Noah: A Dynamic Ridesharing System

Charles Tian, Yan Huang, Zhi Liu
University of North Texas
charlestian@my.unt.edu, huangyan@unt.edu, zhiliu@my.unt.edu

Favyen Bastani
MIT
fbastani@mit.edu

Ruoming Jin
Kent State University
jinruoming@gmail.com

Introduction

This demo presents Noah: a dynamic ridesharing system. Noah supports large scale real-time ridesharing with service guarantee on road networks. Taxis and trip requests are dynamically matched. Different from traditional systems, a taxi can have more than one customer on board given that all waiting time and service time constraints of trips are satisfied. Noah's real-time response relies on three main components: (1) a fast shortest path algorithm with caching on road networks; (2) fast dynamic matching algorithms to schedule ridesharing on the fly; (3) a spatial indexing method for fast retrieving moving taxis. Users will be able to submit requests from a smartphone, choose specific parameters such as number of taxis in the system, service constraints, and matching algorithms, to explore the internal functionalities and implementations of Noah. The system analyzer will show the system performance including average waiting time, average detour percentage, average response time, and average level of sharing. Taxis, routes, and requests will be animated and visualized through Google Maps API. The demo is based on trips of 17,000 Shanghai taxis for one day (May 29, 2009); the dataset contains 432,327 trips. Each trip includes the starting and destination coordinates and the start time. An iPhone application is implemented to allow users to submit a trip request to the Noah system during the demonstration.

Ridesharing Service

Despite high amounts of congestion and pollution in concentrated areas throughout the world, many private and public vehicles continue to travel with empty seats. The mean occupancy rate of personal vehicle trips in the United States is only 1.6 persons per vehicle mile. In this demonstration, we simulate and visualize the effects of taxi ridesharing on the city of Shanghai.

What is Ridesharing?

Real time ridesharing, enabled by low-cost geo-locating devices, smartphones and wireless networks, is a service that dynamically matches each taxi to a potential multitude of taxi requests. Assuming two requests have similar initial and destination locations, it is more resource efficient for one taxi to fulfill both requests with a limited amount of waiting time and route detour, rather than have two taxis each fulfill one request. In our demonstration simulation, we use an optimized kinetic tree algorithm to evaluate each taxi request with the available taxis. A taxi will fulfill the request if it satisfies all user-defined constraints.

Benefits:

- Reduce global energy consumption
- Conserve exhaustible natural resources
- Decrease road congestion

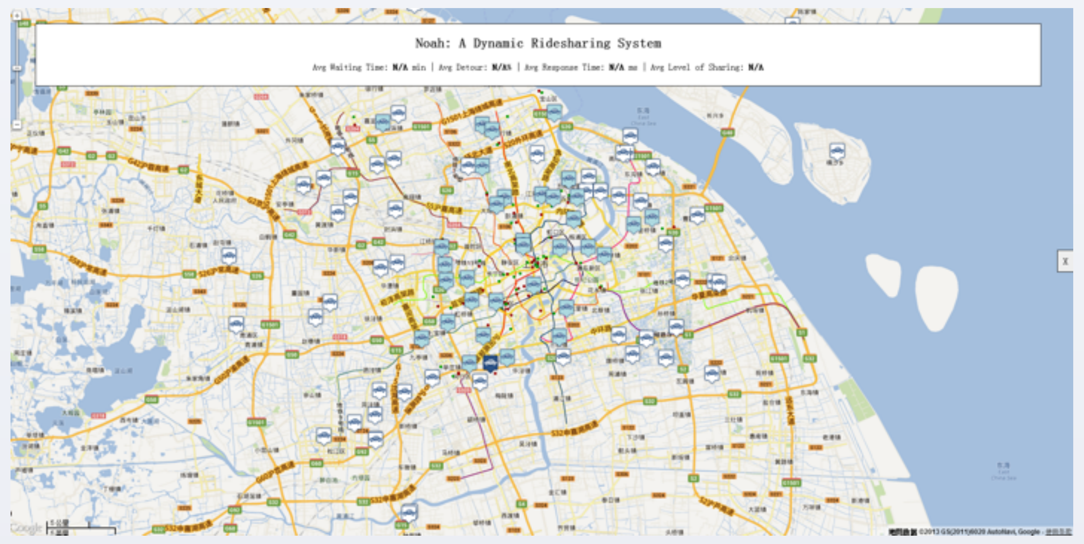
Compromises:

- Increased average waiting time
- Potential route detours

Providing a large-scale urban ridesharing service is a non-trivial issue. At the core of the implementation is a real-time matching algorithm that can quickly determine the best taxi to satisfy incoming service requests. Traditional approaches such as branch-and-bound or mixed integer programming are inefficient when dealing with a problem of this magnitude. Furthermore, most previous solutions focus on scenarios where all requests are known ahead of time. These methodologies are incapable of performing real-time ridesharing in enormous modern situations, where a new request can be made at any time.

