HomeFinder Revisited: Finding Ideal Homes with Reachability-Centric Multi-Criteria Decision Making

Di Weng[1], Heming Zhu[1], Jie Bao[2], Yu Zheng[2], Yingcai Wu[1]*
[1]State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China
[2]Urban Computing Group, Microsoft Research, Beijing, China
{dweng,zhmwoot}@zju.edu.cn,{jiebao,yuzheng}@microsoft.com,ycwu@zju.edu.cn

ABSTRACT

Finding an ideal home is a difficult and laborious process. One of the most crucial factors in this process is the reachability between the home location and the concerned points of interest, such as places of work and recreational facilities. However, such importance is unrecognized in the extant real estate systems. By characterizing user requirements and analytical tasks in the context of finding ideal homes, we designed ReACH, a novel visual analytics system that assists people in finding, evaluating, and choosing a home based on multiple criteria, including reachability. In addition, we developed an improved data-driven model for approximating reachability with massive taxi trajectories. This model enables users to interactively integrate their knowledge and preferences to make judicious and informed decisions. We show the improvements in our model by comparing the theoretical complexities with the prior study and demonstrate the usability and effectiveness of the proposed system with task-based evaluation.

ACM Classification Keywords

H.5.2. User Interfaces: Graphical user interfaces (GUI)

Author Keywords

Reachability; urban visual analytics; location selection.

INTRODUCTION

For many years, finding an ideal home has been a crucial but laborious task [38]. Home location is one of the foremost factors involved in such decisions [14, 37]. The quality of the home location generally depends on the reachability (i.e. accessibility) between the location and points of interest [11, 19, 22], such as places of work, schools, amusement parks, bus stops, and shopping malls. When finding an ideal home, a family with children may establish the following requirements based on their daily routines: (1) good elementary school that is a 15-minute drive away from home, (2) the father can drive his children to school and arrive at work within 40 minutes during the morning rush hour, and (3) the mother can fetch her child from school on the way home from her work within 30 minutes during the evening rush hour. Even with modern retrieval and interaction technologies, browsing through a list of filtered and ranked home candidates to find a competent one remains the most common yet time-consuming method to complete such task.

*Yingcai Wu is the corresponding author.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada
© 2018 ACM. ISBN 978-1-4503-5620-6/18/04. . . $15.00
DOI: https://doi.org/10.1145/3173574.3173821

Prior studies have extensively analyzed the location selection problems in the context of selecting retail stores [20], billboards [25], and ambulance stations [24]. Various location selection models were proposed, such as maximum coverage models [23, 25] and learning-to-rank model [15]. Nonetheless, choosing an optimal home location based on reachability remains unexplored and challenging. Williamson and Shneiderman [38] proposed a rudimentary dynamic queries interface, HomeFinder, allowing users to query real estate data with multiple criteria, including the physical distance between homes and places of work. Despite years of development, the modern online systems for finding ideal homes, such as Hubzu [2] and Zillow [50], exclude reachability queries or merely provide a limited set of reachability filtering options based on the physical distance. However, the distance cannot accurately estimate the reachability of locations in many cases because the traffic conditions may vary significantly over time along with several factors (e.g., weather conditions and traffic control).

The limited applicability of physical distance motivates us to develop a data-driven method that integrates accurate reachability estimations into the workflow of finding an ideal home. This method computes reachability from massive taxi trajectory data, which are relatively easy to acquire [42, 43] and effectively reveal underlying traffic patterns [9, 47]. Given a travel duration, the evidence-based reachability between two locations is measured as the number of trajectory-reachable days divided by the number of days available in the dataset [39]. Hence, each home location can be associated with a probabilistic reachable region of the specified duration, in which the probability of a point represents the reachability between the location and the point. However, developing such method poses three major challenges:

Efficiency of reachability computation: The reachability of locations cannot be easily computed from the sheer volume of taxi trajectories with billions of GPS coordinates because reachability is inferred from continuous taxi trajectories. To avoid iterating over the huge trajectory dataset, Wu et al. [39] proposed a tree-based spatiotemporal index to accelerate the computation of reachability. However, this method is incapable of handling real-time queries and does not scale well with the size of the dataset. Interactions and visualizations demand a new efficient algorithm for computing reachability.

Representations of daily routines: Filtering home candidates based on daily routines requires concise descriptions of these routines. We define daily routines as a series of reachability constraints. A reachability constraint can be quite complex. It may involve multiple chained activities (e.g., go to place B, spend some time, and then go to place C), multiple candidate locations (e.g., prefer any school in a certain area), and constrained reachable probability (e.g., being late for work is unacceptable). An effective method for organizing and visualizing these constraints, such that users can easily specify and refine the constraints progressively, has yet to be proposed.

Integration of individual preferences: People may have different requirements with respect to their preferences. The diversity in opinions requires good flexibility in expressing preferences, thereby denying a completely automated recommendation algorithm. Thus, an interactive method is demanded to assist users in specifying their requirements, thereby enabling them to filter, compare, and rank numerous home candidates involving multiple criteria.

Our study characterizes user requirements and analytical tasks in the context of finding an ideal home, and addresses the aforementioned challenges from three aspects. First, we design a novel reachability model based on a graph-based index and a query algorithm to efficiently estimate the reachability of locations with massive trajectories. Second, we introduce a novel timeline view to support the orchestration, organization, and visualization of complex reachability constraints as a representation of users' daily routines. Third, we design and develop a visual analytics system called **Re**achability-**A**ided **C**ontemporary **H**omeFinder (*ReACH*) to assist users in interactively querying, filtering, and evaluating home candidates to identify their ideal homes. To our knowledge, the system is the first visual analytics system that integrates data-driven computation and user-centric visualization to find ideal homes. The major contributions of this study are summarized as follows.

1. We characterize user requirements and analytical tasks in the context of finding ideal homes.

2. We propose a real-time data-driven computation model with a graph-based index to handle interactive queries for probabilistic reachable regions.

3. We design a new timeline view for orchestrating, organizing, and visualizing reachability constraints. Based on the view, we develop ReACH, a novel visual analytics system, to assist users in finding ideal homes based on multiple criteria.

RELATED WORK

This section presents studies related to our research in four categories, namely, reachability query, location selection, urban visualization, and multi-criteria decision-making.

Reachability query is a classic graph query that determines the reachability of one vertex to another under certain constraints. It has been extensively studied in static [10, 35, 6, 36] and dynamic graphs [40, 49]. Road network is a special type of graphs associated with dynamic data, such as trajectories. Wu et al. [39] proposed a tree-based method for estimating the reachability between roads on the road network based on massive taxi trajectories. However, this method cannot handle interactive queries generated by visual analytics. Therefore, we develop an efficient graph-based method to efficiently approximate reachability and answer queries in real time.

Location selection has become a crucial problem in urban planning as massive heterogeneous data are being collected in cities. Prior studies have proposed many models and algorithms to address these problems, such as the location selection of retail stores [20], ambulance stations [24], and charging stations [23]. However, most models and algorithms are designed to work automatically with several fixed criteria, which tend to generate unsatisfactory results, given the complexity of urban environments. Effort has been exerted to integrate human expertise by visual analytics in making an informed location decision. Liu et al. [25] leveraged the maximum coverage model and adopted state-of-the-art visualization methods to assist advertising experts in evaluating and choosing billboard locations. Nonetheless, our study is distinguished from prior studies by focusing on addressing the location selection problem with the reachability of candidate locations.

Urban visualization has become one of the most popular research topics in the visualization community. Most 2D urban visualization techniques are categorized into point-, line-, and region-based techniques [48]. Point-based visualization techniques directly place points on spatial contexts [13, 21]. Line-based techniques visualize trajectory data on road maps or in traffic networks [1, 17, 45]. Region-based techniques present aggregated information in pre-segmented regions [46, 28, 31]. A few visual analytics systems related to reachability have been developed based on one or more categories of these techniques. BoundarySeer [41] visualizes the changes in the boundaries of reachable regions and assists experts in analyzing the reachability in the urban context by depicting the geometric properties of regions. Zeng et al. [44] designed a system for visualizing the mobility of passengers in public transportation systems by focusing on the temporal patterns of the passenger reachability. Nonetheless, these methods cannot be directly applied to the problem of home location selection, which involves spatial and abstract properties.

