Taxi service automation

A. Usov, A. Anikina, V. Stepanov

This paper describes the system of taxi service automation, which is based on the client-server model. Server represented as Web service. The client part is divided into two mobile applications for the Android OS: applications for the passenger and drivers. The server stores data about each vehicle and distributes orders between taxi drivers. Application for driver is used to accept orders and display the optimal route on the map. With the application for passenger customers can order a taxi.

Introduction

The most common application of computer technology and new software aimed to automating processes associated with manufacturing, accounting, communications and so on. Processes that were previously performed by man, but now we can create software that will do all the hard work for us.

The aim of this work is to develop a system for taxi service that will automate the dispatcher work, simplify calling a taxi via mobile applications, speed up the process of selecting appropriate car, optimize the usage of car resources by assessing road load factor, display the map of traffic jams on driver‘s application, lead detailed statistics of service and so on.

Modern taxi in Ukraine have the following disadvantages:

- Non-optimal usage of car resources;
- There is a need to make a phone call to take a taxi, which can’t be comfortable sometimes;
- Taxi company have to handle a large number of dispatchers;
- The long process of selecting appropriate car;
- Taxi driver is often distracted by excessive information.
- conclusions;

Concept of the following system solves all these problems, making it relevant and unique in its own way.

Server side

Server part of the system should receive and process requests from clients. There are three types of clients:

- *Users that want to order a taxi.* Server must be able to receive request from the client, find the best contender among drivers who will serve the client and alert driver about new order. To choose the best contenders, server must provide the estimates of velocity on particular roads at the moment.
- *Drivers of taxi service.* Server must be able to poll drivers whether they can server an order. Also server must provide drivers traffic information.
- *Volunteers that send and receive information about traffic.* Server must be able to process traffic data and store results in the database.

Due to the mentioned requirements the server part of the system was implemented. It consists of such parts:

- Web-services for data exchange with clients.
- Taxi drivers choosing module.
- Statistical analysis module.
- Database containing roadmap and estimates of velocity on particular roads.
- Asynchronous messaging between different modules.

Statistical analysis module

One of the most important conditions for building efficient taxi management system is traffic analyzing. The main aim of the statistical analysis module is to give an estimate of velocity for any road anytime.

This idea was realized by creating users-volunteers. They send information about average velocities on the roads. This information is gathered using GPS sensor, that's why it is necessary to bind sensor readings to the map in the database.

Suppose that data from clients-volunteers — their trajectory – is divided into tracks. Database stores the roadmap as the set of ways and nodes — intersections.

The following algorithm for such binding is proposed:

1. Find the region (at the moment this area is a rectangle) that covers all probable ways-candidates.
2. Extract ways from database that are located in this area.
3. For every client's track find the way or the chain of ways with the smallest proximity measure to the current track.

For every way database stores such parameters:

- Allowed speed.
- The last known velocity on the way and its timestamp.
- Estimates of velocities for every hour since 6 till 23 o'clock.

To calculate estimates of velocities we can just calculate the average of all velocities, received from the clients. But this method has disadvantages like long memory and using additional fields in the database.

That's why it was proposed to use the following method. Suppose that every new value has the constant weight $1/N$. Then the new estimate is calculated as

$$V_{new} = V_{old} + V_{current} - V_{old}/N$$

where V_{old} is an old estimate, V_{new} is a new estimate of velocity on the way, $V_{current}$ is the value, received from client.

The process of estimate evaluating for particular way can be described as follows:

1. If the timestamp of the last known velocity is actual, return the last.
2. If the database contains needed estimate, return it.
3. Otherwise return the maximal allowed velocity.

