Genetic Algorithm Based Dynamic Route Planner for Public Transport

Abhishek Basu
Former undergraduate student,
Department of Civil Engineering
Indian Institute of Technology Madras,
Chennai, Tamil Nadu 600036
Tel:+91 - 9600153209
Email: abhishek.sfy@gmail.com

and

Lelitha Vanajakshi\(^1\)
Associate professor,
Department of Civil Engineering
Indian Institute of Technology Madras,
Chennai, Tamil Nadu 600036
Tel: +91 - 44 - 2257 4291
Email: lelitha@iitm.ac.in

Paper revised and submitted for presentation and publication in
Transportation Research Record, Transportation Research Board,
National Research Council, Washington, D. C.

Word Count: Body text (5728) + Figures (1000) + Tables (750) = 7478.
Submitted on: November 15, 2015

\(^1\) Corresponding Author
Genetic Algorithm Based Dynamic Route Planner for Public Transport

ABSTRACT
This paper aims to address the issue of unreliability associated with the public transportation system and hence looks at developing a robust and accurate route planner using Global Positioning System (GPS) data recorded from individual buses. Real-time data from the buses of Metropolitan Transport Corporation (MTC) Chennai, were used for demonstrating the results. With the help of such a route planner, users can make informed decisions and they can optimise their schedule in the desired way. Genetic Algorithm (GA) is a robust algorithm, which provides good solutions within computational time constraints for the chosen problem, and hence has been used in this study. The developed route planner was initially tested for static networks, and then extended to real-time data. In order to speed up the GA based framework, the algorithm was segmented and a temporal cache was introduced. The performance of the developed framework was found to be satisfactory showing a great scope for scalability and citywide implementation of the solution for real-time route planning.
Genetic Algorithm Based Dynamic Route Planner for Public Transport

INTRODUCTION AND BACKGROUND
Traffic congestion is a serious problem faced by many urban areas in India. A solution is to attract more travellers to public transport services. However, public transportation in India is unreliable as buses often do not adhere to the schedule due to delays caused by congestion. A route planner for bus transportation system would try to solve this reliability issue to a certain extent. A bus-based trip-planner would help passengers identify the correct routes to be taken with minimum stopovers or waiting or walking times and thus would lead to increased usage of public transportation system. The parameters to be minimised may vary according to personal preferences.

Depending on the data used, a route planner can be static or real-time. A static route planner can help a user plan the trip beforehand, whereas a real-time route planner can be used for updates during the trip and this would take into account the variation in bus schedules. Hence, an ideal route planner should operate at both static and real-time levels to be effective as a solution. The present study aims to develop a trip-planner taking Chennai Metropolitan Transport Corporation (MTC) as the case study. Real-time tracking data collected using onboard GPS units were used. An existing travel time prediction algorithm (1) was used in addition to GPS data, for the real-time trip-planner.

Various approaches have been reported for route optimization of multi-modal travel problem Genetic Algorithm (GA) is one of them. Genetic Algorithms are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and genetics. As such they represent an intelligent exploitation of a random search used to solve optimization problems (2). GA has been used in a host of optimization and search problems. Some of the basic terms associated with GA are explained as follows. The space of all feasible solutions to the problem represents the search space. Each point in the search space represents a unique solution that could be marked by its fitness for the problem. Fitness is an attribute defined as per the problem’s objectives, which distinguishes one solution from the other in terms of its ideal nature. Each solution obtained from GA is represented as a string of elements (values) in a particular order, and is called a chromosome (3). Each element that constitutes the chromosome is called a gene. The specific representation of a chromosome is termed as encoding. Genetic operators are applied on successive generations of population, which is the set of chromosomes generated at a particular iteration, in an attempt to increase the fitness of individual solutions. Selection, crossover and mutation are three commonly used genetic operators. Genetic algorithms have been often used in bus network optimization problems and in shortest path applications.

Some of the relevant literature in this area are discussed here. Nanayakkara et al. (4) implemented a genetic algorithm based route planner for large urban street networks. Each chromosome consisted of a sequence of positive integers that represented the IDs of nodes, which constituted the path. The initialisation step proposed in the paper was to simultaneously search for paths from both origin and destination. It was observed that the population size has a huge impact on the execution time. It was also observed that the population size depended upon the complexity of the route. Greater the complexity, higher is the required population size. It was tested on a Singapore based network (10,000 nodes) and was shown to outperform the ant-based algorithm, which they tried out as an alternate algorithm.

