Trip Planning using past data

29/06/2020

Tasneem Lightwala
183050004
CSE Mtech 2
Guided by: Prof. Abhiram Ranade

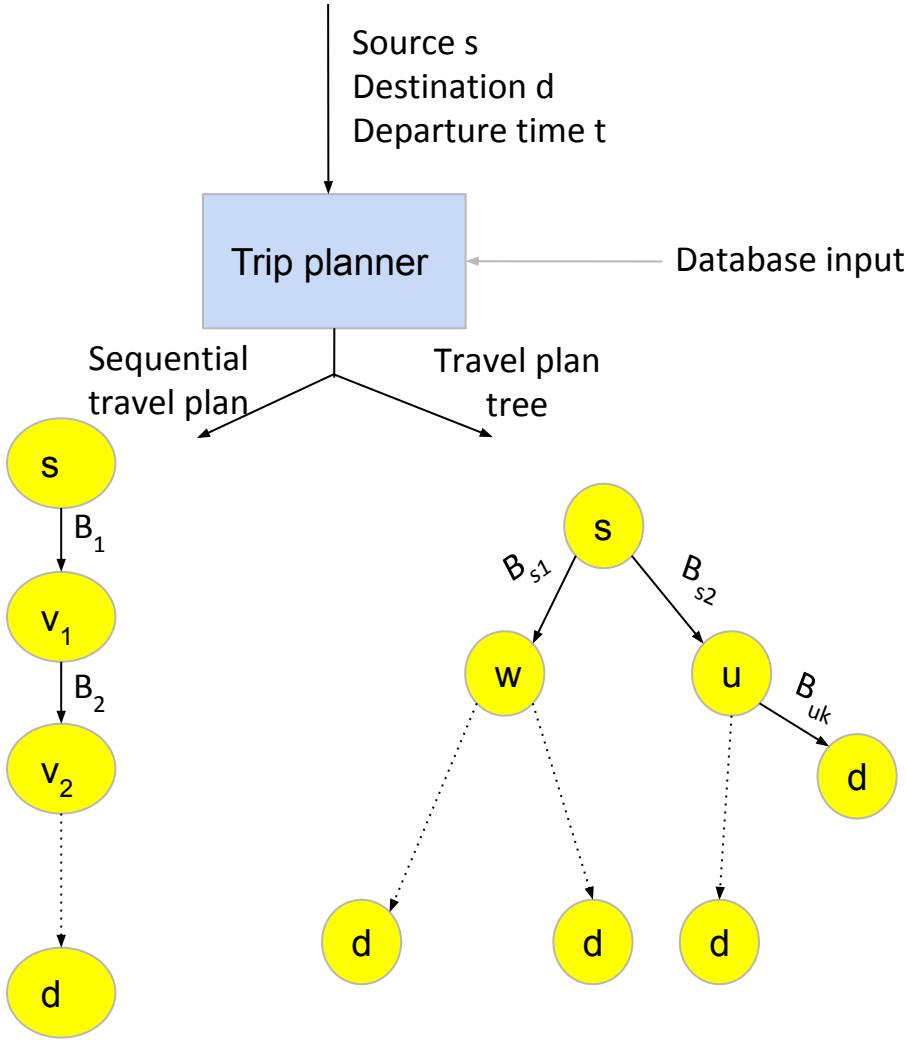
Outline

1. Introduction
    - Problem Statement
    - MTP focus
    - Trip data set
2. MTP Stage - 1
    - BEST ticketing data
    - Need for synthetic data generation
3. MTP Stage 2
    - Synthetic data generation
    - Modified Mumbai Navigator
    - Multi-label correcting (MLC) planner using past data
    - MLC planner using real time data
    - Best time estimator using RAPTOR algorithm
    - Comparison of Planners
        - Accuracy of expected total travel time
        - Ability to generate best plans
        - Plans to be picked in MLC
4. Conclusion & Future Work

Problem Statement

What is the optimal way to go from point A to point B in a city using its public transport service?

- Public transport network consists of bus stops and bus lines, where buses follow specific routes.
- Different optimizing criteria:
  - Travel time
  - Transfers
  - Travel cost
  - Reliability of plans
- Travel plans specify the estimated time to reach the destination.