Multi-criteria decision making (MCDM) aims to make decisions based on multiple criteria. Involving users in a MCDM process contributes to an informed decision [4, 5, 7]. Visual ranking is one of the most common techniques for user-centric decision making. This technique ranks and visualizes candidates to assist users in making decisions transparently and efficiently. Seo and Shneiderman [32] proposed a rank-by-feature approach consisting of a ranked list and scores with ordered bars. RankExplorer [33] visualized data trends using stacked graphs and encodes the variation in rankings using color bars and glyphs. Other studies have attempted to extend visual ranking to multiple criteria. Behrisch et al. [3] demonstrated ranking with multiple attributes by using small multiples and a radial node-link representation. LineUp [16] employed stacked bar charts to compare different objects and explain how the weights of multiple attributes affect the final ranking. WeightLifter [30] further analyzed the space of attribute weights and enables users to investigate the sensitivity of the weights. Accordingly, we use prior studies as inspiration to integrate visual ranking into our system, thereby assisting users in ranking candidate homes based on multiple criteria.

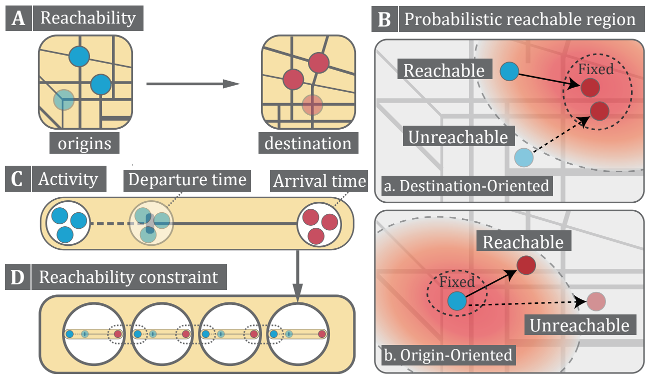


Figure 1. Four major reachability concepts. (A) *Reachability* describes the accessibility between two location sets. (B) A *probabilistic reachable region* comprises all locations with the reachability above a threshold that computes from *fixed locations* (origins (b) or destinations (a)) in an activity. (C) An *activity* defines a reachability query with spatial and temporal parameters. (D) A *reachability constraint* comprises one or more consecutive activities and represents users' daily routines.

DATA AND TASK ABSTRACTION

This section presents the background of our study, describes the dataset, and discusses user requirements and analytical tasks that guide the design of our visual analytics system.

Background and Concepts

ReACH is a visual analytics system designed for people who desire to buy or rent houses, apartments, or condominiums. We refer to these people as the *users* of our system. When finding an ideal home, users need choose from a list of available homes. These homes and their locations are referred to as *candidates* and *candidate locations*, respectively. We characterize the concepts involved in location selection of homes based on reachability as follows.

- *Reachability* (Figure 1A) describes a set of locations' accessibility to another set of locations on the road network in a given duration. We measure the accessibility empirically as the ratio of days when locations in two sets are connected by some trajectories under the time constraint.
- A *probabilistic reachable region* (Figure 1B) comprises all locations with the reachability, which is measured as the ratio of the reachable days from a set of given origins within a travel duration, higher than a specified threshold. The threshold is also referred to as the *reachability threshold* (i.e., the *reliability* of reachability computation results).
- An *activity* (Figure 1C) defines a reachability query, which comprises a set of origins, estimated time spent at the origins, departure time, reachability threshold, arrival time, and a set of destinations. An activity is *satisfied* if a user can depart from any origin and arrive at any destination under the given conditions. Either the origins or the destinations are *fixed*: by computing the reachability from the locations at the fixed end (e.g., places of work), the unreachable locations at the non-fixed end (e.g., candidate locations) are excluded in the result, as illustrated in Figure 1B. In this way, we prune the search space of satisfactory candidates by computing incrementally from a small number of locations because they must be reachable.
- A *reachability constraint* (Figure 1D) comprises one or more *consecutive activities*: the destinations of an activity are the origins of the following activity. Either the origins or the destinations of a reachability constraints are fixed, such that the system generates a set of candidates by computing non-fixed locations satisfying each activity sequentially from the fixed end of the constraint. Thereafter, the intersection of the sets generated from all constraints will match users' requirements according to their daily routines.

Apart from the above reachability concepts, the *intrinsic properties* of candidates (e.g., price, floor size, and number of bedrooms) are crucial factors and should be noted in the process of finding an ideal home. However, the importance of an intrinsic property may vary across different groups of users because of the substantial diversity in their preferences.

Data Description

Four types of data are used in our study. These data were collected in the same city to ensure consistency.

- *Road network data* comprise a directed graph that describes a large city, where the vertices represent road intersections and the edges represent roads. The graph has 183,749 vertices and 244,233 edges.
- *Taxi trajectory data* comprise 1,783,249,500 snapshots of taxi status collected from 8,816 taxis over a two-month period. These snapshots were reported every 30 seconds on average by the sensors installed on vehicles. Each taxi snapshot contains a plate number, GPS coordinates, the current speed, a current direction, and a timestamp.
- *POI data* include 243,713 points of interest in the city, such as restaurants and shopping malls. Each point is tagged with GPS coordinates and a predefined category.
- *Candidate data* include 1,927 candidates that were available for sale in the city in December 2016. Each entry comprises several spatial and intrinsic properties, such as GPS coordinates, floor size, and price.

Task Analysis

In the design process of our system, we organized three brainstorming sessions with 17 people who have experience in buying or renting apartments. In addition, we identified their requirements in finding an ideal candidate. Two participants were faculty members, two other participants had several years of expertise in urban planning, and the rest were undergraduate and graduate students. Based on their comments, we summarized the collected user requirements into two major categories, namely, **R.1**, which concerns the spatial properties of the candidates (e.g., location and neighborhood); and **R.2**, which is relevant to the intrinsic properties of the candidates (e.g., price and floor size). We carefully analyzed these requirements and compiled a list of tasks as follows.

T.1 **Orchestrate reachability constraints interactively.** All participants agreed that an interactive method should be implemented to assist users in formulating and organizing reachability constraints that closely represent their daily routines (R.1). The proposed method must support conveniently specifying the spatial (e.g., origins and destinations) and temporal (e.g., time spent and travel duration) parameters of an activity in constraint. The system should compute the reachability for each activity and generate a list of candidates that match the specified constraints.

T.2 **Explore computed reachability visually.** Participants need an effective way to explore the reachability computed

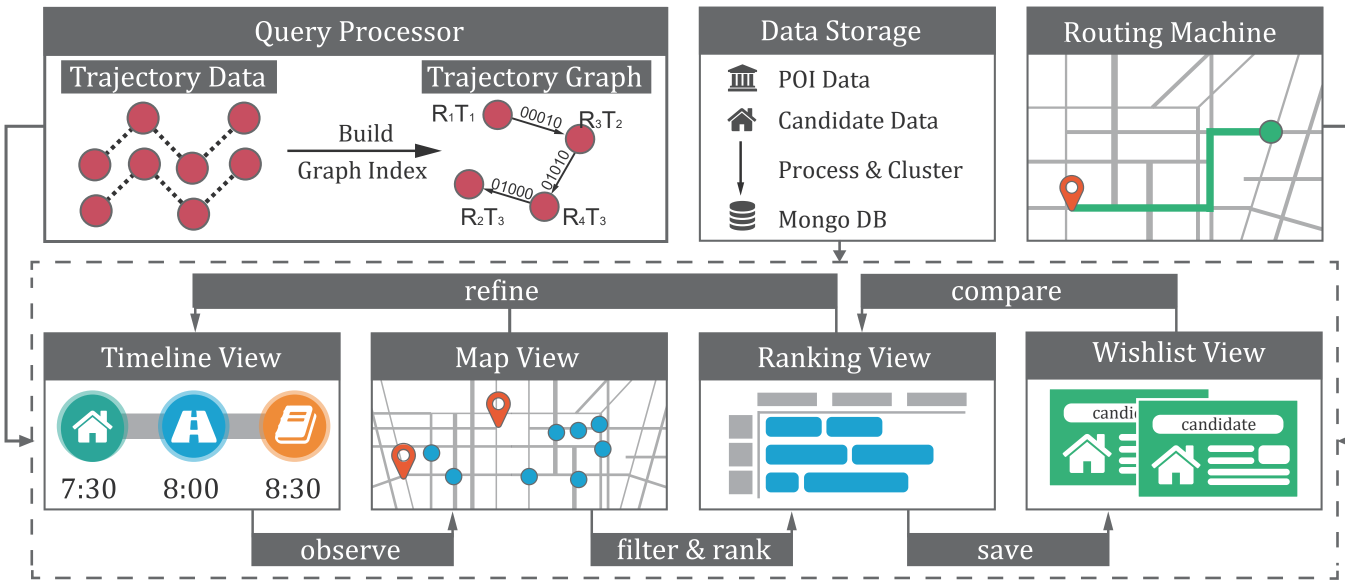


Figure 2. The backend of ReACH comprises three major components. Query Processor handles interactive queries for reachability; Data Storage delivers the processed data from the database on demand; Routing Machine generates the fastest routes among locations. The frontend of ReACH comprises four views, which provide a novel integrated solution for finding ideal homes based on multiple criteria.

in each activity, such that they can further refine the parameters or rearrange daily routines. However, participants indicated that the concept of the reachability and the probabilistic reachable region could be complex for average users. Thus, the proposed visual presentation of reachability should be concrete, accurate, and readily interpretable.