Nilesh et al. (5) developed a real-time trip-planner that used GPS data from buses and found out multi-criterion optimal paths using K-shortest path algorithm. Breadth First Search
(BFS) algorithm was used to search the travel plans. GPS data from only two routes were utilized in presenting the results. Rahim et al. (6) used GA for time-dependent personal tour planning and scheduling. This was solved using GA to search the solution of the shortest multi-modal path problem. They encoded the chromosome in a non-binary fashion, with alternate genes representing node and mode. During initialisation, randomly generated genes were appended sequentially until the destination node was reached, to construct a chromosome (path). The algorithm was terminated, if the temporal differences between fifteen paths with best costs in the successive iterations reached a certain level (chosen as zero). The algorithm was successfully implemented for the city of Tehran, and about 400 test cases were evaluated. Chen et al. (7) explored the idea of a multi-modal trip-planner, however it did not consider real-time scenario. Jariyasunant et al. (8) developed an algorithm to calculate the travel times of K-shortest paths in a public transportation network, and all wait and travel times were known at real-time. The precomputation of paths was shown to speed up the solution generation. Kumar et al. (9) showed that the solution obtained should satisfy the minimum criteria of reaching a fixed number of generations and optimal budget allocation (time and money) and that no successive iterations should produce better results.

Route optimization of multi-modal travel being a NP-hard problem and it is difficult to get global optimal solution by exact algorithms in an acceptable amount of time (10). Genetic algorithm was adopted in this paper for this purpose, since they were reported to have worked well for time-constraint shortest path problems in earlier studies.

**METHODOLOGY**

Due to the advantages offered by GA, it was adopted to develop the framework for the route planner presented in this paper. The encoding scheme used is as follows: each odd gene represents bus stop (or any other boarding location) and the even genes represent the route IDs of the vehicles that are used to traverse from one bus stop to the other. Each step of the genetic algorithm was modified to suit the computational needs of the problem in hand. The basic structure of the algorithm is as follows: the first step involves initialisation of the population, followed by evaluation of their fitness value and ranking them. The second step involves application of genetic operators. The invalid chromosomes are repaired to make them feasible, followed by evaluation and ranking. The second step is repeated until the termination criteria is satisfied.

Broadly, the GA could be divided into two steps, the initialisation step and the mating step. In the initialisation step, the chromosome is constructed starting from the source, after which a random node adjacent to the source and a corresponding mode of travel are chosen. This process is continued further with the newly reached node and so on, until the destination is reached. The length of a chromosome depends on the number of intermediate nodes in each path. Thus, its limit is fixed at the beginning based on the maximum number of transfers that the user is willing to take. As a particular node is reached, it is added to the scan list to keep track of all the nodes that have been visited, thereby avoiding loops. The initialisation step may sometimes never reach the destination. For example, during the initialisation process if a node from which no arcs exist was reached, or if all nodes adjacent to this particular node were already present in the scan list, the algorithm would get stuck in a loop and the solution gets trapped at that particular node. To avoid this, a counter is maintained, which counts the number of random attempts to select an adjacent node, and the partial solution is discarded upon exceeding the limit. Since GA is probabilistic, it is imperative to use a pre-defined limit, which is independent of the node chosen. To generate the initial population of chromosomes, this process was repeated until the predefined population size was achieved. If the population size is too low, the computation
time would be less. However the diversity in the solution set would be low, which might lead to convergence to a local minima. On the other hand, if the population size is too high, the computation time would be high, but the diversity in the solution would be greater and there would be a greater probability of convergence to a global minimum.

Cost associated with each chromosome is computed and validity is examined as per the schedule. This is a cost minimisation problem; hence the fitness function is inversely related to cost function \((f)\). If a solution is invalid, a high penalty is imposed, which makes the cost very high, thereby rendering the solution infeasible. The cost function considered here is the sum of travel time and the waiting time, or in other words, the total duration of the trip.