MTP Focus

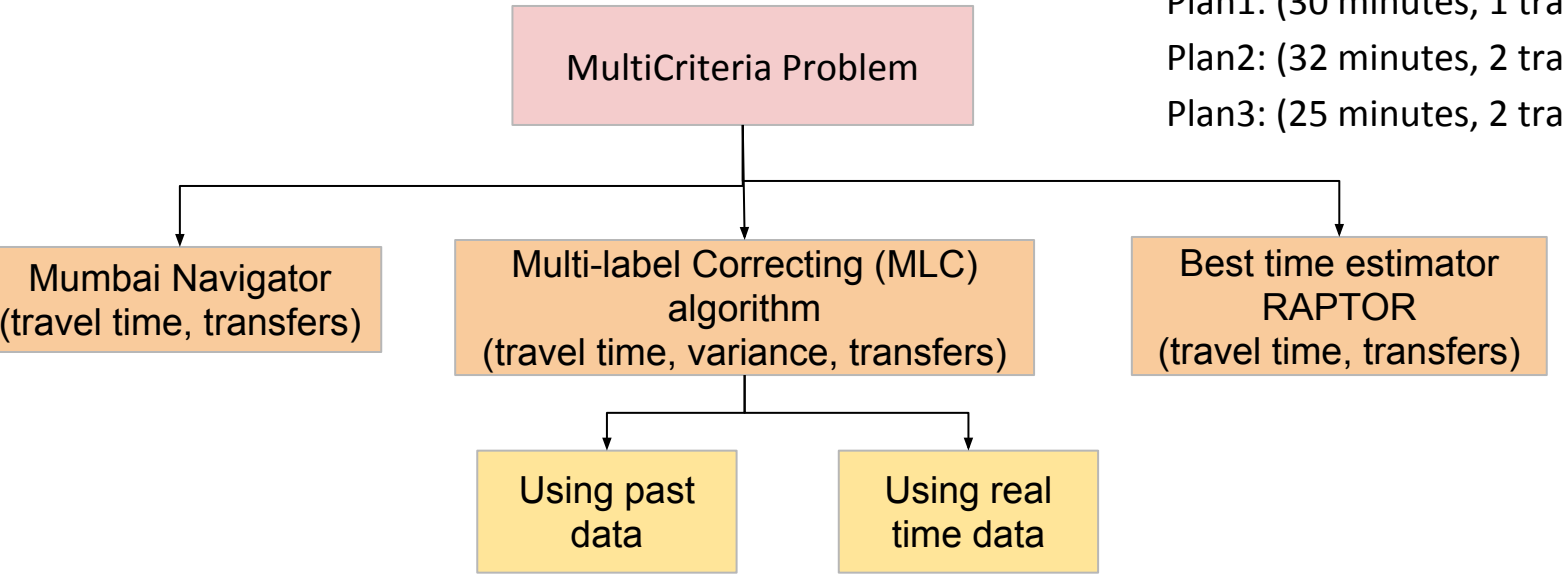
Non dominating set of plans:

Optimizing criteria (travel time, number of transfers)

Plan1: (30 minutes, 1 transfer) ✔

Plan2: (32 minutes, 2 transfers) ✘

Plan3: (25 minutes, 2 transfers) ✔



Database input: Network description, frequency of routes, travel time taken between stops

- BEST ticketing data : noisy and incomplete. Hence, we generated our own data
- Synthetic data

Comparison of these planners based on the accuracy of predicted time and its ability to generate best plans

Trip data

- Common database input provided to all planners
- Each planner then models it into its appropriate form
- Easier to answer queries

Trip ID	Stop number	Arrival time	Occupancy	Trip direction
4941_1/7/17_11	333	08:45	2	U
4941_1/7/17_11	254	08:52	6	U
. . .	. . .	. . .	. . .	
4941_2/7/17_13	412	09:54	23	D

Trip ID is a combination of route, date and trip number

MTP Stage 1

BEST tickets data

- 52 million ticket records of size 14 GB
- Each ticket record has 35 fields

single ticket cost	total cost	tickets bought	route no	bus no	ticket date	ticket time	“from” stage no	“from” stage name	“to” stage name	“to” stageno	trip direction	tripno
8	16	2	440	196	01/07/2017	07:00	710	SHRAWAN YESHWANT CHOWK	LOWER PAREL STATION	56	U	2

- 514 routes and 986 stage-stops identified in the data
- **Aim:** Convert this data to a much smaller “Trip data”. Leads to easy querying.
- **Conversion requirements:** Grouping tickets into trips, calculating occupancy and arrival time of trips at each stage
- **Result:** Trip data set of size 460 MB