Proximity measure

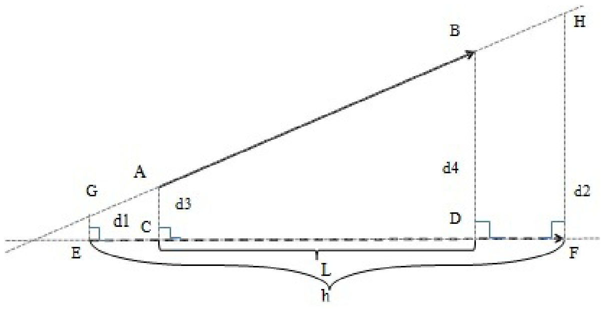

Figure 1. Track doesn't cross the way

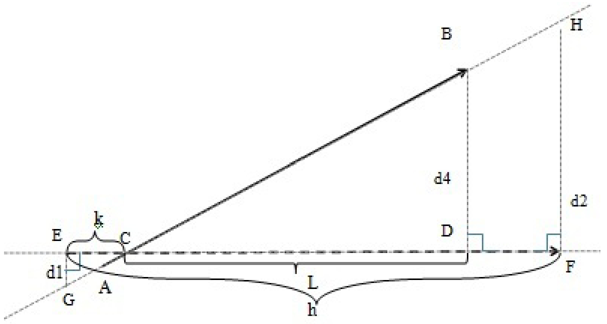

Figure 2. Track crosses the way

The following algorithm to figure the proximity measure is proposed:

1. If the angle between way's vector (EF) and track's vector (AB) is less then 90 degrees return infinity.

2. Project track AB onto a line that contains the way EF. Find the part of the projection that lies between the vertices of the way E and F.
3. Build trapeze $CABD$, calculate it's midline $(AC + BD)/2$.
4. Calculate proximity measure using the following formula:

$$\mu = S \times \frac{h}{l} \times \frac{d3 + d4}{2}$$

, where S is an area of trapeze $EGHF$ (Fig. 1) or two triangles $\triangle ECG$ and $\triangle CBD$, if the track crosses the way (Fig. 2), h is the length of way EF, l is the length of projection CD, $d3$ and $d4$ are the distances from points A, B to the line EF.

Note: if the track crosses the way, S can be calculated as:

$$S = \frac{1}{2} \times h \times \frac{d1^2 + d2^2}{d1 + d2}$$

Otherwise:

$$S = \frac{1}{2} \times h \times (d1 + d2)$$

Taxi drivers choosing module

Let the taxi station has n vehicles. Among them have to be selected k vehicles, which satisfy client's request the best. The aim is informing vehicles, which are closest to the chosen location.

To solve such problem was chosen algorithm A^* which belongs to heuristic search algorithms[2]. A^* finds a least-cost path path between two graph vertex with positive weights of edges. The algorithm uses an auxiliary function (heuristics) to guide the search direction and shorten its duration. Algorithm is complete in the sense that it always finds the optimal solution, if it exists. It uses a distance-plus-cost heuristic function of node (usually denoted $f(x)$) to determine the order in which the search visits nodes in the tree. The distance-plus-cost heuristic is a sum of two functions:

- the path-cost function, which is the cost from the starting node to the current node (usually denoted $g(x)$)
- an admissible "heuristic estimate" of the distance from to the goal (usually denoted $h(x)$).

Function $h(x)$ must be an admissible heuristic; that is, it must not overestimate the distance to the goal. At each step, A^* reviews all paths from the initial vertex to the end, until it finds a minimum. Like all "informed search algorithms", it looks at first those routes that "seem" leading to the goal.

Client side

The client side consists of three types of applications for: passenger, taxi driver and volunteer. Passenger application will send taxi orders to Server. A passenger has the ability to select the departure and destination points and type of the vehicle. After the Server selects a car, passenger will receive a message with information about the selected car.

Volunteer application will give an opportunity to see the map of the city with information about traffic jams on user's mobile device. Also, this application will gather information about the speed of the car on the roads.

Application for taxi driver has the same functionality as a volunteer app, but allows to accept or reject orders and evaluate passengers additionally.

Further an algorithm that allows to gather information about road conditions will be described. The task is to break the trajectory of the car to the straight sections and send data about the time of driving through these areas. As we don't have the whole data sample, but it builds gradually, so the local task is to determine the critical point where the car starts to turn. The algorithm, based on linear regression analysis[3] was developed for this.

Description of the algorithm

GPS coordinates are tracking down during the whole work of the application. Once the coordinates of the vehicle changed, their values stored in the list of coordinates to be processed. These actions occur with some frequency.