T.3 **Refine reachability constraints iteratively.** Participants requested that they would like to refine constraints by comparing the candidates with those generated from alternative configurations (R.1). For example, a user may wonder how the candidates change if he drives his children to schools in a region instead of those in another region. Furthermore, an automated candidate filtering algorithm based on the given reachability constraints may produce unsatisfactory results (e.g., producing considerably few or many candidates). The system should enable users to control the results of the algorithm. This capability can be realized by the manipulation of the parameters of the activities, such that users can evaluate the results and refine constraints iteratively.

T.4 **Filter and rank candidates effectively.** Different participants have different ideal homes. Some are price-sensitive, while some are reachability-sensitive. Individual preferences should be accurately reflected in the system (R.2). Particularly, participants complained that the filtering features of the extant online systems were “too limited”, and “only few options are available”. These features are also suffered from the all-or-nothing phenomenon [38], where users perform considerably broad or restrictive queries without familiarizing with the statistics of candidates beforehand. As such, the system is required to provide insights into the multidimensional candidate data as users filter and rank candidates progressively (R.2).

SYSTEM ARCHITECTURE

ReACH is a web-based application written in JavaScript. Figure 2 illustrates the system architecture. Users interact with the system with the frontend of ReACH running in the browser, while the backend stores data and answers reachability queries. The frontend of ReACH comprises four views, namely, timeline, map, ranking, and wishlist. Users can create and refine reachability constraints in the timeline view (T.1, T.3), explore reachability in the map view (T.2), filter and rank candidates in the ranking view (T.4), and save ideal homes for later comparisons in the wishlist view (T.4). The backend mainly serves three purposes, namely, (1) storing candidates and other types of data, (2) processing reachability queries with our model, and (3) handling spatial queries with MongoDB [29] and Open Source Routing Machine [27].

MODEL

This section presents a novel graph-based index and a query algorithm for efficient reachability computation.

Background and the Prior Method

The road network of a city can be viewed as a directed graph $G = (V,E)$, where $V = \{n_1,n_2,...\}$ is the set of road intersections and $E = \{r_1,r_2,...\}$ is the set of roads. The GPS coordinates in the taxi trajectories are map-matched [26] to the corresponding roads. Each map-matched taxi trajectory T_i comprises a set of consecutive records $\{R_1,R_2,...\}$. Each record $R_j = (t_j,t_{j+1},r_j)$ indicates that the vehicle was on road $r_j \in E$ during the time span $[t_j,t_{j+1})$.

We briefly restate the computational approach introduced by Wu et al. [39]. The number of days covered by the trajectory dataset is denoted by D. Given two locations A and B, we map-match these locations to the nearest road segments r_A and r_B. By iterating over the dataset with indices, the number of days when a trajectory passes through r_A and then r_B during a specified duration is counted and denoted by d. Thus, the reachable probability from location A to B is estimated as d/D. However, such approach cannot directly support the analytical tasks because of the following limitations:

Computation is slow and space inefficient. Computing a 20-minute reachable region with 1-month taxi trajectory data requires approximately 20 seconds on a cluster of 3 machines.

Reachability is computed only for partial roads. The prior method guarantees efficiency by excluding roads from the computation if the reachability of these roads is below a threshold. However, as requested by T.2, all roads must be iterated over to generate an overview of reachability.

Only individual trajectories are considered. By connecting multiple trajectories, the system can infer the reachability between A and C from one trajectory between A and B and another between B and C. This way, trajectories are consumed efficiently, thereby reducing space overhead.

Graph-based Method

We designed a graph-based method to circumvent the aforementioned limitations. Instead of scanning trajectories directly, we compress them into a trajectory graph based on the assumption that numerous trajectories share partial routes with each other. These identical routes can be compressed to remove redundancy and accelerate the search process.

Building a Trajectory Graph

We split a day (1440 minutes) into a set of k-minute time slots, $M = \{m_1, m_2, \ldots\}$, to conform with the discretization step of the prior approach. Given a road network $G = (V, E)$, we define a *trajectory graph* on the trajectory set T: $G_T = (M \times E, E_T)$, the nodes of which are indexed by the combinations of a time slot m_i and a road r_j. The goal of the building process is to obtain the edge set E_T of the trajectory graph.

Constructing an uncompressed set of edges: First, we scan trajectories $T_1, T_2, \ldots$ to construct an edge set $E'_T = \{(m_i, r_u; m_j, r_v), \ldots\}$ that contains the edges connecting from the node (m_i, r_u) to the node (m_j, r_v).

- For each record $R_j = (t_j, t_{j+1}, r_j) \in T_i$, the time slots m_j and m_{j+1} corresponded by t_j and t_{j+1} are identified. If $m_j \neq m_{j+1}$, then an edge $(m_j, r_j; m_{j+1}, r_j)$ is added.
- For each consecutive pair of records $\{R_j = (t_j, t_{j+1}, r_j), R_{j+1} = (t_{j+1}, t_{j+2}, r_{j+1})\} \subseteq T_i$, the time slot m_{j+1} corresponded by t_{j+1} is identified. If $r_j \neq r_{j+1}$, then an edge $(m_{j+1}, r_j; m_{j+1}, r_{j+1})$ is added.

Compressing the set of edges: We apply the following compression procedure to obtain $E_T = \{(m_i, r_u; m_j, r_v; S), \ldots\}$, where S is a set of days numbered from 1. For each edge $e = (m_i, r_u; m_j, r_v)$ in the uncompressed edge set E'_T, day d is identified when the corresponding trajectory was recorded. If an edge $e' \in E_T$ exists, of which the origin and destination nodes are the same as those of e, then we have $e'.S = e'.S \cup d$. Otherwise, we add an edge $(m_i, r_u; m_j, r_v; \{d\})$. For efficiency, the set S is implemented with bit sets.

Figure 3 illustrates the construction of a trajectory graph. The red and blue trajectories were recorded on days 2 and 3, respectively,and each trajectory comprises three records. The second and third records in two trajectories can be compressed, thereby producing the edges in green. Bit sets on the edges represent the days when the trajectories were recorded.

Computing Reachability with the Graph

Given a starting road and a set of time slots representing travel duration, the query algorithm determines a set of reachable days for every road by executing depth-first search on the trajectory graph. Algorithm 1 illustrates a detailed implementation of the querying algorithm.

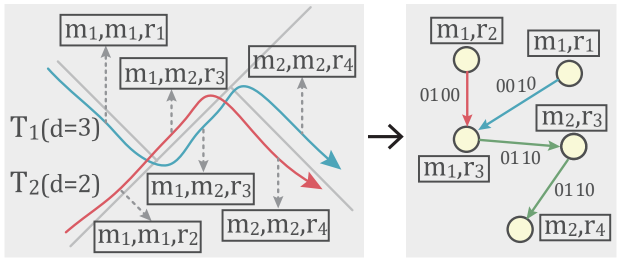


Figure 3. A trajectory graph with 5 nodes constructed from 2 trajectories (blue on day 3 and red on day 2), where m_i represents a time slot and r_i represents a road. Compressed edges are drawn in green.