Errors in BEST data

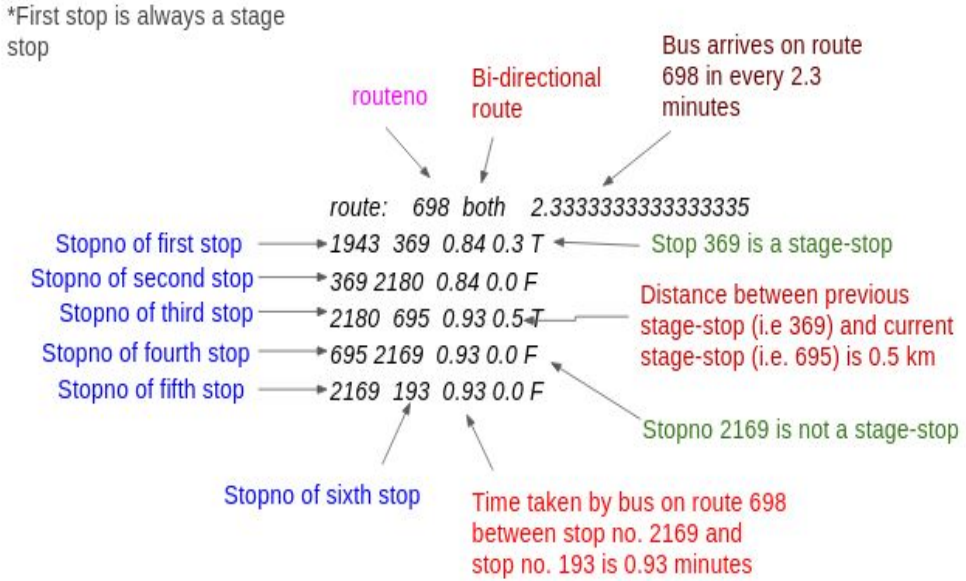
- Incomplete and noisy
- “tripno” field in ticketing data is not unique for a given route, date, vehicle
- wrong route no written with tickets
- Incorrect direction mentioned for single directional routes
- In 11% of the trips, either incorrect time was written with the ticket or ticket was issued much later (led to earlier arrival times for later stops)
- For some routes, no trips were made on a day
- For some routes, no trips were made for a long period of time on a day

Due to errors in BEST ticketing data, we decided to create our own data to be used with different planners

Synthetic Data Generation

Synthetic Data Generation

- Inconsistencies in BEST data, therefore constructing data synthetically
- Original Mumbai Navigator's "busdata" input file is used to get basic information like routes, stages, average frequency of routes, stage to stage distances and travel times.



Synthetic Data Generation

- For every route *r*
  - there is base frequency F_r → *taken from old MN data*
  - Frequency at time t on route r = $F_r + f_{tr,}$
  - f_{tr} is uniformly distributed between 0 and 3 minutes
  - Frequency function is called every time we want to start a new trip. To the last trip of the route, add the frequency as returned by the function to get the start time of the new trip on a route
- For every stage s
  - there is base speed SP_s of buses around it → *calculated using distances and travel time between stages in old MN*
  - Base speed is calculated by taking average speed of all routes passing through that stage stop
  - Speed of buses at time t around stage s = $SP_s + sp_{ts}$
  - Sp_{ts} is uniformly distributed between (-0.25*SP_s, 0) in peak hours and between (-0.01*SP_s, 0.1*SP_s) in non peak hours.
  - Speed function is called to predict the arrival time of trip at next stage stop

*Stages are taken instead of stops because ticketing data also had only stage information available

*Peak hours are assumed to be from 08:00 to 14:00 and from 16:00 to 21:00.

Past Data Generator

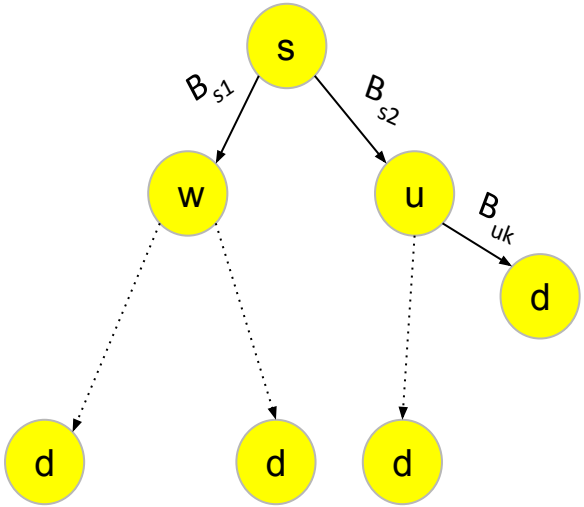
- generates trips for each route using appropriate frequencies and speeds corresponding to the time of the day
- starting time of the first trip of each route is taken as 6 a.m.
- Trip id is a combination of route, direction, date and trip number
- generated one week Trip data
- **Network Description file** stores routes, stages and stage to stage distances
- This Trip data and Network desc file is given as input to all planners

