Modified BUSTRAP: An Optimal BUS Travel Planner for Commuters using Mobile

Farhan Razi Siddiqui¹ & Ajeet Kumar Pandey²

¹Department of Computer Science & Engineering, Integral University, Lucknow, India
²Reliability Engineering Centre, IIT Kharagpur, Kharagpur, India

Abstract: In this paper, we present an enhanced, efficient and reliable travel planner named Modified BUSTRAP (BUS TRavel Planner). Modified BUSTRAP can be used for planning a bus travel in two well-known metro cities of India. Locations of buses are recorded at a few instances on the roads through sensors. The Planner finds their current and future locations based on these records. We have considered changeover points on a route according to which a route plan would be formed. This helps commuters to break their journey and change/board to another bus to reach their destination. The path from source to destination is found by A* Algorithm, which gives an optimal path. Database Caching has been used to reduce the response time for frequently used queries. The solution employs mobile communication facilities like SMS for proper guidance on the route plan for a given destination and source of the mobile user.

Keywords: Travel Planner, SMS, Graph Search, Caching, and Dynamic Attribute

1. INTRODUCTION

In this paper, we present a travel planner named Modified BUSTRAP (BUS Travel Planner). A detailed presentation of Modified BUSTRAP is made in [6]. This system can be used for planning a bus travel in metropolitan cities¹.

Public transport system is an essential and important part of metropolitan cities of India like New Delhi [3] and Mumbai [1], where traveling through city bus is very time consuming. There are so many ways to reach a desired place, but commuters do not have knowledge about these alternate routes. Also, buses cannot follow their schedules due to heavy traffic, traffic jam and breakdown.

Throughout the work, we have taken two well-known metropolitan cities of India (New Delhi and Mumbai) as reference model.

This leads to increase waiting time at bus stops. Due to lack of information most people select non-optimal route to reach their desired destination. To know the information about city buses one have to consult the person sitting on the enquiry. The person sitting on enquiry does not have any exact idea about the current location of bus. Therefore in this paper we present such a planner to overcome above problems.

The BUSTRAP came into existence in May 2004 [7]. This paper presents its enhancement here after written as Modified BUSTRAP. The Proposed Modified BUSTRAP system provides comparatively more accurate information about optimal way to reach a particular destination.

Commuter can send an SMS giving about source and destination. The system will give out optimal ways to reach that place by bus, displaying exact bus route numbers and bus schedules according to the bus movement on the city roads at that time. A* algorithm [10] based on graph search procedure has been used to find the shortest path quickly. Database caching [8] has been used to reduce the query processing and response time. The rest of the paper is organized in the following way:

In section 2, we discussed about related works in this domain. In section 3, we present design methodology of Modified BUSTRAP. We have discussed implementation in section 4. We have made comparison of previous and current work in section 5. We have concluded in section 6 by concluding the work and proposing possible future directions in this work.

2. RELATED WORKS

A lot of efforts have been made in the area of query processing in database. Jain et al [16] developed a database for moving vehicles, which contains information about moving objects. The queries are submitted from stationary or moving users. The database is used to retrieve the current location of moving vehicles. These databases are also called Moving-Object-Database (MODs). An example of MOD can be found in [16].

Culler and Hong [2] give the basics of wireless sensor network. These networks are very useful for solving problems in various domains like military, health, transportation etc. Sensors provide easier way to record periodic locations of moving objects. Sensors are generally

*Corresponding Author: ¹frs_lko@rediffmail.com, ²ajeet.mnnit@gmail.com
fitted with on-board processor and capable of performing simple computations. They generate signals for programs running on server to update MOD.

In a study conducted by Garmin Corporation [17], about GPS (Global Positioning System) indicates that it is mostly used in sensing the location of objects. It works in any weather condition at any place. GPS receivers are very accurate due to their parallel multi-channel design. Gupta Sandeep [7] has developed a metropolitan bus travel planner (BUSTRAP) to find out current and future locations of buses. We found that the model was focusing on a general metro city and were not considered the real world data. We enhanced this system for two well-known metropolitan cities of India viz. New Delhi [3] and Mumbai [1].

The next section describes the design methodology for the system.

3. DESIGN METHODOLOGY

In this section, we briefly describe design of Modified BUSTRAP on the basis of BUSTRAP [7]. In [6] detailed design description of Modified BUSTRAP has been provided. The system has been reproduced with slight modification. This design discusses about using sensors for recording location of buses but underlying technology should not affect the working of the system. Following assumptions have been made prior to setup of Modified BUSTRAP:

- All routes are bi-directional, i.e., if a route exists from source to destination, then a route also exists from destination to source.
- All buses have, a fixed starting times, which they always follow. If schedule is changed, it gets reflected in the database.
- Some buses stops only on selected stops. These buses have been considered as “Express buses”. Ordinary buses stop on all the bus-stops.