After initialisation, the next step is the Mating step. In this step, genetic operators such as crossover, selection and mutation are applied on the successive generations of the population, till the termination criterion is satisfied. The genetic operators - Selection, Crossover and Mutation, have been used in the study. A selection mechanism in GAs is simply a process that favors the selection of better individuals in the population for the mating pool. The present study uses elitism, which ensures that better solutions have greater chance of selection as compared to poorer solutions. The crossover operator looks for common genes in two parent chromosomes and then swaps parts of these to produce two new offspring chromosomes. A crossover probability \((P_c)\) decides whether crossover takes place in a particular iteration. The first step during the implementation of the crossover mechanism is to choose two chromosomes. During crossover mechanism, two parent chromosomes are chosen randomly. Potential sites (nodes) for crossover are identified in the parent chromosome. If multiple such sites are identified, one of them is chosen randomly. Following this, crossover takes place, and two new offspring chromosomes are generated. A post-processing procedure is employed after this, which removes loops and repetitions. The cost associated with each solution obtained in the crossover step is computed and infeasible solutions are rejected. Mutation represents a random order change that ensures that the solution does not converge to a local minima. In the present study, order change was carried out for odd genes (representing bus stops, except for the first and last gene). The probability, \(P_m\) was used to denote the probability of mutation. A repair function was used in case the resulting chromosome was infeasible.

**IMPLEMENTATION ON A STATIC NETWORK**

The GA based framework was implemented using Python language. Parameters for the planner were based on user preferences obtained through survey conducted in Chennai city. Static networks contain pre-determined and invariant temporal information at the time of computation of solution. For the purpose of implementation, GPS data were obtained from buses operated by the Metropolitan Transport Corporation (MTC), Chennai. Each record (or log) containing the data included a timestamp and the corresponding longitude and latitude of the location of the bus.

The constraints of the problem were broadly divided into two categories: hard constraints and soft constraints. Hard constraints were used during computation of results and the soft constraints were used after the computation of results to filter the solution set obtained, in order to present user-relevant solutions. The hard constraints were of two types. The first type was the network constraint such as buses that travelled on certain fixed routes, and that could not travel on any physical link (road) at free will. The buses were also constrained by the schedule and trip route. To make it computationally easy to incorporate this condition, the bus network was represented in the form of a directed graph, with links corresponding to a particular route ID, and the to and fro directions within the same route were assigned different route IDs. The second
type of hard constraint was the maximum number of transfers in a particular solution. This limit
not only helped in eliminating a large number of solutions but also, reduced the computation
time of the algorithm. The soft constraints included the limit on transfers, maximum walking
distance and the maximum travel time, as per the user’s preferences. These limits helped in
presenting user-specific solutions.

The selection of GA parameters, as discussed in the earlier section, is usually decided
considering the tradeoff between accuracy and computational efficiency. In GA domain, these
parameters are heuristically decided for the specific problem under consideration and the same
was done in this study. A low probability value of mutation was considered since mutation
results in breaking up of already fit solutions to generate new solutions ($P_m$ value of 0.1). Since
the new solutions generated need to undergo additive repair and the network chosen in the study
was “sparse”, this often led to increased length, implying that solutions with multiple transfers,
often exceeding the global limit, were generated. Hence, the crossover operator was given
preference and a comparatively higher probability value was chosen in the study ($P_c = 0.8$).
After the operator probabilities were decided, the remaining parameter was the population size.
To find the optimal value for this, the computational time and percentage of iterations producing
optimal solution were taken as the main criteria and a trial and error procedure was followed.
Thus, the final GA parameters chosen were as follows: for the initialisation step, a chromosome
length limit of 7 and a predefined population limit of 10 were chosen. For the mating step, a
chromosome length limit of 7, Probability of crossover ($P_c$) value of 0.8, Probability of mutation
($P_m$) value of 0.1 and Total number of generations value of 21 were chosen.

All the network and schedule data that was used for the algorithm were stored in a
temporal-geospatial database called the Service and Time Table (STT) database. This had two
sub-components. First was the network structure data, which was stored in the form of an
adjacency matrix and it contained the information of all nodes that were adjacent to a given node,
with corresponding arcs representing various bus routes. Second component of STT was the
temporal data for the routes. The GPS data obtained from the buses and the corresponding
predictions constituted the temporal data. Using this data, the departure times from each bus stop
were found out and this was stored in the STT as independent trip files for each route.