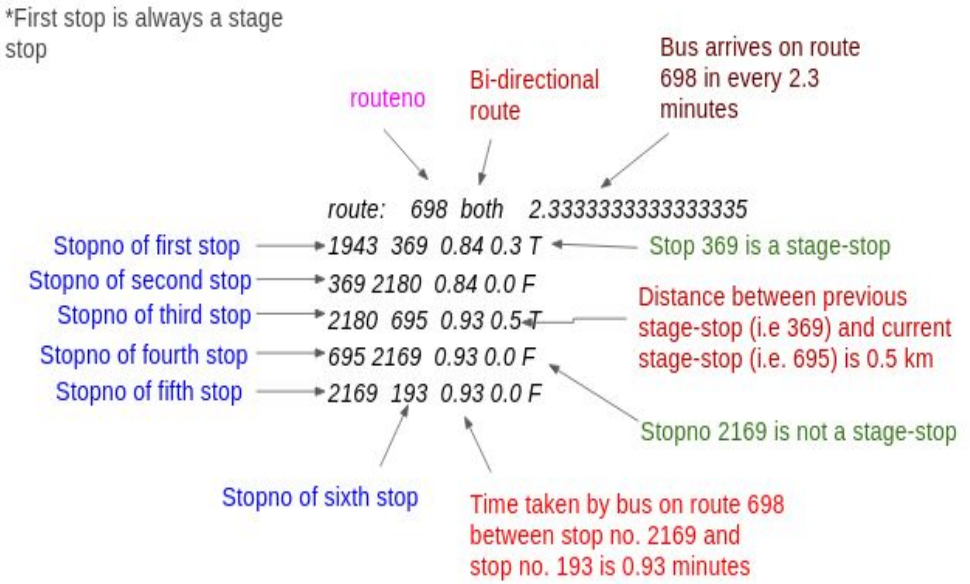
Trip ID	Stage no.	Arrival time
1_U_1_1	2220	06:00:00
1_U_1_1	43	06:02:57
. .	. .	. .
28_U_2_23	2650	09:37:41
. .	. .	. .

Modified Mumbai Navigator

Mumbai Navigator

- **User input:** source, destination
- **Database input:** “busdata” file (frequency of each route, stop to stop time along route)
- **Output:** Plan trees with 0, 1 and 2 changeovers with minimum expected time
- Optimizing criteria: travel time and transfers
- Independent of the departure time of the user





Busdata file

MN output

Software developed at IIT Bombay

Mumbai Navigator

Software developed at IIT Bombay

To travel from I.I.T.MAIN-GATE-POWAI to ELECTRIC-HOUSE-COLABA

No plan with 0 change overs

Plan with 1 change overs *(Expected Travelling Time: 137.59)*

1: I.I.T.MAIN-GATE-POWAI to AGARKAR-CHK.-ANDHERI-(E) by 496LTD,545LTD,403LTD.

2: AGARKAR-CHK.-ANDHERI-(E) to ELECTRIC-HOUSE-COLABA by 2LTD.

Plan with 2 change overs *(Expected Travelling Time: 83.56)*

1: I.I.T.MAIN-GATE-POWAI to DOCKYARD-CLY.-KANJURMARG-(W) by
44EXP,403LTD,459LTD,522LTD,523LTD,307,396LTD,398LTD,409LTD,422,424,425,428,460LTD,461LTD,469LTD,496LTD,602.

Walk from DOCKYARD-CLY.-KANJURMARG-(W) to KanjurMarg

2: KanjurMarg to MumbaiC.S.T. by C.S.T.Slow,

Walk from MumbaiC.S.T. to CHH.SHIVAJI-MAHARAJ-TERMINUS(BHATIA-FORT

3: CHH.SHIVAJI-MAHARAJ-TERMINUS(BHATIA-FORT to ELECTRIC-HOUSE-COLABA by 2LTD.

Modifications in Mumbai Navigator

- Using **different “busdata” files for different intervals of the day**: morning , afternoon, evening
  - Each busdata file stores frequency of the route and stage to stage along the route in that particular interval
  - We use synthetic Trip data to generate busdata files
  - Arrival time of trips at stages is used to segregate trips into different intervals
- Adding a **Simulator code**, which calculates the actual time taken by the plan
  - A random day from past is picked as travel day
  - **Input:** plan tree and trip data of the travel day (sorted on arrival time)
  - Follows the adaptive plan (taking buses that comes first at each stop)
  - **Output:** Actual time taken by the plan
- Now, the planner is although independent of the specific departure time, is dependent on the interval of the departure time.
- **Output of modified MN:** Generates plan trees using average whole day busdata and also specific interval busdata.