The setup has been taken from BUSTRAP [7]. We have not modified the proposed setup. It is reproduced here for the purpose of clarity.

“A city can be divided into five zones as shown in figure 1”. These zones are north, south, west, east and central (Other zones, can also be introduced according to the requirements and geography of the city).

Each zone will have sub-zones with bus-terminals at center of each sub-zone. Bus-terminals would be responsible for maintaining data regarding buses in that sub-zone. Sensors are fitted at some selected locations, throughout the city. These sensors would be used to keep track of buses. It has been assumed that these sensors are placed near to bus-stops/bus-terminals. Similarly, buses also fitted with sensors. So, whenever a bus approaches a sensor, a signal is generated and current bus-position gets recorded at an appropriate place. Hence, a sensor network [2] would be formed for tracking buses.

3.1. Database Management and Design

We are proposing a distributed database approach because movement of buses is generating a lot of data. Handling huge amount of data traffic would become easier with distributed approach. There would be one central database at a central server. This server would be responsible for resolving queries. The central database would consist of static data, whose values do not change frequently, like the city graph, bus-routes and actual distance. Non-static data would be taken from sub-zonal databases. These sub-zonal databases would reside at bus-terminals. Last recorded location of buses in a sub-zone can, thus, be found in respective sub-zonal database. The query processing has been improved with the help of cache which contains frequently used queries. Cache-oblivious model [15], which contains multilevel memory hierarchy, can be further used to achieve the consistent and good performance. If the frequency of updating the database is not expected to be high, a centralized database and a single server can be used. We have used a centralized database, a single server and single level memory for implementation purposes. Real data have been used for testing. The real data corresponds to the DTC [3] and BEST [1] buses of two well known metropolitan cities of India.

Following tables describe the database for Modified BUSTRAP:

- **Bus-route**: Contains details of bus stops covered on a specific route. This also shows the stop sequence on that route.
- **Bus-schedule**: Contains data about route no’s, start time and direction of buses.
- **Bus-stop**: Contains data about all bus stops of city with their names and their id’s. All changeover points are identified in this table. “A changeover
point is a bus stop where bus-routes from different directions converge”. Commuters can get down at a changeover point and board a different bus. So, changeover points are useful for presenting plans, which include break-journey.

- **Bus Location**: Last recorded location of buses can be found out from this table.
- **Cache**: A table that contains the detail of routes queried frequently by commuters.

### 3.2. Calculating Current and Future Bus Locations With ‘Weights’

We have used dynamic attribute function concept [11] for calculating current and future positions of buses. It helps us to present the travel plans effectively. Dynamic attribute concept proposes that location of an object is represented by a motion plan containing speed and direction. As the object moves, its dynamic attribute also changes. For calculating current and future positions of buses the weights between each pair of bus stops (that are connected) are calculated. These weights are equal to number of minutes that a bus would take to reach one stop from the other. These weights can, then, be used for finding position of buses. It is assumed that all buses that ply on overlapping routes will take same amount of time to cover same distance.

For calculating initial weights, following formula can be used:

\[ w_{ij} = t_{day} \times \frac{d_{ij}}{A_{ij}} \]

where,

- \( i \) & \( j \) are two connected bus stops, \( t_{day} \) is time of the day parameter. It will adjust weight, \( w_{ij} \), according to the traffic during that time, \( d_{ij} \) is the distance between \( i \) and \( j \) and \( A_{ij} \) is equal to average speed between stops \( i \) and \( j \) when \( t_{day} \) is equal to 1.0.

### 3.3. Algorithm to Find Plans to Reach Destination from Source

Following algorithm has been taken from BUSTRAP [7]. It has been reintroduced in Modified BUSTRAP with slight modifications. It is used for making plans to reach a destination from a source at a given time.

\((t \) is a predetermined value, which is equal to number of next time units for which plans would be prepared\)

1. User sends source, \( s \) and destination, \( d \).
2. Determine the shortest route from \( s \) to \( d \) (using Graph Search Algorithm-2.4).
3. Identify \( c \) changeover points in the route.
4. Break the complete route into \( c + 1 \) sub-routes with \( s1 \) to \( c + 1 \) starting points and \( e1 \) to \( c + 1 \) ending points (where \( s1 = s \) and \( ec + 1 = d \)).
5. Let \( i = 1 \), \( et = 0 \) (\( et \) is a variable which is used in step 3 of procedure plan).
6. To retrieve buses for finding route plans).
7. Call procedure plan\((i, et)\).
8. Call procedure plan\((i, et)\).