Evaluation
Five routes were chosen with route IDs - 5A, 5B, 5E, 21D, 21L, 23C and 154. The algorithm was
tested for different origin-destination combinations and the performance was checked. Solution
for a sample query for origin as “Broadway”, destination as “ESI Hospital” and time of travel as
10:30 am is shown in Figure 1.
Solution #1
Bus Stop >>> Broadway
Take Bus >>> 21L
Bus Stop >>> Madhya Kailash/CLRI
Take Bus >>> 5E
Bus Stop >>> ESI Hospital
Cost Function Value is 4500

Solution #2
Bus Stop >>> Broadway
Take Bus >>> 21L
Bus Stop >>> Santhome Church
Take Bus >>> 21D
Bus Stop >>> Vannandurai
Take Bus >>> 5E
Bus Stop >>> ESI Hospital
Cost Function Value is 5891

Solution #3
Bus Stop >>> Broadway
Take Bus >>> 21D
Bus Stop >>> Ashtalakshmi Koil/Velankanni Koil/Besant Nagar Church
Take Bus >>> 5E
Bus Stop >>> ESI Hospital
Cost Function Value is 5940

Solution #4
Bus Stop >>> Broadway
Take Bus >>> 21D
Bus Stop >>> Vannandurai
Take Bus >>> 5E
Bus Stop >>> ESI Hospital
Cost Function Value is 5940

Time for the solution was 0.27 seconds

FIGURE 1 Program output for sample query – static case.
Computation Time Measurement
To measure the computing time required to produce solutions for a general query, an origin node was chosen at random, and then solutions were computed for all destination nodes. Time taken for each such iteration was noted. It was observed that iterations that failed to yield a solution within a time limit of 0.6 second were those that had no feasible solution (for the given origin-destination pair). Hence, for the network chosen, 0.6 seconds was taken as the cutoff time limit. Sample result for a randomly selected origin node (Broadway) and time (10:30 am) was calculated.

A total of 168 valid iterations were produced, which implies that a total of 168 destination nodes could be reached from the selected origin node, that is “Broadway”. The average time taken to compute each solution was found to be 0.27 seconds. The standard deviation was calculated and was found to be 0.08 seconds. Since the network chosen was sparse, the number of iterations were fixed, and this served as the termination criterion.

REAL-TIME INTEGRATION
Traffic networks are dynamic in nature and hence the trip-planner needs to work with real-time data or predicted data. In order to achieve this, the algorithm has to be integrated functionally with real-time data input (predictions). Real-time data were available for a subset of buses in sixteen routes namely - 19B, 154, 21L, 221H, 23C, 47A, G18, M119A, M14, 5A, 5B, 5E, 7B, M7A, M7 and M70. A prediction algorithm was used, which received raw data from buses and generated the Estimated Time of Arrivals (ETAs) at various bus stops that were part of its route. The prediction algorithm generated a separate file referred to as the ETA file, for each bus on a particular route and for a specific bus stop. These predictions were made at small time intervals until the specific bus stop was reached, at which point the prediction process stopped. Although the prime focus of this study was to create a trip-planner for bus routes, walking and a portion of Chennai Mass Rapid Transit System (MRTS) was also included in the geospatial database to extend the trip-planner’s capabilities. The MRTS section between “Velachery” and “Chennai Beach” (to and fro) was included as a separate mode. For walk mode, a simple distance calculation based on GPS coordinates was proposed. A connector was defined as a link between a bus stop and the nearest MRTS station which would be traversed by walk. A fixed time was also added towards "ticket purchase" at the MRTS station.

Evaluation
A sample query is discussed here. A hypothetical user made a query at 11:48 am for a trip required at 12:15 pm. The origin was Karapakkam (bus stop) and the destination was Chennai Beach Station (MRTS station). The route suggested to the user is shown in Figure 2.
Let us assume, the user chose this solution and made multiple queries at different times before and during the journey, each representing a scenario, as follows.

- Scenario 1: Before starting the trip, roughly 30 minutes before journey.
- Scenario 2: Before reaching the origin bus stop “Karapakkam”, ten minutes before journey start time (as estimated in Scenario 1).
- Scenario 3: During journey between origin and terminal bus stop.
- Scenario 4: Ten minutes before estimated time of reaching the terminal bus stop. In this case, “Madhya Kailash” bus stop.
- The actual time of arrival is represented as scenario 5. This information is the real trip information available only after the entire trip is completed.