Modified MN output

Plans with 0 changeovers

Software developed at IIT Bombay

Mumbai Navigator

Software developed at IIT Bombay

To travel from SHAHU-NGR.-DHARAVI to BYCULLA-STN.(W)-BYCULLA-(W)

Using day data
Plan with 0 change overs *(Expected Travelling Time: 62.92)*

1: SHAHU-NGR.-DHARAVI to BYCULLA-STN.(W)-BYCULLA-(W) by 164_D.

Using morning data
Plan with 0 change overs *(Expected Travelling Time: 69.05)*

1: SHAHU-NGR.-DHARAVI to BYCULLA-STN.(W)-BYCULLA-(W) by 164_D.

Using afternoon data
Plan with 0 change overs *(Expected Travelling Time: 59.27)*

1: SHAHU-NGR.-DHARAVI to BYCULLA-STN.(W)-BYCULLA-(W) by 164_D.

Using evening data
Plan with 0 change overs *(Expected Travelling Time: 68.51)*

1: SHAHU-NGR.-DHARAVI to BYCULLA-STN.(W)-BYCULLA-(W) by 164_D.

Expected travelling time

Plans with 1 changeovers

Using day data

Plan with 1 change overs *(Expected Travelling Time: 54.21)*

1: SHAHU-NGR.-DHARAVI to LOKMANYA-TILAK-HOSPITAL-SION-(E) by 166_D.

2: LOKMANYA-TILAK-HOSPITAL-SION-(E) to BYCULLA-STN.(W)-BYCULLA-(W) by 63_D.

1: SHAHU-NGR.-DHARAVI to VEER-KOTWAL-UDN.-(PLAZA)-DADAR-(W)--D by 164_D.

2: VEER-KOTWAL-UDN.-(PLAZA)-DADAR-(W)--D to BYCULLA-STN.(W)-BYCULLA-(W) by 63_D.

Using morning data

Plan with 1 change overs *(Expected Travelling Time: 60.74)*

1: SHAHU-NGR.-DHARAVI to LOKMANYA-TILAK-HOSPITAL-SION-(E) by 166_D.

2: LOKMANYA-TILAK-HOSPITAL-SION-(E) to BYCULLA-STN.(W)-BYCULLA-(W) by 63_D.

1: SHAHU-NGR.-DHARAVI to VEER-KOTWAL-UDN.-(PLAZA)-DADAR-(W)--D by 164_D.

2: VEER-KOTWAL-UDN.-(PLAZA)-DADAR-(W)--D to BYCULLA-STN.(W)-BYCULLA-(W) by 63_D.

Using afternoon data

Plan with 1 change overs *(Expected Travelling Time: 53.18)*

1: SHAHU-NGR.-DHARAVI to LOKMANYA-TILAK-HOSPITAL-SION-(E) by 166_D.

2: LOKMANYA-TILAK-HOSPITAL-SION-(E) to BYCULLA-STN.(W)-BYCULLA-(W) by 63_D,164_D.

1: SHAHU-NGR.-DHARAVI to VEER-KOTWAL-UDN.-(PLAZA)-DADAR-(W)--D by 164_D.

2: VEER-KOTWAL-UDN.-(PLAZA)-DADAR-(W)--D to BYCULLA-STN.(W)-BYCULLA-(W) by 63_D.

Using evening data

Plan with 1 change overs *(Expected Travelling Time: 60.32)*

1: SHAHU-NGR.-DHARAVI to BHARATMATA-CINEMA/MAHADEO-PALAV-MAR-CURRY-ROAD by 166_D.

2: BHARATMATA-CINEMA/MAHADEO-PALAV-MAR-CURRY-ROAD to BYCULLA-STN.(W)-BYCULLA-(W) by 14_D.

1: SHAHU-NGR.-DHARAVI to VEER-KOTWAL-UDN.-(PLAZA)-DADAR-(W)--D by 164_D.

2: VEER-KOTWAL-UDN.-(PLAZA)-DADAR-(W)--D to BYCULLA-STN.(W)-BYCULLA-(W) by 63_D.

Plans with 2 changeovers

Using day data

Plan with 2 change overs *(Expected Travelling Time: 42.01)*

1: SHAHU-NGR.-DHARAVI to LOKMANYA-TILAK-HOSPITAL-SION-(E) by 166_D,164_D.

2: LOKMANYA-TILAK-HOSPITAL-SION-(E) to COM.GULABRAO-GANACHARYA-CHK.-ARTHUR-ROAD by 30LTD_D,66_D.