**procedure plan\((i, et)\)**

1. If \( i > c + 1 \), display plan, along with expected arrival time of buses at each changeover point; return.
2. Identify bus routes that run from start point, \( si \) to end point, \( ei \).
3. For buses that run on above routes, retrieve locations of buses that are either running and are \( et \) to \( et + t \) time units away from \( si \) or are scheduled to start and are expected to reach \( si \) within next \( et \) to \( et + t \) time units.
4. For each such bus, \( bj \), selected in step 3.
   a) Calculate \( etbji \), estimated time that bus, \( bj \), would take to reach starting point, \( si \), from now.
   b) Calculate estimated time, \( esti \), to reach \( si + 1 \) from \( si \).
   c) \( etbji + 1 = etbji + esti \)
   d) If next sub-route is also served by same bus
      i) \( i = i + 1 \);
      ii) If \( i < c + 2 \), go to step 4(b); Else, display plan, along with expected arrival time of buses at each changeover point.

Else, call procedure plan\((i + 1, et + etbji + 1)\).

**procedure cache()**

1) Store the recently generated new plan in cache table.
2) If cache is full and there is new plan then remove least recently used plan from cache table and update the table.

The above algorithm will display plans telling bus numbers to take and expected arrival time of buses at source and changeover points. The shortest route from source to destination is found out. This route is, then, divided into sub-routes by identifying changeover points. Next, buses that will reach the identified changeover point at around the calculated time are found out. Similar procedure is followed till all sub-routes are covered and all plans are found out.

### 3.4. A* Algorithm to Find the Shortest Path

The A* algorithm [10] determines the shortest path from start node to end node. It uses graph search procedure and an evaluation function. The evaluation function is

\[ F(n) = g(n) + h(n) \]
Where \( g(n) \) is the cost of optimal path from source node to intermediate node and \( h(n) \) is the cost of optimal path from that node to destination.

The function \( h(n) \) is based on heuristic approach. Therefore it reduces the search effort and gives the optimal path in short time. It also reduces the time complexity of algorithm.

3.5. Database Caching

The performance of the system has been improved by using database caching [8]. Data stored in a database cache is accessed by the application using database queries in just the same way the backend database is accessed. The database caching provides scalability and reduces response time while executing the query. The system becomes more faster while using database caching [8].

This section discussed about techniques, which can be used to implement our system. The next section shows the implementation of the system.

4. IMPLEMENTATION

The Modified BUSTRAP simulation uses C program and MySQL database [9]. GUI has been developed in Qt 3.0 [12], which uses C++. All other modules have been written in C using MySQL API [13].

Figure 2: Simulation Setup

The block diagram of different simulation modules is shown in figure 2. The database of the simulation can be created either using MySQL commands or through ‘Database populator’ module. A ‘Simulator’ module has been taken from [7] and modified according to real life case. It works in close association with another module ‘Updater’. The simulator reads the database, generates signals and sends it to the ‘Updater’ in a way that would be similar to signals being generated by sensors in the real world. The ‘Updater’ receives the signals and takes necessary actions to update the database for running buses at a particular moment. It is based on ‘Dynamic Attribute’ concept [11]. The GUI (figure 3) displays the city map (in terms of bus-stops). It receives signals from the ‘Simulator’ module, according to which it displays movement of buses.

Figure 3: A Snapshot of GUI

The ‘PhoneSim’ sub-module is an interface for users to send an SMS to find out route plans. The ‘Query Resolver’ module finds out the route plans by using the above mentioned algorithm.

5. COMPARISON WITH PREVIOUS WORK

BUSTRAP [7] was based on hypothetical city and contain arbitrary data. Modified BUSTRAP contains real data in database tables and it consists of two real life case studies on public transport system of two well known metropolitan cities of India. In BUSTRAP [7] the algorithm to generate plans was based on Dijkstra’s algorithm to find the optimal routes from source to destination. Modified BUSTRAP uses A* algorithm [10] to find the optimal routes. The time complexity of A* algorithm [10] depends on the heuristic. In the worst case, the number of nodes expanded is exponential in the length of the solution, but it is polynomial when the heuristic function \( h \) meets the following condition:

\[
|h(n) - h^*(n)| = O(\log h^*(n))
\]

where \( h^* \) is the optimal heuristic.
the query processing. We present the Comparison between cached and non-cached database figure 4.

6. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a system that will help commuters to plan their travel for metropolitan cities New Delhi [3] and Mumbai [1]. Although, there are some systems which track public transport buses to check whether they follow to their schedules properly, there is not a single system that directly helps commuters in planning their travel. We believe that having such system would improve the overall experience of commuters and would drastically reduce waiting time at bus stops. Such a system demonstrates use of Information Technology for the society. The Mobile Communication Network [14] helps the system to transmit the information globally. It can also be used for planning travel by other public transport systems like metro rail and ‘local’ sub-urban trains. Integrating the metro railways and bus transport, can enhance Modified BUSTRAP. It can be further extended to other metro cities. Further, one can extend Modified BUSTRAP by introducing certain features in it, like reservation of seats in across all relevant buses for a particular plan, which can prove highly useful for commuters. Introducing the feature of GPRS can further enhance the system.

REFERENCES