Algorithm 1 Query reachable days in a trajectory graph

```
    INPUT: A trajectory graph $G_T$, starting road $r_0$, time slots $M' = \{m_1, m_2, \ldots, m_t\}$, mapping from every road to an average time cost $C$, the length of a time slot $k$.
    OUTPUT: Mapping from every road to a set of reachable days $P$.
 1: procedure SEARCH(node, daysReachable, timeSpent)
 2:     if $|$daysReachable$| = 0$ or timeSpent $> k$ then return
 3:     (timeSlot, road) $\leftarrow$ node
 4:     $P$[road] $\leftarrow P$[road] $\cup$ daysReachable
 5:     for each edge (node; nextNode; daysValid) $\in E_T$ do
 6:         (nextTimeSlot, nextRoad) $\leftarrow$ nextNode
 7:         if timeSlot = nextTimeSlot then
 8:             SEARCH(nextNode,
                    daysReachable $\cap$ daysValid, timeSpent $+ C$[road])
 9:         else
10:             SEARCH(nextNode, daysReachable $\cap$ daysValid, 0)
11: $Q \leftarrow \{(m, r) \in M \times E | m \in M', r = r_0\}$
12: for each node $\in Q$ do SEARCH(node, $\mathbb{N}$, 0)
```

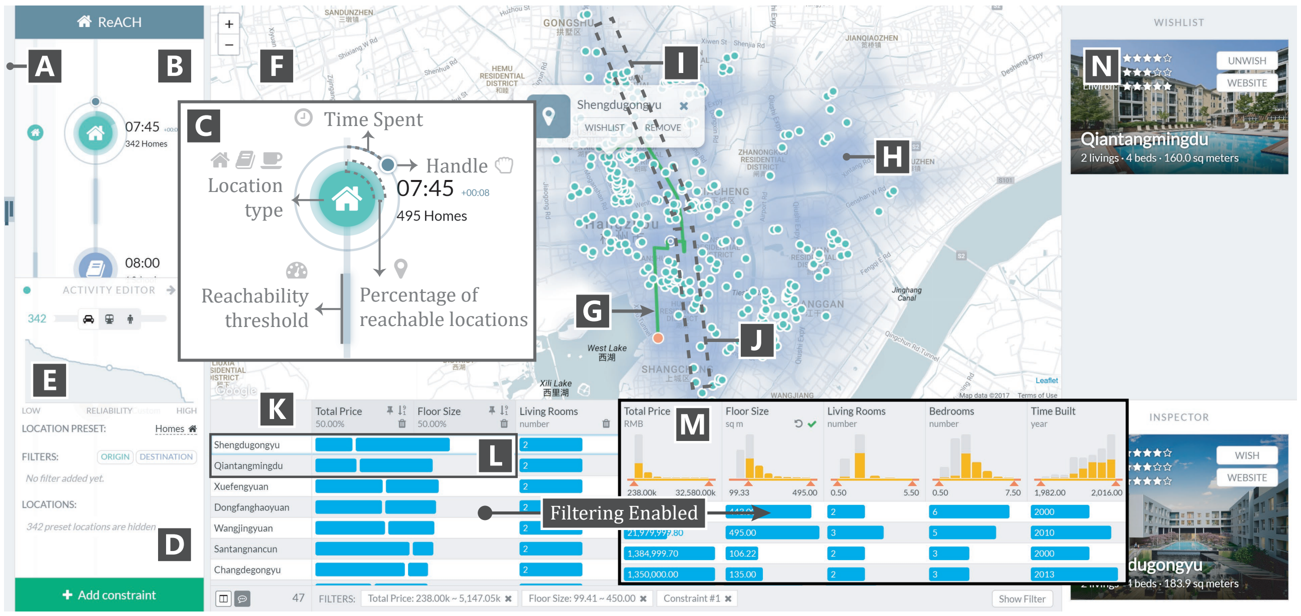
First, the algorithm locates all starting nodes in the trajectory graph that match the given travel duration (cf. line 11) and performs depth-first search from these nodes (cf. line 12). If the stopping criteria were not met (cf. line 2), then the algorithm updates the reachable days for roads (cf. line 4) and continues searching with neighbors (cf. lines 7-10). The reachability for each road is obtained by simply dividing the number of reachable days by the total number of days.

VISUAL DESIGN

To support the aforementioned tasks, we designed four views for ReACH, namely, timeline, map, ranking, and wishlist. Figure 4 illustrates the interface of the system. These views provide a novel integrated solution for finding ideal homes based on multiple criteria. First, the timeline view visually depicts users' daily routines as node-link diagrams, where users can conveniently orchestrate constraints with simple interactions (T.1) and refine activities by modifying parameters or adding spatial filters (T.3). Then, our model computes the reachability between origins and destinations in realtime, and the result is revealed in the map view with heatmap and predicted routes (T.2). Next, the ranking view assists users in filtering and ranking candidates with multiple attributes interactively and progressively (T.4). Finally, users can save interesting candidates in the wishlist view for further comparison.

Timeline View

We design a novel timeline view (Figure 4B) for orchestrating, organizing, and visualizing reachability constraints as a representation of users' daily routines. The proposed timeline view comprises three components, namely, a time axis, user-defined reachability constraints, and an activity editor.



	Total Price 50.00%	Floor Size 50.00%	Living Rooms number
Shengdugongyu			2
Qiantangmingdu			2
Xuefengyuan			2
Dongfanghaoyuan			2
Wangjingyuan			2
Santangnancun			2
Changdegongyu			2

Total Price RMB	Floor Size sq m	Living Rooms number	Bedrooms number	Time Built year
238.00k – 32,580.00k	99.33 – 495.00	0.50 – 5.50	0.50 – 7.50	1,982.00 – 2,016.00
[illegible]	495.00	2	6	2000
21,979,999.80	495.00	3	5	2010
1,384,999.70	106.22	2	3	2000
1,350,000.00	135.00	2	3	2013

Figure 4. The interface of ReACH. (A-E) Timeline view enables the orchestration, organization, and visualization of reachability constraints based on users' daily routines with novel visual designs. (F-J) Map view depicts the locations of candidate houses and the result of reachability computation. (K-M) Ranking view assists users in filtering, comparing, and ranking candidates. (N) Wishlist view keeps desired candidates for further comparison.

Time axis: The vertical time axis (Figure 4A) serves as a minimap of reachability constraints. It shows each constraint vertically as black thin bars, the position and length of which depict the duration occupied by the corresponding constraint. The brushed part of the time axis is displayed on the right.

Reachability constraints: Constraints (Figure 4B) are laid out vertically as node-link diagrams in the timeline view. Only one constraint is active for editing at a moment. Each node, identified by its color in a constraint, represents a set of locations, which can be selected from 10 location presets or picked from the map. We design the node as follows (Figure 4C): (a) inside the node, the glyph indicates the type of locations if a preset is selected; (b) a donut chart surrounding the node encodes the percentage of locations in this node that satisfy the constraint; (c) around the node, a handle is designated to assist users in specifying the time spent on these locations intuitively; and (d) the radius of the node is adaptive, such that the node will shrink and hide annotations to avoid overlapping while keeping the structure of the constraint visible as an overview, if the space is limited. Links between the nodes represent activities. The length of the dark bar on each link shows the reachability threshold of an activity. Moreover, the departure and arrival time of an activity can be adjusted simply by moving nodes in the vertical direction.

Activity editor: The editor (Figure 4D) enables users to view and modify parameters of an activity, including the traveling method, reachability threshold, and locations. The traveling method can be toggled with buttons at the top of the editor. In addition to driving, we also estimate travel durations for walking and public transportation to support flexible combinations of traveling methods. Reachability thresholds are encoded with reachability charts (Figure 4E). To concretize concepts as per T.2, we draw the reachability chart to show the current reachability threshold and visualize how the number of reachable locations changes with the threshold. The x and y axes of the chart encode the reachability threshold and the number of reachable non-fixed locations, respectively. With the reachability chart, users can visually configure the reachability threshold and investigate patterns, such as traffic congestions, that may affect their decisions. Locations can be specified either with the presets extracted from POI data or by interacting with the map view (Figure 4F).

Design alternatives: Graph (Figure 5A) and tree (Figure 5B) representations illustrate the constraints as graphs and trees, respectively. Unlike the linear representation introduced above, these alternatives offer branching in the constraints, such that multiple activities can be simultaneously satisfied. In Figure 5A and B, for example, either activity A2 or A3 can occur after A1, and both of them will be followed by activity A5. However, despite the flexibility in orchestrating complex constraints, users found a branching timeline counter-intuitive and difficult to manipulate shared activities. Thus, based on their feedback, we design the timeline view with the linear representation.