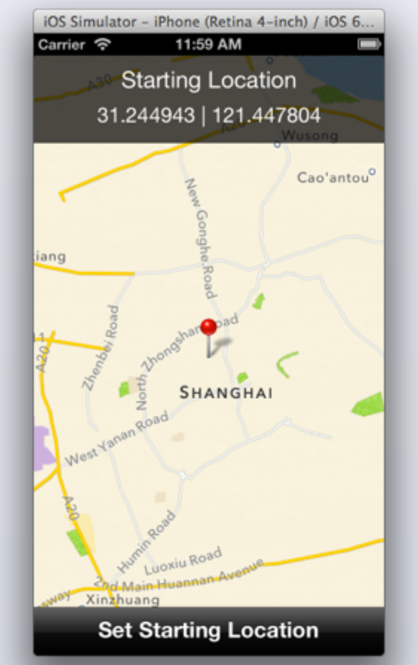
Demonstration

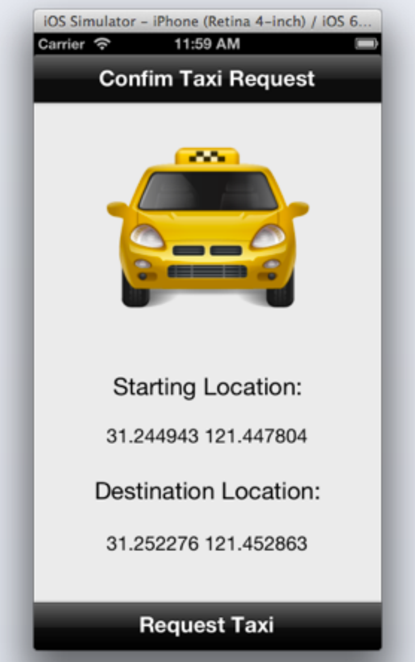
The Noah demonstration consists of three components: a web-based client, a backend simulator, and an iOS application. These three components interact with a MySQL database, reading and writing information to calculate and visualize taxi and request locations in real time. Noah simulates ridesharing for one hour in the city of Shanghai.



Web Client

The web client provides the demo attendee with a method of controlling the server-side simulation, as well as viewing taxi and request location information. Taxi and request information is displayed using Google Maps, and is synchronous with server output and will be updated in real time. Clicking on a taxi marker will display relevant taxi information, such as taxi ID, location, and number of passengers. Controls are also provided for pausing, resuming, slowing down, speeding up, and otherwise manipulating the simulation.





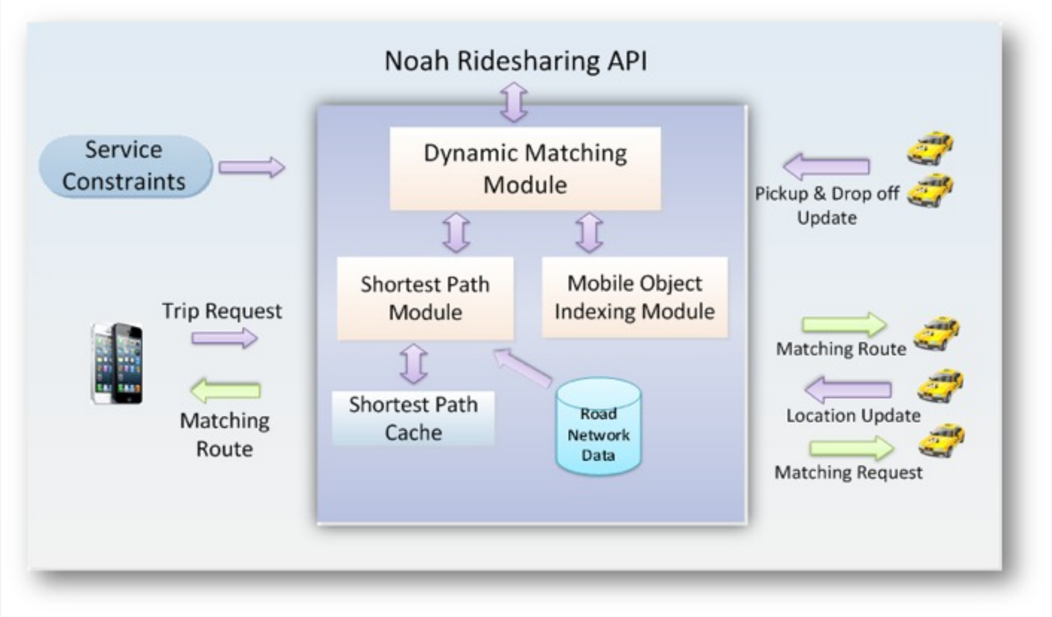
iOS Application

Using an iOS application, demo attendees are able to create and queue a new taxi request in real time, choosing the user's pickup location and intended destination on a map view and sending this information to the server. Once the request has been made, it will be placed in a pickup queue, and a taxi will be allocated to pick up the passenger. The iOS application will allow also allow the user to view relevant information regarding their taxi.