3: COM.GULABRAO-GANACHARYA-CHK.-ARTHUR-ROAD to BYCULLA-STN.(W)-BYCULLA-(W) by 63_D,2LTD_D,14_D.

Using morning data

Plan with 2 change overs *(Expected Travelling Time: 46.31)*

1: SHAHU-NGR.-DHARAVI to LOKMANYA-TILAK-HOSPITAL-SION-(E) by 164_D,166_D.

2: LOKMANYA-TILAK-HOSPITAL-SION-(E) to COM.GULABRAO-GANACHARYA-CHK.-ARTHUR-ROAD by 30LTD_D,66_D.

3: COM.GULABRAO-GANACHARYA-CHK.-ARTHUR-ROAD to BYCULLA-STN.(W)-BYCULLA-(W) by 63_D,2LTD_D,14_D.

Using afternoon data

Plan with 2 change overs *(Expected Travelling Time: 40.81)*

1: SHAHU-NGR.-DHARAVI to LOKMANYA-TILAK-HOSPITAL-SION-(E) by 164_D,166_D.

2: LOKMANYA-TILAK-HOSPITAL-SION-(E) to COM.GULABRAO-GANACHARYA-CHK.-ARTHUR-ROAD by 30LTD_D,66_D.

3: COM.GULABRAO-GANACHARYA-CHK.-ARTHUR-ROAD to BYCULLA-STN.(W)-BYCULLA-(W) by 2LTD_D,14_D,63_D.

Using evening data

Plan with 2 change overs *(Expected Travelling Time: 46.14)*

1: SHAHU-NGR.-DHARAVI to LOKMANYA-TILAK-HOSPITAL-SION-(E) by 166_D,164_D.

2: LOKMANYA-TILAK-HOSPITAL-SION-(E) to COM.GULABRAO-GANACHARYA-CHK.-ARTHUR-ROAD by 30LTD_D,66_D.

3: COM.GULABRAO-GANACHARYA-CHK.-ARTHUR-ROAD to BYCULLA-STN.(W)-BYCULLA-(W) by 2LTD_D,14_D,63_D.

Day plan vs Specific interval plans

Source stage no. = 2431
Destination stage no. = 432
Departure time = 15:00

	Using whole day database		Using afternoon database	
Plan ↓	Estimated time using whole day data (mins)	Actual time taken	Estimated time using afternoon data	Actual time taken
With 0 transfers	63	53	60	53
With 1 transfer	54	46	53	53
With 2 transfers	42	36	40	36

Average difference between estimated and actual time = 8 mins

Average difference between estimated and actual time = 4 mins

MLC planner using past data

MLC 1

MLC planner using past data (MLC1)

Multi-label correcting algorithm

User input: source, destination, departure time

Database input: Synthetic trip data, N/w desc file, interval duration

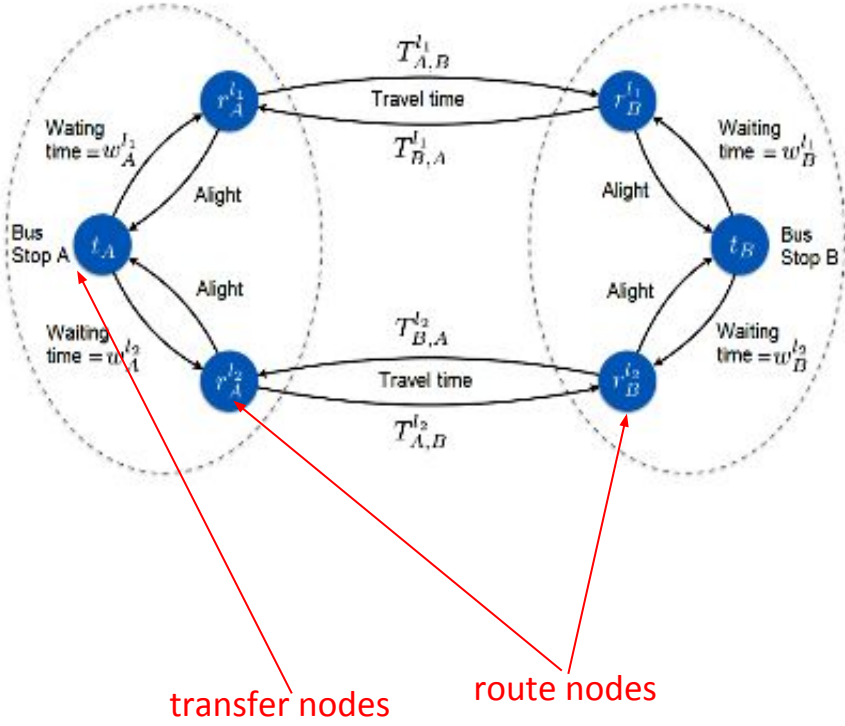
Planner output: Non dominated plans with up to 2 changeovers minimizing total time and variance