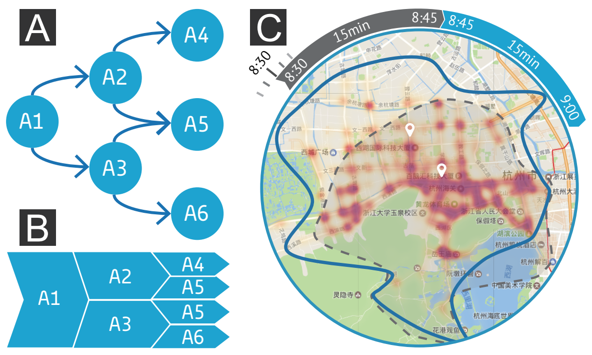


Figure 5. Design alternatives for the timeline and map views. (A) A graph representation of constraints. (B) A tree representation of constraints. (C) A map view encapsulated in a circular time axis.

Map View

To assist users in making decisions in the spatial context effectively, we adopt a map-centered exploratory approach [18] by placing a map (Figure 4F) in the map view with two additional layers, namely, candidate and constraint layers.

Candidate layer: Candidates are represented by the green circles on the layer. When users click on a circle, the system will compute the fastest route that starts with the corresponding candidate and passes each location as specified by the constraint. The route is then drawn on the map as a node-link diagram (Figure 4G), where the colors of intermediate nodes are the same with those of constraint nodes.

Constraint layer: This layer becomes visible when users view an activity in the activity editor. The origins and destinations of the activity are plotted as circles on the layer in respective colors. To visualize the reachability from the fixed locations of the activity, the constraint layer provides a heatmap based on kernel density estimation (Figure 4H). The color of the heatmap varies from transparent to the color of the fixed locations of the activity. The deeper the color is, the more likely users can reach from these locations.

Design alternatives: We explored two alternative methods for specifying the temporal and spatial parameters of activities. One alternative is to directly place the timeline view onto map or use embedding visualization techniques, such as RouteZooming [34]. However, this approach renders the spatial context and reachability constraints incomprehensible, because the map would be occluded or deformed by the scattered locations of constraints. The other alternative is to encapsulate the map view in a circular timeline based on a clock metaphor (Figure 5C). In addition, we estimated the boundaries of reachable regions based on concave hull algorithms and visualized them with contour lines. However, despite the convenience of setting spatial and temporal parameters in a single view without moving back and forth, users complained that such layout was unfamiliar to them, and the contour lines that crossed the water and terrain caused confusion. Consequently, we reverted to the simplified design with separate views for improved usability.

Ranking View

The ranking view (Figure 4K) shows all candidates as a list. Inspired by LineUp [16], the multidimensional candidate data are organized in a table. For every candidate, we estimate the total travel durations for given daily routines and list them in the table. Users can rank the candidates based on an attribute or several aggregated attributes with the separate weights controlled by the width of columns.

Interactive filtering: We improved the range filtering of candidates compared with the simple sliders in HomeFinder [38]. A bar chart is embedded into the header of each column with a slider at the bottom (Figure 4M). Bars in the bar chart evenly split the value range of an attribute, and the height of each bar encodes the number of candidates that match the corresponding value range. Users can either click on the bars or adjust the slider to select a value range for an attribute. Thereafter, the table will be updated to show the candidates that satisfy the selected value range. Additionally, all bar charts will change dynamically according to the remaining candidates, while leaving the original bars in gray for comparisons. This tool enables the easy and interactive creation of filters based on the insights revealed by the distribution of candidates.

Wishlist View

Users may explore and rank candidates with different daily routines, activity parameters, or settings of the attribute weights. In the trial of the first prototype, the participants requested a list for storing the desired candidates they found while using the system. Their feedback prompted us to add the wishlist view (Figure 4N) for storing candidates.

Interactions

Several interaction techniques have been implemented in the system to enable users to orchestrate constraints and analyze candidates across multiple views.

Timeline: Users can navigate to any duration by selecting and brushing on the time axis. New constraints can be created with the “Add Constraint” button, and users can add a new node by clicking on the links in the node-link diagram and adjust its precise time by dragging it vertically. Moreover, a non-linear constraint can be established by adding multiple linear ones, and the filtered sets of houses are eventually intersected.

Spatial filtering: The spatial filtering tool in the timeline view enables users to draw polygons on the map to exclude the locations that are not covered by the drawings. The timeline and ranking views will be updated to reflect the changes on locations accordingly. In addition, every location can be manually deleted in the popup on the map.

Highlighting: Users can select and highlight a candidate in the map, ranking, and wishlist views. When a candidate is selected, the map view pans to the candidate’s position and shows a popup with its name and action buttons, such as removing the candidate. The ranking view highlights the corresponding row and move the candidate to the top of the table. The wishlist view shows a home card for the candidate, where the candidate can be added into the wishlist.

EVALUATION

This section shows the improvements in our model compared with the prior method and demonstrates the effectiveness and usability of ReACH with two usage scenarios and a task-based evaluation. Our system is deployed on a workstation with two Intel Xeon E5-2620 CPUs and 128 GB of memory, and interactive performance is achieved. Users can access the system through web browsers on standard PCs and engage in the experiments conducted with the data described.

Complexity Analysis of the Model

This section theoretically compares two methods in terms of space and time complexities. To simplify the problem, we fixate the total number of days D to predict the complexities of the algorithm against the data complexities of the trajectory dataset and city structure. Without loss of generality, we assume each vehicle has only one continuous trajectory and each trajectory passes $\beta(T)$ roads on average in a time slot.

Space Complexity

Prior method: The ST-Index, which indexes the trajectory data, comprises three layers. The first two layers contain the spatial indexes that are nested in a temporal index. Given that each road has $\beta(T)/|E|$ trajectories on average, the space consumption of the ST-Index is computed as $|M|+|M||E|+\beta(T)D|M||T|$. The space consumption of another index (i.e., Con-Index) is disregarded, because computing the reachability for all roads does not require this index. Thus, the overall space complexity is

$$\Theta(\beta(T)|M||T|+|M||E|).$$

In practice, the ST-Index can easily consume hundreds of gigabytes. Hence, maintaining such index in memory will be substantially difficult if the complexities of the data scale up.

Graph-based method: Assuming that the average trajectory compression rate is $\gamma(T)=|E_T|/|E'_T|$, we estimate in two folds that the number of edges in a trajectory graph is $\gamma(T)\beta(T)D|M||T|+\gamma(T)D(|M|-1)|T|$. Thus, we predict the space complexity of the trajectory graph as follows:

$$\Theta(\gamma(T)\beta(T)|M||T|).$$

Compression rate $\gamma(T)$ can significantly affect the size of a trajectory graph, but we can only obtain the value empirically. In our case, approximately 11.4 GB is required to store the two-month trajectory data in a graph ($\beta(T)\approx 7.81$, $\gamma(T)\approx 0.106$). In the worst case scenario, the trajectory graph would become completely connected. Therefore, the upper bound of the space complexity is $O(|M||E|^2)$. Graph compression techniques[12] can be applied to reduce the space complexity because the graph is often sparse.

Time Complexity

Prior method: An exhaustive search is performed in the ST-Index. If we compare the trajectories passing each road with those passing r_0 in every time slot ($\Theta(n)$ each comparison if roads were sorted), the overall time complexity is

$$\Theta(\beta(T)|M|^2|T|).$$

Graph-based method: In the worst case scenario, the algorithm must traverse all edges in the graph and the estimated number of edges is $\gamma(T)\beta(T)D|M||T|+\gamma(T)D(|M|-1)|T|)$. We use the time complexity of the depth-first search as basis to predict the complexity of our method as follows:

$$\Theta(|M||E|+\gamma(T)\beta(T)|M||T|).$$

Although the growth of $\gamma(T)$ cannot be determined, we argue that the graph-based method has the following advantages.

- The stopping criteria (cf. line 2 of Algorithm 1) and optimizations allow the algorithm to skip most of the edges.
- The algorithm computes without disk reads because the generated trajectory graph can fit into the memory.
- In practice, 20-minute reachable regions can be generated in 3-4 seconds on a workstation in contrast to approximately 20 seconds demonstrated in the prior study with a cluster.

Usage Scenarios

We demonstrate the effectiveness and usability of ReACH with two usage scenarios by following John, a software engineer who has recently acquired a new job in a city. He is looking for an apartment to move in with his family.

Orchestration of Constraints

To avoid the classic all-or-nothing phenomenon [38] where users attempt broad or restrictive queries before knowing the content, attribute filters in the ranking view can serve as a preliminary tool for exploring candidate data. Therefore, John toggles attribute filters from the ranking view (Figure 4M). He immediately notices the unbalanced, long-tailed distribution of price ranging from 238 thousand to nearly 33 million dollars. Since John cannot afford such luxurious apartments, he adjusts the upper bound of price to 5 million. In addition to the price, floor size is also one of the major concerns for a two-kid family, so John rises the lower bound of the floor size to 100 square meters. After filtering, roughly 600 candidates remain in the ranking view.