Let the cost function chosen in the study was total travel time, and a high variation in the value of this function can happen enroute due to a missed transfer. Or a better alternate solution could come up after the journey had begun. This would mean that the solution could change mid-journey. This may be unfavourable for a class of users who would want a “conservative” solution that would not drastically change during mid trip, a situation that could lead to unforeseen transfers and lack of an accurate estimate of the total journey time. These fluctuations are especially serious for Scenario 3 and 4, since these queries are made after the journey has begun. For example, Table 1 lists the ETAs at different times of the sample query and the corresponding cost function values. Figure 3 shows the corresponding scenario plot with cost function value, where the start of the journey is represented by a vertical line at Scenario 3. It could be observed that there is significant variation in the cost function value beyond Scenario 3, which implies that the user would get fluctuating values for the estimated travel time.
TABLE 1 ETAs at different times of query and corresponding cost

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Time of query</th>
<th>Bus stops</th>
<th>Walk &amp; ticket time</th>
<th>Boarding: MRTS</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Karapakkam</td>
<td>Madhya Kailash</td>
<td>Kasturbai Nagar</td>
<td>Chennai Beach Station</td>
</tr>
<tr>
<td>3</td>
<td>12:30</td>
<td>N/A</td>
<td>12:42</td>
<td>12:51</td>
<td>13:11</td>
</tr>
<tr>
<td>4</td>
<td>12:32</td>
<td>N/A</td>
<td>12:40</td>
<td>12:49</td>
<td>12:51</td>
</tr>
</tbody>
</table>

FIGURE 3 Cost Function value against scenario.

Since, this fluctuation in cost function value may cause discomfort to certain commuters, they may prefer a conservative solution. These fluctuations happen mainly due to uncertainty in predicted values (ETAs), and they should be captured within the trip-planner to provide buffers around the predicted time and ensure that the solution does not change drastically, if the user wanted “conservative” solutions. However, there may be users who would like to know the updates and make changes accordingly. The present solution tries to address the requirements of both these group of users: those who wants to stick to a conservative solution and those who want live updates, as follows.

Addressing the errors

When a user queries, the algorithm uses predicted data, which is available at that instant. This predicted data may contain prediction error, and this needs to be accounted for within the trip-planner to generate “conservative” solutions. Broadly speaking, these errors are dependent on the route, the bus stop and the time of the day. To explore this error, data corresponding to bus on route “19B” and bus stop “Madhya Kailash” were selected as an example. The errors were categorised into different groups, depending on the time of the day and the time to reach the bus stop. The mean error values were calculated separately for each category. Separate values were obtained for positive and negative errors. A positive error occurs when the bus reaches earlier than predicted and a negative error occurs when the bus reaches later than predicted.
sample data is presented in Table 2. It can be seen that reasonable errors exist when the prediction horizon is more than 30 minutes. Thus, if a user queries more than half an hour before the time of travel, the solution given may have errors up to 5 minutes. These should be absorbed by the solution in order to prevent the solution from changing drastically.

**TABLE 2 Mean error values (all values in seconds, magnitude only)**

<table>
<thead>
<tr>
<th>Time to reach bus stop</th>
<th>Period 1</th>
<th>Period 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean positive error</td>
<td>Mean negative error</td>
</tr>
<tr>
<td>&gt; 1800</td>
<td>295.15</td>
<td>135.33</td>
</tr>
<tr>
<td>1200 - 1800</td>
<td>114.47</td>
<td>67.48</td>
</tr>
<tr>
<td>600 - 1200</td>
<td>85.41</td>
<td>43.8</td>
</tr>
<tr>
<td>300 - 600</td>
<td>67.75</td>
<td>43.8</td>
</tr>
<tr>
<td>&lt; 300</td>
<td>36.67</td>
<td>5</td>
</tr>
</tbody>
</table>

To incorporate the above errors, an error function was constructed, which took into account the time of day, the bus stop ID concerned, the route ID and the time of query. Let $t_{ai}$ be the ETA at time $t_i$. Let $t_a$ be the actual/observed time of arrival at the bus stop. To buffer the expected time of arrival, error limits were imposed as shown in equation (1).

$$t_{ai} - \varepsilon_+ \leq t_a \leq t_{ai} + \varepsilon_- \quad (1)$$

Here $\varepsilon_+$ represents the magnitude of mean positive error and $\varepsilon_-$ represents the magnitude of the mean negative error. It is difficult to choose the error value since $t_a$ is unknown at the time of prediction. Hence, the error values of $(t_{ai} - t_i)$ from pre-computed tables such as Table 2 are used. To ensure that this solution remains valid with a high probability, bounds are placed on the predicted data to consider the worst-case scenario. This also ensures that the solution or the cost function value does not fluctuate, so that the user gets an upper limit value of total time of the journey. In other words, this leads to the generation of conservative solutions.