Optimizing criteria: travel time, variance, transfers

- Planner is time-dependent. Generates different plans for different departure times
- Uses only past data (obtained from synthetic Trip data) to generate plans
- Output of the planner is a set of non-dominated travel plans. Each travel plan has the estimated travel time to reach the destination and variance which tells by how can the travel time vary and also the number of transfers required in that plan

MLC transportation graph

- made using Network desc file
- Let Interval duration = 120 mins
- For each interval, we have calculated mean time and variance for each edge using past data
- For each interval j, calculate frequency of all routes r at each stage i
  - $f_i^{j,r}$ = M / 120
    - M = number of buses on route r arriving at stage i in interval j
  - Mean waiting time at stage i for route r in interval j
    - $w_i^{j,r} = 0.5/f_i^{j,r}$
- Similarly, for each interval j, for each consecutive stage pair (s_1,s_2) of route r, we calculate mean travel time $t_{s1,s2}^{j,r}$ and variance $v_{s1,s2}^{j,r}$ using past data
- Edge cost are selected based on the departure time and edge type

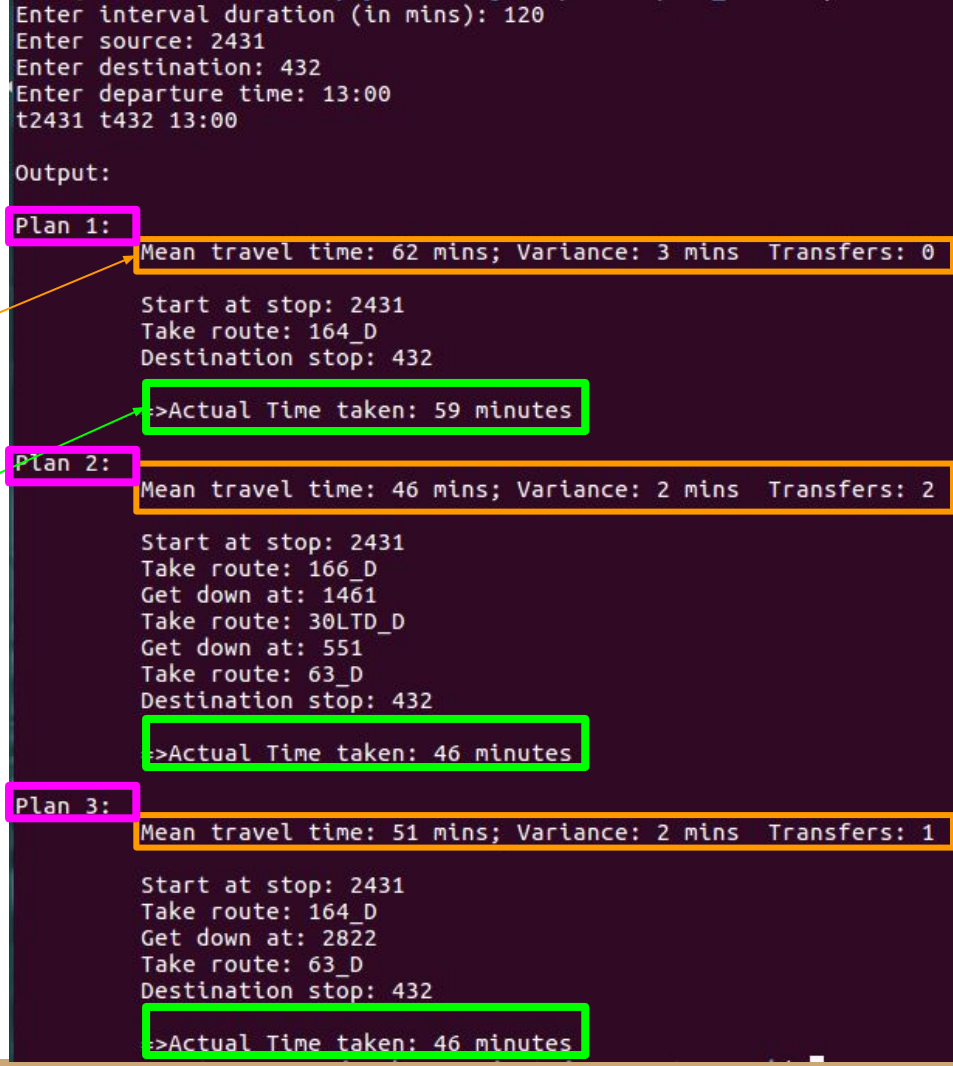


MLC1 output

source stage no. = 2431
destination stage no. = 432
Departure time = 13:00

- Planner generated 3 non-dominating plans.
- Each plan specifies estimated mean travel time, variance and number of transfers
- Complete plan with routes to take and stops to get down at is provided

To calculate the **actual time taken** in following the plan, we pick a random day from past.
Say, day 1 is picked for travelling. We provide complete Trip data of day 1, which contains exact arrival time of trips at stages. Generated plans are followed and Actual time taken by the plan is outputted.