The algorithm, which processes the coordinates starts when the list has more than ten items.

We have to determine longitude as a free variable, and latitude - as dependent for linear regression. Then the following steps are performed:

1. A thread-safe lists A and B of size $n - 10$ is creating, where n - number of items that need to process;
2. In the cycle where $i = 0..(n-10)$, set of items is divided into two subsets of $5 + i$ and $n - 5 - i$ first points:
   - For each subset two threads are creating;
   - Each thread builds regression model for appropriate set of points $y_i = f(\omega, x_i) + \epsilon$, where ω - vector of unknown parameters, ϵ - additive random variable. We will seek the vector of unknown parameters by the least squares method. Because we have to build a straight line, so we define the dependence model, as $y_i = \omega_1 + \omega_2 x_i + \epsilon_i$. According to the method of least squares, the desired vector of parameters $\omega = (\omega_1, \omega_2)^T$ - is the solution of the normal equation: $\omega = (A^T A)^{-1} A^T y$, where y - vector, which consists of the values of the dependent variable $y = (y_1, ..., y_m)$. The columns of the matrix A is a free variable values substitution $x_i^0 \to a_{i1}$ and $x_i^1 \to a_{i2}$. The matrix has the following form:

   $$A = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \multicolumn{2}{c}{\ldots\ldots} \\ 1 & x_m \end{pmatrix}$$

   Dependent variable is restored from obtained parameters and given values of free variable $y_i^* = \omega_1 + \omega_2 x_i$. Criterion of sum of squared regression errors (SSE) is used for assessing the quality of model:

   $$SSE = \sum_{i=1}^{m} (y_i - y_i^*)^2 = (y - y^*)^T (y - y^*).$$

   - Each thread writes value of the considered regression remainder of their set to the list A and found regression model parameters to the list B;
3. When all threads will complete their work, such pair of subsets are chosen, which have the least sum of regression remainders;
4. In the result we have equations of two lines: $y_1 = \omega_1 + \omega_2 x_1$ and $y_2 = \omega_1' + \omega_2' x_2$. Options ω_2 and ω_2' - are the angular coefficients and $tan^{-1}\omega_2$ and $tan^{-1}\omega_2'$ - angles of slope. This lines are the approximate trajectory of the vehicle;
5. Then the angle between the lines are found. If the tangent of the angle is greater than k, then we can conclude that the car made a turn. The value of k was selected experimentally. It equals 0.785;
6. All points from the first subset lie on a straight part of the road. The first and the last point of the first subset are sent to the server. Before that we find the difference between the time when these coordinates were token from GPS sensor. This difference will be the time, what the car spent to pass this segment of road;
7. The first subset are removed from the total set of points;
8. Items 1 - 7 are repeated until the new coordinates are coming.

Conclusion

As the result, using described algorithms, taxi automation system was built. This system fully solves all the problems that have been mentioned in the introduction. With further development of the project we plan to develop custom applications for all popular mobile platforms, such as IOS, Windows Phone and Symbian. Also we plan to add rating system for drivers and passengers, and an effective communication between server and drivers. We are going ahead to improve the system into a real taxi service.

References

[1] Mathematical Statistics with Applications Dennis D. Wackerly,William Mendenhall,Richard L. Scheaffer 2008

[2] Lauriere J.L. Artificial Intelligence Systems/from french 1991.

[3] John Wiley & Sons, Applied Linear Regression, Sanford Weisberg, 2005.

[4] Android developer‘s guide http://developer.android.com/guide/index.html

Authors

Andrii Vasylovych Usov — the 4th year student, faculty of cybernetics, Taras Shevchenko national university of Kiev, Kiev, Ukraine; E-mail: *andreusov13@gmail.com*

Oleksandra Volodymyrivna Anikina — the 4th year student, faculty of cybernetics, Taras Shevchenko national university of Kiev, Kiev, Ukraine; E-mail: *saneshka2509@gmail.com*

Vladyslav Valeriiovych Stepanov — the 4th year student, faculty of cybernetics, Taras Shevchenko national university of Kiev, Kiev, Ukraine; E-mail: *shto.shto.nu.horosho@gmail.com*