The sample real-time query presented earlier is explored again, with lower and upper bounds placed on the ETAs with the help of error function. Table 3 provides the bounds on the ETAs. The scenario notation remains the same as earlier. Plot of the cost function value against scenario is shown in Figure 4. It can be observed that the cost function value does not fluctuate at all after the trip begins due to inclusions of bounds. Hence, the error function proves to be a useful feature to reduce fluctuation in the solutions (and corresponding cost value).

However, if the user decides to obtain a dynamically optimised solution, then an alternate mode of solution generation is adopted. This feature termed as “LiveUpdate Mode” is offered as a choice at the time of query. Under this feature, the user is initially provided with an optimised solution based on data available at the time of initial query. This solution is then updated and may be modified depending on conditions, to provide the user alternate travel routes that are optimal. The term "stop-wise" is used since the solution is updated at each bus stop in the transit path of the initial solution so that the user is able to make a transfer in case the alternate optimal path is preferred over the current path. Moreover, the latter option is especially useful for the user in case the original solution suggested by the trip planner is void due to delays, congestion, etc. This could often lead to missed transfers, and thus necessitates an alternate solution to be presented to the user.
TABLE 3 Bounds on ETAs: Real-time sample query (UB – Upper Bound, LB – Lower Bound)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Time of Query</th>
<th>Bus Stops</th>
<th>Walk &amp; Ticket Time</th>
<th>Boarding: MRTS</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Karampakkam</td>
<td>Madhya Kailash</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LB</td>
<td>UB</td>
<td>LB</td>
<td>UB</td>
</tr>
<tr>
<td>3</td>
<td>12:30</td>
<td>N/A</td>
<td>N/A</td>
<td>12:38</td>
<td>12:44</td>
</tr>
<tr>
<td>4</td>
<td>12:32</td>
<td>N/A</td>
<td>N/A</td>
<td>12:36</td>
<td>12:43</td>
</tr>
</tbody>
</table>

FIGURE 4 Cost function value against scenario with error function.

Through experimentation, it was found that a higher population size gives better diversity that helps to prevent early convergence to a local minima and increases the chance of finding better solutions, but this also increases the computation time. Hence, a modification in algorithm was carried out to achieve the dual objective of diverse population set and less computation time.

The algorithm was segmented so that the search space could be narrowed down during a real-time query. Hence, partial network solution was generated beforehand and stored as solution files. The solutions stored in these files could be retrieved and temporally validated during a real-time query. In this manner, the initialisation step could produce a diverse population since it would get enough time for computation.

In the segmented algorithm, seed individuals (solutions from a pre-calculated solution pool) were injected into the population during initialisation. Two different seeding mechanisms were identified and tested on the bus network data available. In the first mechanism, the usual procedure for initialisation was carried out; hence the initial population was created using solutions built by random selection of genes and appending, till a predefined population limit was reached. In the second mechanism, a modified initialisation technique was suggested which involved crossover and selection operation on the set of solutions obtained during initialisation.

In the first mechanism, initialisation was performed, with a chromosome limit of 10. And
in the second mechanism, initialisation with a chromosome limit of 30 followed by random crossover step, with equal fitness assigned to each chromosome was carried out. The probability of crossover was 0.8 ($P_c = 0.8$). After the solutions to the network problem were generated and stored, these could be retrieved whenever a query was made to seed the GA. After retrieving the partial network solutions, these would be assigned a cost value based on the real-time data available at the time of query. Following this, the usual mating step is carried out. The two seeding mechanisms mentioned above were tested and the number of nodes reached in the second scenario was considerably more compared to that in the first scenario (20.41% approximately). Hence, due to the advantage offered, the seeding mechanism proposed in the second scenario was chosen.

