TEMPLE TOUR SG

Cohort 3 Team 10 – Yap Wei Lok, Kok Hanyi, Doan Thanh

Implementation

The tour app uses a simple *ImageView* with overlapping *Button* for its landing activity. When the button is clicked, it goes to the planning activity. The planning activity contains a lot of features. Firstly, the activity uses a custom *ListView* that covers 80% of the screen, while the other 20% is taken up by two *Buttons*, "Add Attraction" and "Done". The custom *ListView*, named *PlanListView* contains an *ImageView*, two *TextViews* and a small *ImageButton* for delete. It uses an adapter to manage its functions. Furthermore, the *ListView* items can be clicked and will direct the user to the item's info. All of the info is stored in a *SQLiteDatabase*. The activity also has a *MenuInflater* that allows the user to add all or remove all attractions from the selected list. When the user has selected some attractions and the done button is pressed, an *AlertDialog* will pop up to prompt the user to enter his/her budget and also a *CheckBox* to enable brute force algorithm in the next activity, which contains a *MapFragment* that shows the user the route to take.

Route Finding

When the user has entered his/her budget and choose to proceed, the app will calculate the optimal route to take, minimizing time taken to travel while keeping within budget. For the brute force algorithm, we generated all possible routes and calculated their total time if all trips were by taxi. After that, we started doing replacements for each individual path in all the routes, until the budget is met for each route. For the replacements, we will first pre-calculate whether taking the public transport or walking for that path is more cost-time efficient and save that as the alternative. Then we will get the time increase per cost decrease (TIPCD) for choosing public transport or walking as compared to the taxi. This TIPCD is then used to determine which path in the route we should replace with alternative transport modes first, lowest to highest. This is because lower TIPCD implies that the alternative transport mode, ie. public transport or walking, has similar time-cost efficiency compared to taxi. We will replace in this order until the budget constraint is met. After doing all the replacements, we sort the list of routes according to the total time taken and return the one with the least. This ensures that we get the optimal route under the budget.

For the fast solver, we simply used the nearest neighbor heuristic to get an approximation of the optimal route. At the current attraction (or Marina Bay Sands in the case of origin), we get the next nearest attraction based on taxi time and add that path to the route. After getting the full route, we employ the same replacement strategy as above. But since the nearest neighbor heuristic only returns one route, we do the replacement for just one route instead of multiple as in the case of brute force. From our testing, the approximated optimal time is within 20% of optimality while saving a lot of time, average 0.016s as compared to 30s for brute force with 7 attractions. We did not try to compare computation times for more than 7 even though we have 10 attractions because it was too high for the brute force, making it hard to gather computation data for comparison.

While 20% seem like a lot, we believe that it is suitable for a day tour app because the route finding needs to be fast and yet reasonable. Even if we try to use better heuristics which are within 10-15% of optimality, the slight improvement in travel time approximation is not worth the increase in computation time. The app is for one-day trip planning, so 20% would not be as significant.

Map Directions

After calculating the route, the user is directed to a *Google Map Fragment* which supports Implicit Intent for more specific directions in the Google Maps app. An azure marker is placed on the origin, ie. Marina Bay Sands, and red markers on the attractions. Polylines are drawn from one place to another, with red for public transport or taxi and blue for walking. The public transport can be either bus or MRT. There's also a "Show Details" button for the user to view the specifics of each path like the time, cost and mode of transport, and the total time and cost. We also added a computation time indicator for the sake of this project.