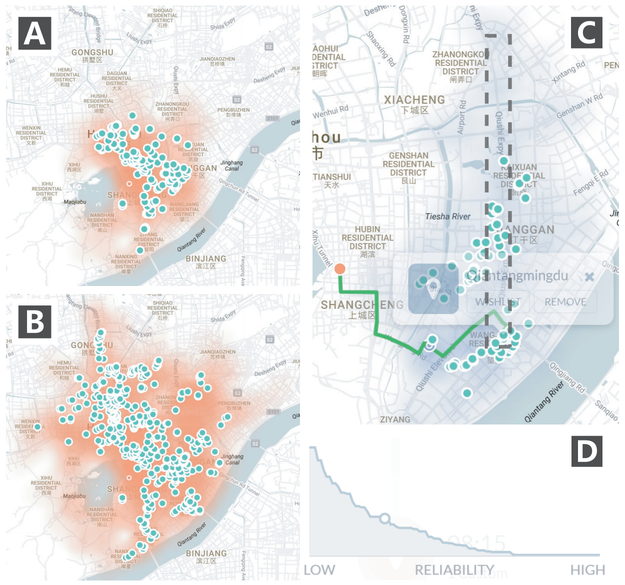


Figure 6. Heatmaps illustrating the reachability within 15 minutes (A) and 30 minutes (B). Green nodes represent candidates before being filtered by the ranking view. (C) High reachability along Qiushi Expressway circled in the dashed line. (D) The number of candidates drops fast as the reachability threshold increases.

John then proceeds to the timeline view. He decides to leave home at 8:00 in the morning and adds another 8:15 node with his place of work picked from the map view. However, only 16 candidates match his requirements after being filtered by the ranking view because the generated reachable region (Figure 6A) is smaller than expected. By browsing the ranking list, John is unable to find ideal candidates, because apart from old apartments, two recently-built ones (year built ≥ 2010) are not cost-effective: these two apartments are expensive with small floor size. Hence, John changes his mind and decides to leave home 15 minutes earlier. This time, a larger reachable region (Figure 6B) is generated with 77 candidates after being filtered. With the area chart in the activity editor, John increases the reachability threshold of the activity, such that he maintains the number of candidates to choose from while reducing the probability of being late for work.

Comparison with Reachability and Ranking

Planning to drop off his children at school on his way to work, John adds a node at 8:00 and chooses a school location preset in the activity editor. With the result of preliminary surveys prior to using our system, John has found two equally-preferred schools, School #27 and School #325. Since either school is suitable for his children, he would like to compare candidates based on the school choice he makes.

First, John locates School #27 in the map view, which is near Qiushi Elevated. A spatial filter is applied, such that the school

No.	*Task Description*	*A. Task(s)*
E1	Leave home at 7 a.m. and arrive at place A at 8 on weekdays. How many candidates are left?	T.1
E2	Observe the heatmap. Is it possible to reach place A in time from candidates B, C, or D? Which one has the highest reachable probability?	T.2
E3	Increase the reachability threshold. Which one in B, C, and D has the highest reachable probability?	T.2, T.3
E4	Add a intermediate node for schools. Compare the results of 3 regions of schools X, Y, and Z. Which one leads to the largest number of candidates?	T.3
E5	Find in the result of each region the top candidate with price lower than 5 million and with 2 bedrooms ranked by floor size. Which region leads to the best candidate?	T.4

Table 1. Evaluation tasks and questions.

No.	*Question*
P1	Is the interface of ReACH intuitive and easy to use?
P2	Is the workflow of ReACH easy to learn?
P3	Does ReACH correctly reflect the needs in finding ideal homes?
P4	Is the generated reachable region reasonable?
P5	Does the timeline view help you filter candidates with reachability based on daily routines effectively?
P6	Is the timeline view intuitive and easy to use?
P7	Does the map view help you compare the reachability clearly?
P8	Does the ranking view help you filter and rank candidates based on requirements effectively?
P9	Is the ranking view intuitive and easy to use?
P10	Does the wishlist view help you in the comparisons of candidates?
P11	Is the wishlist view intuitive and easy to use?

Table 2. Post-study questionnaire.

node in the timeline view only consists of School #27. From the reachability heatmap (Figure 6C), John discovers that high reachable regions are mainly distributed along Qiushi Expressway (circled with the dashed line), which extends to the north of the city. However, he notices that the number of distant candidates reduces quickly as he increases the reachability threshold between homes and the school (Figure 6D), indicating high possibility of traffic congestions along Qiushi Expressway. Thus, he lowers the threshold for more choices and obtains an ideal candidate, Qiantangmingdu, by ranking the candidates with 50% price and 50% floor size. This candidate is saved to the wishlist for further comparison.

Then, John locates School #325 and applies another spatial filter. The heatmap reveals the excellent reachability of this school (Figure 4H), which is mainly contributed by two roads, Shangtang Elevated (Figure 4I) and Zhonghe Elevated (Figure 4J), circled with the dashed lines. Moreover, the number of candidates reachable from School #325 (Figure 4E) is considerably more stable than that reachable from School #27 (Figure 6D). By browsing the ranking view with the same weight settings, John discovers that Shengdugongyu is the best choice if he sends his children to School #325. To make the comparison, he adds Qiantangmingdu back from the wishlist. The former best choice is clearly outperformed by the current one (Figure 4L): despite the slightly cheaper price, Qiantangmingdu is smaller than its competitor by more than 20 square meters. Therefore, John decides to contact the owner of Shengdugongyu for more details about the apartment.

Task-based User Study

We conducted a task-based user study to evaluate the proposed system and find potential usability issues.

Participants and Data

We recruited 14 subjects (7 males and 7 females) from different departments, including Agronomy (4), Computer Science (3), Medicine (2), Electronic Engineering (1), Journalism (1), Mathematics (1), Oceanography (1), and Civil Engineering (1). These subjects were identified by S1-S14, respectively. None of them was involved in the development of the proposed system. The study was conducted based on candidate data that comprise the apartments for sale during December 2016.

Evaluation Procedure and Tasks

The study was conducted on an individual basis. First, the subjects were presented with an introduction to the functions and views of the proposed system. Then, the subjects practiced and gained familiarity with the system for 5 minutes. Afterward, they performed a series of evaluation tasks and answered the questions listed in Table 1. These evaluation tasks were designed to correspond with the analytical tasks that we previously summarized. Lastly, the subjects were asked to fill in a post-study questionnaire designed using a 7-point Likert scale, where they rated their experience and the quality of our system. Questions in the questionnaire are listed in Table 2. In addition, their feedback was collected and analyzed after the evaluation.

Experiment Result

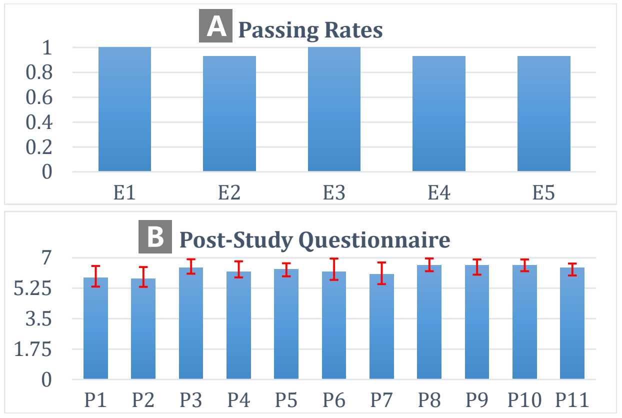


Figure 7. Analysis of passing rates (A) of evaluation tasks (Table 1) and user ratings (B) of post-study questionnaire (Table 2).

Figure 7 shows the result of the evaluation, including the passing rates of the evaluation tasks and mean values of post-study ratings with 95% confidence intervals.

Passing rates: Each session of the task-based user study lasted 15-20 minutes, and we considered a subject passed a task if he/she answered the question specified in each task correctly. The passing rates of the evaluation tasks (Figure 7A) are satisfactory: the subjects achieved 4.79 points out of 5 on average. The effectiveness of our system in depicting the reachability is proved by the high passing rates of E1-E3. Only subject S10 failed E2, because she could not distinguish the difference in the densities of colors on the heatmap. Another subject S9 was unfamiliar with regions X, Y, and Z, thereby giving incorrect answers for E4 and E5. Nonetheless, most of the subjects completed the evaluation tasks smoothly and efficiently with our system.