Server Simulation

The backend simulator time is kept in sync with the web client time, and taxi locations are updated once every simulator second. The simulator writes all taxi information to a MySQL database table, which is then read and visualized by the web client. The server also implements a system for handling a new request sent by the iOS application, and adding that request to the queue.

Architecture



The core components of the architecture of the Noah system include a Dynamic Matching Module, a Shortest Path Module, and a Mobile Indexing Module. When a taxi request is received, the Dynamic Matching Module will find a set of taxis that can serve the request and suggest the taxi that provides the shortest route to the passenger. The selected taxi will also be notified of the new request and and be provided an updated route incorporating the new trip. Taxi locations are updated once per simulator second, and taxis themselves update the Noah system whenever they pick up or drop off a passenger. Taxi locations are indexed by the Mobile Indexing Module, which helps the Dynamic Matching Module filter taxis that are clearly unable to fulfill a request. The Dynamic Matching Module implements various request-to-taxi matching algorithms that are are scalable with taxi and request quantities. A ridesharing matching algorithm typically involves many pairs of shortest path queries on a road network. In order to improve efficiency, the Shortest Path Module also implements a caching scheme that prevents repeated calculations of the same pair of shortest paths.

Algorithms

The primary challenge of large-scale ridesharing lies in determining the method in which trip requests are handled as they flow into the system in real time. A new request may be made in close proximity of hundreds of taxis. Some taxis will be full, and others may not be able to fulfill the request within detour and waiting time constraints. Noah implements several dynamic matching algorithms, allowing users to select different algorithms and observe system performance from the web client.

Branch & Bound

The branch & bound algorithm systematically enumerates all candidate schedules and organizes the candidates into a schedule tree. It estimates and maintains a lower bound of each partially constructed schedule and stops building candidate schedules that have lower bounds greater than the best solution found so far. The algorithm first expands the partial candidate with the lowest lower best (the best first search). The bound that Noah uses is the sum of the minimum-cost edge incident to the nodes that are not yet scheduled.

Mixed-Integer Programing

Mixed-Integer Programing is a popular alternative. Noah divides the location indexes of pickup and drop-off locations into the following subsets:

- Drop-off locations of passengers already picked up
- Pickup locations of requests not yet started
- Drop-off locations of trips not yet started

Constraints are coded such that schedule structure is enforced. That is, each node is visited exactly once. The waiting and service requirements that users select from the web client are programmed as constraints. Traditional solvers are used by noah to determine taxi routes.

Kinetic Tree Algorithm

The two aforementioned approaches both reschedule the unfinished pickups and drop-offs with each new request from scratch, without re-using the computations performed before. The structure of the two algorithms make them difficult to adapt to the dynamic nature of large-scale ridesharing, and thus makes them inefficient. Noah implements a kinetic tree structure that can maintain and update the calculations performed up to the current simulator second, and use these calculations effectively when a new request is issued.

Noah's kinetic tree algorithm also optimizes upon the basic tree algorithm, which would explode in size when there are multiple pickup or drop-off locations in close proximity with one another. Our kinetic tree algorithm uses slack time and hotspot clustering optimization, implementing an approximation approach with bounds to reduce the search space of taxis fulfilling a request.

Out of the three algorithms Noah uses, our optimized kinetic tree algorithm is the most efficient in handing taxi requests made by passengers on such a large scale.

Conclusion

This demonstration visualizes real-time ridesharing on the Shanghai road network, dynamically matching real time trip requests with available taxis. Our kinetic tree algorithm implements slack time and clustering optimizations, and is able to outperform common approaches to large-scale ridesharing, such as brand-and-bound and mixed-integer programing. Through our web client, users are able to control the ridesharing simulation in real time, view live algorithmic performance statistics, and observe animated request and taxi location information based off predetermined simulation parameters.

References

[1] K. Ghoseiri, A. Haghani, and M. Hamedi, "Real-time rideshare matching problem," Final Report of UMD-2009-05, U.S. Department of Transportation, 2011.

[2] G. Gidofalvi, T. B. Pedersen, T. Risch, and E. Zeitler, "Highly scalable trip grouping for large-scale collective transportation systems," in Proceedings of the 11th international conference on Extending database technology: Advances in database technology, ser. EDBT '08, 2008, pp. 678–689.

[3] D. Delling, P. Sanders, D. Schultes, and D. Wagner, "Algorithms of large and complex networks," D. Lerner, J.and Wagner and K. Zweig, Eds., 2009, ch. Engineering Route Planning Algorithms.

[4] I. Abraham, A. Fiat, A. V. Goldberg, and R. F. Werneck, "Highway dimension, shortest paths, and provably efficient algorithms," in SODA '10, 2010.

This work was partially supported by the National Science Foundation under Grant No. IIS-1017926.

UNT UNIVERSITY OF NORTH TEXAS · MIT · KENT STATE UNIVERSITY