could add edges to the route whenever a complete edge is traversed (or some similar criterion). This is rather like letting users "steer" along the desired path [1]. A step to assess the promise of this technique is to log use of our current path selection technique at the mouse motion – not just mouse click – level. This would let us see whether users typically do steer along a path or whether they "cut corners" to reach desired nodes. As with other potential techniques, making the design details right is challenging; for example, the conditions under which an edge is added to a route must reflect user expectations.

SUMMARY

We present a novel technique for selecting routes in a graph based on dynamic shortest-path computation and continuously updated visual feedback. The technique adds significant benefits beyond state-of-the-art-techniques for selecting single objects: faster route selection, fewer errors, and greater user satisfaction. Our analysis revealed what properties of routes make this new technique especially beneficial: long straight segments and a close alignment between optimal and visually attractive nodes.

ACKNOWLEDGEMENTS

We thank the people who participated in our experiment and the members of the GroupLens research group for their support and feedback. We also thank the NSF (grants IIS-0626930 and IIS-0534692) and the University of Minnesota Undergraduate Research Opportunities Program for supporting this research.

REFERENCES

1. Accot, J. and Zhai, S. Beyond Fitts' Law: Models for trajectory-based HCI tasks. In *Proc. CHI 1997.*
2. Aurenhammer, F. and Klein, R. *Voronoi Diagrams. Chapter 5,* in *Handbook of computational geometry,* J. Sack and J. Urrutia, Editors. North-Holland: Amsterdam, Netherlands. pp. 201-290, 2000.
3. Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., and Zierlinger, A. Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch- and pen-oriented systems. In *Proc. Interact 2003.*
4. Blanch, R., Guiard, Y., and Beaudouin-Lafon, M. (2004). Semantic pointing: Improving target acquisition with control-display ratio adaptation. In *Proc. CHI 2004.*
5. Chin, J. P., Diehl, V. A., and Norman, K. L.. Development of an instrument measuring user satisfaction of the human-computer interface. In *Proc. CHI 1988.*
6. Cockburn, A. and Firth, A. Improving the acquisition of small targets. *In Proc. British HCI Conference 2003.*
7. Counts, S. and Smith, M. Where were we: Communities for sharing space-time trails. In *Proc. GIS 2007.*
8. *http://cyclopath.org*
9. Fitts, P.M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology,* 47. p 381-391
10. Grossman, T. and Balakrishnan, R. The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area. In *Proc. CHI 2005.*
11. Guiard, Y., Blanch, R., and Beaudouin-Lafon, M. Object pointing: A complement to bitmap pointing in GUIs. In *Proc. Graphics Interface 2004.*
12. Kabbash, P. and Buxton, W. The "Prince" technique: Fitts' Law and selection using area cursors. In *Proc. CHI 1995.*
13. MacKenzie, S. Fitts' Law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7. p 91 – 139, 1992
14. MacKenzie, S. and Buxton, W. Extending Fitts' Law to two-dimensional tasks. In *Proc. CHI 1992.*
15. McGuffin, M. (2002). Fitts' Law and expanding targets: An experimental study and applications to user interface design, M.Sc. Thesis, Department of Computer Science, University of Toronto
16. McGuffin, M. and Balakrishnan, R. Acquisition of expanding targets. In *Proc. CHI 2002.*
17. Priedhorsky, R., Jordan, B., and Terveen, L. How a personalized geowiki can help bicyclists share information more effectively. In *Proc. WikiSym 2007.*
18. Priedhorsky, R. and Terveen, L. The computational geowiki: What, why, and how. In *Proc. CSCW 2008.*
19. Worden, A., Walker, N., Bharat, K., and Hudson, S. Making computers easier for older adults to use: Area cursors and sticky icons. In *Proc. CHI 1997.*
20. Zhai, S., Conversy, S., Beaudouin-Lafon, M., and Guiard, Y. Human on-line response to target expansion. In *Proc. CHI 2003.*