Post-study ratings: Generally, the subjects gave high ratings for all questions and confirmed the effectiveness and usability of our system (Figure 7B). The subjects agreed that the criteria involved in this study were critical in the context of finding an ideal home (P3). They also approved the accuracy of our model with a high rating for P4, based on their commuting experience. Furthermore, they liked the design of the timeline view and rated high scores for its effectiveness (P5) and usability (P6). However, the average ratings for overall intuitiveness (P1) and approachability (P2) were the lowest among all, mainly because: (a) the approachability of our system slightly suffered from the novelty and flexibility of reachability concepts, compared to the simple yet straightforward interfaces of the extant online systems; and (b) some subjects concerned that manual guidance might be required before using the system. This can be addressed by integrating visual guidance techniques [8] to help average users quickly familiarize with the visualizations. We also noticed that the rating for the map view was lower than those of other views, which was contributed by the difficulty in comparing the reachability of multiple candidates by observing the density of colors. The rating implied that adjusting the reachability threshold incrementally coubld be more suitable for reachability comparisons than observing heatmaps at a finer scale.

Users' feedback: The subjects were interviewed after the user study. They appreciated the novelty of our system. "It's a really nice idea to query candidates based on people's daily routines." commented subjects S4 and S7. They also conveyed a positive attitude toward the usability of our system. Subject S1 told us, "It is convenient to specify and view daily routines with the timeline view." The subjects were also interested in the implementation of the system and gave affirmative comments, such as "Interactions are very smooth and fluid."

We also received valuable feedback regarding improvements of the system. Subjects S1 and S9 would like a search box in the map view to assist them in locating roads or points of interest. Subject S1 also suggested that attribute filters should allow users to specify exact value ranges with text boxes. We improved our system accordingly.

DISCUSSION

Lessons learned: We summarize several lessons learned from the design process of ReACH. First, a good balance between flexibility and simplicity plays an important role in encouraging the masses to use the proposed system. Second, carefully designed animated transitions and interactions among multiple coordinated views enable users to immediately familiarize themselves with the system. Third, the familiarity and intuitiveness of the visual encodings exert a crucial effect on the usability of the visualizations for the masses. We selected several well-established visualization techniques with gentle learning curves, such as area charts, heatmaps, and table-based ranking. These techniques are seamlessly integrated to support the decision-making workflow of users.

Limitations: Our system has two limitations. The first limitation is from the model. The performance analysis in subsection 7.1 demonstrates the time and space efficiency of the proposed model compared with the state-of-the-art model [39]. Our model can efficiently handle two months of full trajectory data of nearly all taxis ($8,816$) in a large city with 10-million inhabitants on a single workstation. In addition, the user study proves the reliability of the proposed model with the two-month data. However, increasing the volume of the data can result in degraded time performance of this model, particularly when the constructed trajectory graph is considerably large to fit into the memory. To mitigate this problem, we plan to integrate techniques, such as graph compression and distributed computing, into ReACH. The second limitation is from the visual design. Despite the reasonable balance between usability and flexibility, average users who are unfamiliar with the system may still experience difficulty. Accordingly, visual guidance techniques can be integrated into the system to improve its usability [8].

Generalization: Although ReACH is developed for people to find an ideal home location, this system can be adopted in other location selection scenarios, in which the reachability of locations is considered (e.g., selecting a location for a convenience store). The owner of the store may want to find a competitive location that is less reachable from other stores but more reachable from residential areas. Moreover, the overall framework is general and applicable to other datasets and problems. The proposed model is not limited to the trajectory data of taxis. Other movement and mobility data of moving objects, such as the trajectories of sharing bicycles and telco data, can be easily imported to the model to estimate the reachability of locations. The visual design of ReACH is flexible and can easily be extended to integrate other types of information, such as census and crime data. Moreover, the design is independent of the home location selection problem. It can be beneficial for spatial sense-making and decision-making in different domains, such as transportation and urban planning, in which users need to iteratively generate candidate locations and visually compare the generated locations.

CONCLUSION

This study characterizes the problem of reachability-centric multi-criteria decision-making for choosing ideal homes. To assist users in expressing their preferences, we introduce several reachability concepts involved in this problem, including activities and constraints. We propose a new graph-based mining model that significantly extends the state-of-the-art method [39] to achieve real-time performance. We used the model as basis to design and develop a novel data-driven system called ReACH. To the best of our knowledge, ReACH is the first interactive visualization system for people to find an ideal home based on massive urban data.

The proposed system has been deployed on a local workstation. In the future, we intend to deploy this system as a cloud service on the web. The mining model will be improved to reduce the space complexity, thereby lowering the cost for the cloud deployment. Other types of data, which are of interest to people, will be incorporated to the proposed system.

ACKNOWLEDGEMENTS

We thank all participants and reviewers for their thoughtful feedback and comments. The work was supported by National 973 Program of China (2015CB352503), NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Informatization (U1609217), NSFC (61761136020, 61502416), Zhejiang Provincial Natural Science Foundation (LR18F020001) and the 100 Talents Program of Zhejiang University. This project is also partially funded by Microsoft Research Asia.