**Computing Time in Segmented Algorithm**

The time required to produce solutions for a query in case of segmented algorithm was obtained for multiple iterations, to find out the variation in time required to produce valid solutions from a solution file. An origin node “Saidapet” and time of travel as 10:30 am was selected. A total of 24 valid iterations were produced, which implies that a total of 24 destination nodes could be reached from the selected origin node, that is, “Saidapet”. The average time taken to compute each solution was found to be 0.02 seconds and the standard deviation was calculated and found to be 0.01 seconds. Thus, segmentation produced results faster than the basic algorithm.

**APPLICATION DEVELOPMENT**

The above findings have helped in determining the essential requirements of a trip-planner that operates at both the static and real-time levels. The trip-planner was implemented for a set of routes within the MTC using Genetic Algorithm (GA) based framework. Once satisfactory results were obtained for the static networks, the algorithm was implemented with real-time data. A survey was conducted to understand the needs of a typical bus commuter in the city of Chennai and based on those suitable modes of information dissemination were developed. A dedicated website aimed at static route planning and a mobile application aimed at real-time route planning, were developed. The mobile application gives the user to choose real time updates, if interested. Such a real-time strip-planner would enable a user to access the information on the go and the user can opt for a route that matches their requirements, both “conservative” as well as a dynamically updated solution.

Once such a solution is open to the public, the problem of handling multiple queries in real time can be a major practical issue and is discussed in the section below.
Handling Multiple Queries in Real-Time

If multiple requests were sent to the trip-planner simultaneously, it would lead to delays. A way of optimising computing time in case of multiple simultaneous queries is to maintain a cache, which is like a container that would store recent queries. Results of previous queries were stored in Temporary Solution Files (TSF) and were assigned a Validity parameter that would denote the time after which the solution file would no longer be valid. Each time a fresh query was made, the TSF would be checked. This cache would help when there are multiple similar (same origin and destination) queries in the queue. To evaluate the performance of this, three queries were sent to the trip-planner, with similar first and last query, and the intermediate query for a different origin and destination. It was observed that, due to the cache, solution to the repeated query (query 3) was generated 56% faster compared to the scenario without a cache.

SUMMARY AND CONCLUSION

The present study was an attempt towards developing a trip-planner operating at both the static and real-time levels for the Metropolitan Transport Corporation (MTC) bus service. The MTC bus service often does not conform to the timetable due to a variety of reasons, like the uncertainty associated with Indian traffic due to heterogeneity and lack of lane discipline, delays caused due to traffic jams, etc. These factors often lead to delays and hence commuters have to suffer, as a consequence of which they shift to private modes of transport.

An accurate trip-planner can address this issue to a great extent and is the motivation for the present study. The study developed a trip-planner using Genetic Algorithm (GA) based framework. The trip-planner was initially implemented for a set of routes within the MTC, considering it as static. Once satisfactory results were obtained for the static network, the algorithm was implemented with real-time data. It was noted that the prediction values contained an inherent error that could sometimes lead to breakdown of solution suggested to the user, during the journey. To resolve this issue, an error function was created. Further, inclusion of MRTS data showed that this framework could be integrated with other modes of public transport.

In order to speed up the GA based framework, the algorithm was segmented and a temporal cache was introduced. Overall, the results showed that there is great scope for scalability and citywide implementation of the real-time route planner developed in this study. The only requirements would be GPS data obtained from the buses and any other mode of transportation that is included. The algorithm is is optimised to be computationally efficient and can be easily deployed as long as network information is available for a locality. The solution can be easily used for other public modes of travel, thereby expanding the network information available to a commuter. Web based and mobile based applications were developed for disseminating this solution based on the requirements of the users identified based on surveys.

As a future extension, the cost function could be improved and the weights could be altered for travel and waiting time based on field experiments and user surveys. Instead of a linear equation based cost function, a criterion based on pareto-optimality could also be looked into. Introduction of a recommender system that would suggest similar solutions to similar group of people (similarity based on factors such as economic background, gender, age group, etc.), which may accommodate each group's needs and preferences can also be attempted.

ACKNOWLEDGEMENT

The authors acknowledge the support for this study as a part of the sub-project CIE/10-11/168/IITM/LELI under the Centre of Excellence in Urban Transport project funded by the Ministry of Urban Development, Government of India, through letter No.
REFERENCES