REFERENCES

1. Shamal Al-Dohuki, Yingyu Wu, Farah Kamw, Jing Yang, Xin Li, Ye Zhao, Xinyue Ye, Wei Chen, Chao Ma, and Fei Wang. 2017. SemanticTraj: A New Approach to Interacting with Massive Taxi Trajectories. *IEEE TVCG* 23, 1 (2017), 11–20.
2. Altisource Inc. 2017. Hubzu: Homes For Sale | Online Real Estate Auctions | Property Listings. `http://www.hubzu.com`. (2017). Online; accessed 07-Sept-2017.
3. Michael Behrisch, James Davey, Svenja Simon, Tobias Schreck, Daniel Keim, and Jörn Kohlhammer. 2013. Visual Comparison of Orderings and Rankings. In *Proc. of EuroVis Workshop on Visual Analytics*.
4. Steven Bergner, Michael Sedlmair, Torsten Möller, Sareh Nabi Abdolyousefi, and Ahmed Saad. 2013. ParaGlide: Interactive Parameter Space Partitioning for Computer Simulations. *IEEE TVCG* 19, 9 (2013), 1499–1512.
5. Maryam Booshehrian, Torsten Möller, Randall M. Peterman, and Tamara Munzner. 2012. Vismon: Facilitating Analysis of Trade-Offs, Uncertainty, and Sensitivity In Fisheries Management Decision Making. *Computer Graphics Forum* 31, 3 (2012), 1235–1244.
6. Jing Cai and Chung Keung Poon. 2010. Path-hop: efficiently indexing large graphs for reachability queries. In *Proc. of ACM CIKM*. 119–128.
7. Giuseppe Carenini and John Loyd. 2004. ValueCharts: analyzing linear models expressing preferences and evaluations. In *Proc. of AVI*. 150–157.
8. Davide Ceneda, Theresia Gschwandtner, Thorsten May, Silvia Miksch, Hans-Jörg Schulz, Marc Streit, and Christian Tominski. 2017. Characterizing Guidance in Visual Analytics. *IEEE TVCG* 23, 1 (2017), 111–120.
9. Wei Chen, Fangzhou Guo, and Fei-Yue Wang. 2015. A Survey of Traffic Data Visualization. *IEEE TITS* 16, 6 (2015), 2970–2984.
10. Yangjun Chen and Yibin Chen. 2008. An Efficient Algorithm for Answering Graph Reachability Queries. In *Proc. of IEEE ICDE*. 893–902.
11. Peter F. Colwell, Carolyn A. Dehring, and Geoffrey K. Turnbull. 2002. Recreation Demand and Residential Location. *Journal of Urban Economics* 51, 3 (2002), 418 – 428.
12. Wenfei Fan, Jianzhong Li, Xin Wang, and Yinghui Wu. 2012. Query preserving graph compression. In *Proc. of ACM SIGMOD*. 157–168.
13. Nivan Ferreira, Jorge Poco, Huy T. Vo, Juliana Freire, and Cláudio T. Silva. 2013. Visual Exploration of Big Spatio-Temporal Urban Data: A Study of New York City Taxi Trips. *IEEE TVCG* 19, 12 (2013), 2149–2158.
14. Amy Fontinelle. 2017. Buying A Home: Choosing Your Location. `https://goo.gl/8ZKyJc`. (2017). [Online; accessed 07-Sept-2017].
15. Yanjie Fu, Yong Ge, Yu Zheng, Zijun Yao, Yanchi Liu, Hui Xiong, and Jing Yuan. 2014. Sparse Real Estate Ranking with Online User Reviews and Offline Moving Behaviors. In *Proc. of IEEE ICDM*. 120–129.
16. Samuel Gratzl, Alexander Lex, Nils Gehlenborg, Hanspeter Pfister, and Marc Streit. 2013. LineUp: Visual Analysis of Multi-Attribute Rankings. *IEEE TVCG* 19, 12 (2013), 2277–2286.
17. Xiaoke Huang, Ye Zhao, Chao Ma, Jing Yang, Xinyue Ye, and Chong Zhang. 2016. TrajGraph: A Graph-Based Visual Analytics Approach to Studying Urban Network Centralities Using Taxi Trajectory Data. *IEEE TVCG* 22, 1 (2016), 160–169.
18. Piotr Jankowski, Natalia Andrienko, and Gennady Andrienko. 2001. Map-centred exploratory approach to multiple criteria spatial decision making. *IJGIS* 15, 2 (2001), 101–127.
19. John F. Kain. 1962. THE JOURNEY-TO-WORK AS A DETERMINANT OF RESIDENTIAL LOCATION. *Regional Science* 9, 1 (1962), 137–160.
20. Dmytro Karamshuk, Anastasios Noulas, Salvatore Scellato, Vincenzo Nicosia, and Cecilia Mascolo. 2013. Geo-spotting: Mining Online Location-based Services for Optimal Retail Store Placement. In *Proc. of ACM SIGKDD*. 793–801.
21. Robert Krüger, Dennis Thom, Michael Wörner, Harald Bosch, and Thomas Ertl. 2013. TrajectoryLenses - A Set-based Filtering and Exploration Technique for Long-term Trajectory Data. *Computer Graphics Forum* 32, 3 (2013), 451–460.
22. Brian H Y Lee, Paul Waddell, Liming Wang, and Ram M Pendyala. 2010. Reexamining the Influence of Work and Nonwork Accessibility on Residential Location Choices with a Microanalytic Framework. *Environment and Planning A* 42, 4 (2010), 913–930.
23. Yuhong Li, Jie Bao, Yanhua Li, Yingcai Wu, Zhiguo Gong, and Yu Zheng. 2016. Mining the most influential *k*-location set from massive trajectories. In *Proc. of ACM SIGSPATIAL*. 51:1–51:4.
24. Yuhong Li, Yu Zheng, Shenggong Ji, Wenjun Wang, Leong Hou U, and Zhiguo Gong. 2015. Location Selection for Ambulance Stations: A Data-driven Approach. In *Proc. of ACM SIGSPATIAL*. 85:1–85:4.
25. Dongyu Liu, Di Weng, Yuhong Li, Jie Bao, Yu Zheng, Huamin Qu, and Yingcai Wu. 2017. SmartAdP: Visual Analytics of Large-scale Taxi Trajectories for Selecting Billboard Locations. *IEEE TVCG* 23, 1 (2017), 1–10.
26. Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-matching for low-sampling-rate GPS trajectories. In *Proc. of ACM SIGSPATIAL*. 352–361.
27. Dennis Luxen and Christian Vetter. 2011. Real-time routing with OpenStreetMap data. In *Proc. of ACM SIGSPATIAL*. 513–516.
28. Fábio Miranda, Harish Doraiswamy, Marcos Lage, Kai Zhao, Bruno Gonçalves, Luc Wilson, Mondrian Hsieh, and Cláudio T. Silva. 2017. Urban Pulse: Capturing the Rhythm of Cities. *IEEE TVCG* 23, 1 (2017), 791–800.

29. MongoDB Inc. 2017. MongoDB for GIANT Ideas | MongoDB. `https://www.mongodb.com/`. (2017). [Online; accessed 07-Sept-2017].

30. Stephan Pajer, Marc Streit, Thomas Torsney-Weir, Florian Spechtenhauser, Torsten Möller, and Harald Piringer. 2017. WeightLifter: Visual Weight Space Exploration for Multi-Criteria Decision Making. *IEEE TVCG* 23, 1 (2017), 611–620.

31. Roeland Scheepens, Christophe Hurter, Huub van de Wetering, and Jarke J. van Wijk. 2016. Visualization, Selection, and Analysis of Traffic Flows. *IEEE TVCG* 22, 1 (2016), 379–388.

32. Jinwook Seo and Ben Shneiderman. 2005. A Rank-by-Feature Framework for Interactive Exploration of Multidimensional Data. *Information Visualization* 4, 2 (2005), 96–113.

33. Conglei Shi, Weiwei Cui, Shixia Liu, Panpan Xu, Wei Chen, and Huamin Qu. 2012. RankExplorer: Visualization of Ranking Changes in Large Time Series Data. *IEEE TVCG* 18, 12 (2012), 2669–2678.

34. Guodao Sun, Ronghua Liang, Huamin Qu, and Yingcai Wu. 2017. Embedding Spatio-Temporal Information into Maps by Route-Zooming. *IEEE TVCG* 23, 5 (2017), 1506–1519.

35. Sebastiaan J. van Schaik and Oege de Moor. 2011. A Memory Efficient Reachability Data Structure Through Bit Vector Compression. In *Proc. of ACM SIGMOD*. 913–924.

36. Renê Rodrigues Veloso, Loïc Cerf, Wagner Meira Jr., and Mohammed J. Zaki. 2014. Reachability Queries in Very Large Graphs: A Fast Refined Online Search Approach. In *Proc. of ICDT*. 511–522.

37. Elizabeth Weintraub. 2017. What Location, Location, Location Means In Real Estate. `https://goo.gl/ydJgKU`. (2017). [Online; accessed 07-Sept-2017].

38. Christopher Williamson and Ben Shneiderman. 1992. The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System. In *Proc. of ACM SIGIR*. 338–346.

39. Guojun Wu, Yichen Ding, Yanhua Li, Jie Bao, Yu Zheng, and Jun Luo. 2017. Mining Spatio-Temporal Reachable Regions over Massive Trajectory Data. In *Proc. of IEEE ICDE*. 1283–1294.

40. H. Wu, Y. Huang, J. Cheng, J. Li, and Y. Ke. 2016. Reachability and time-based path queries in temporal graphs. In *Proc. of IEEE ICDE*. 145–156.

41. Wenchao Wu, Yixian Zheng, Huamin Qu, Wei Chen, Eduard Gröller, and Lionel M. Ni. 2014. BoundarySeer: Visual analysis of 2D boundary changes. In *Proc. of IEEE VAST*. 143–152.

42. Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with knowledge from the physical world. In *Proc. of ACM SIGKDD*. 316–324.

43. Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: driving directions based on taxi trajectories. In *Proc. of ACM SIGSPATIAL*. 99–108.

44. Wei Zeng, Chi-Wing Fu, Stefan Müller Arisona, Alexander Erath, and Huamin Qu. 2014. Visualizing Mobility of Public Transportation System. *IEEE TVCG* 20, 12 (2014), 1833–1842.

45. Wei Zeng, Phillip Chi-Wing Fu, Stefan Müller Arisona, Alexander Erath, and Huamin Qu. 2016. Visualizing Waypoints-Constrained Origin-Destination Patterns for Massive Transportation Data. *Computer Graphics Forum* 35, 8 (2016), 95–107.

46. Jiawan Zhang, E. Yanli, Jing Ma, Yahui Zhao, Binghan Xu, Liting Sun, Jinyan Chen, and Xiaoru Yuan. 2014. Visual Analysis of Public Utility Service Problems in a Metropolis. *IEEE TVCG* 20, 12 (2014), 1843–1852.

47. Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban Computing: Concepts, Methodologies, and Applications. *ACM TIST* 5, 3 (2014), 38.

48. Yixian Zheng, Wenchao Wu, Yuanzhe Chen, Huamin Qu, and Lionel M. Ni. 2016. Visual Analytics in Urban Computing: An Overview. *IEEE TBD* 2, 3 (2016), 276–296.

49. Andy Diwen Zhu, Wenqing Lin, Sibo Wang, and Xiaokui Xiao. 2014. Reachability Queries on Large Dynamic Graphs: A Total Order Approach. In *Proc. of ACM SIGMOD*. 1323–1334.

50. Zillow Group Inc. 2017. Zillow: Real Estate, Apartments, Mortgages & Home Values. `http://www.zillow.com`. (2017). Online; accessed 07-Sept-2017.