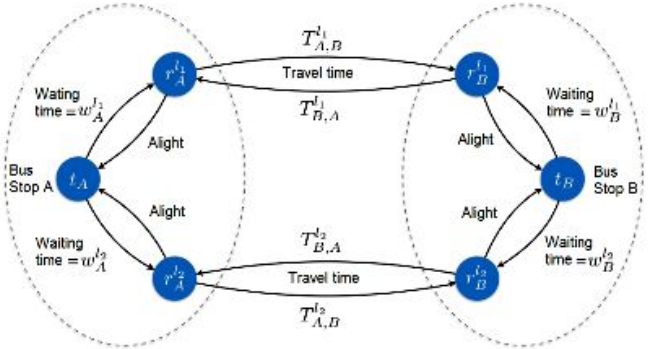
MLC planner using real time data

MLC 2

MLC planner using real time data (MLC2)

Multi-label correcting algorithm

- Uses same graph and algorithm as MLC planner with past data
- **Difference:** uses real time data of the travel day to calculate waiting time for buses at source
- To calculate real waiting time, Trip data of the travel day is provided to the planner up til the departure time.
  - Say travel day is day 1, and departure time is 11 a.m., planner gets to see the position of all buses on routes till 11 a.m. on day 1
  - Using these position of buses, real waiting time for routes at source is predicted.



User input: source, destination, departure time, travel day

Database input: Synthetic trip data (past data), Travel day data uptil departure time (real time data), N/w desc file, interval duration

Planner output: Non dominated plans with up to 2 changeovers minimizing total time and variance

MLC2 output

source stage no. = 2431 ,destination stage no. = 432

Departure time = 13:00

- Planner generated 3 non-dominating plans.
- Each plan specifies estimated mean travel time, variance and number of transfers
- Complete plan with routes to take and stops to get down at is provided

To calculate the **actual time taken**, we follow the plan on the day whose real time data was used. Say, day 1 is picked for travelling. We provide complete Trip data of day 1, which contains exact arrival time of trips at stages. Generated plans are followed and Actual time taken by the plan is outputted.

*estimated travel time generated by MLC 2 is closer to actual time taken as compared to estimated travel time generated by MLC1

```
Enter interval duration (in mins): 120
Enter source: 2431
Enter destination: 432
Enter departure time: 13:00
t2431 t432 13:00

Output:

Plan 1:
        Mean travel time: 59 mins; Variance: 4 mins  Transfers: 0

        Start at stop: 2431
        Take route: 164_D
        Destination stop: 432

        =>Actual Time taken: 59 minutes

Plan 2:
        Mean travel time: 44 mins; Variance: 2 mins  Transfers: 2

        Start at stop: 2431
        Take route: 164_D
        Get down at: 1461
        Take route: 30LTD_D
        Get down at: 551
        Take route: 63_D
        Destination stop: 432

        =>Actual Time taken: 46 minutes

Plan 3:
        Mean travel time: 48 mins; Variance: 3 mins  Transfers: 1

        Start at stop: 2431
        Take route: 164_D
        Get down at: 2822
        Take route: 63_D
        Destination stop: 432

        =>Actual Time taken: 46 minutes
```

Best time estimator

RAPTOR algorithm

Best time estimator

- finds best possible plan for a given source, destination, departure time and travel day
- Uses RAPTOR (Round-bAsed Public Transit Optimized Router) algorithm to generate best travel plans
- takes synthetic schedule of entire travel day, with actual arrival time of trips at stages on that day
- Optimizes travel time and number of transfers

  User input: source, destination, departure time, travel day
  Database input: Synthetic trip data of travel day, Network desc file
  Estimator output: best possible plans on that day with 0, 1 and 2 transfers

- Works roundwise, one for each transfer.
- In round k , it finds the fastest way to reach every stage from source with k trips (k-1 transfers)
- Takes the solution from previous round (k-1) and extend by one trip if it improves arrival time at stages.
- Each route is processed once in each round

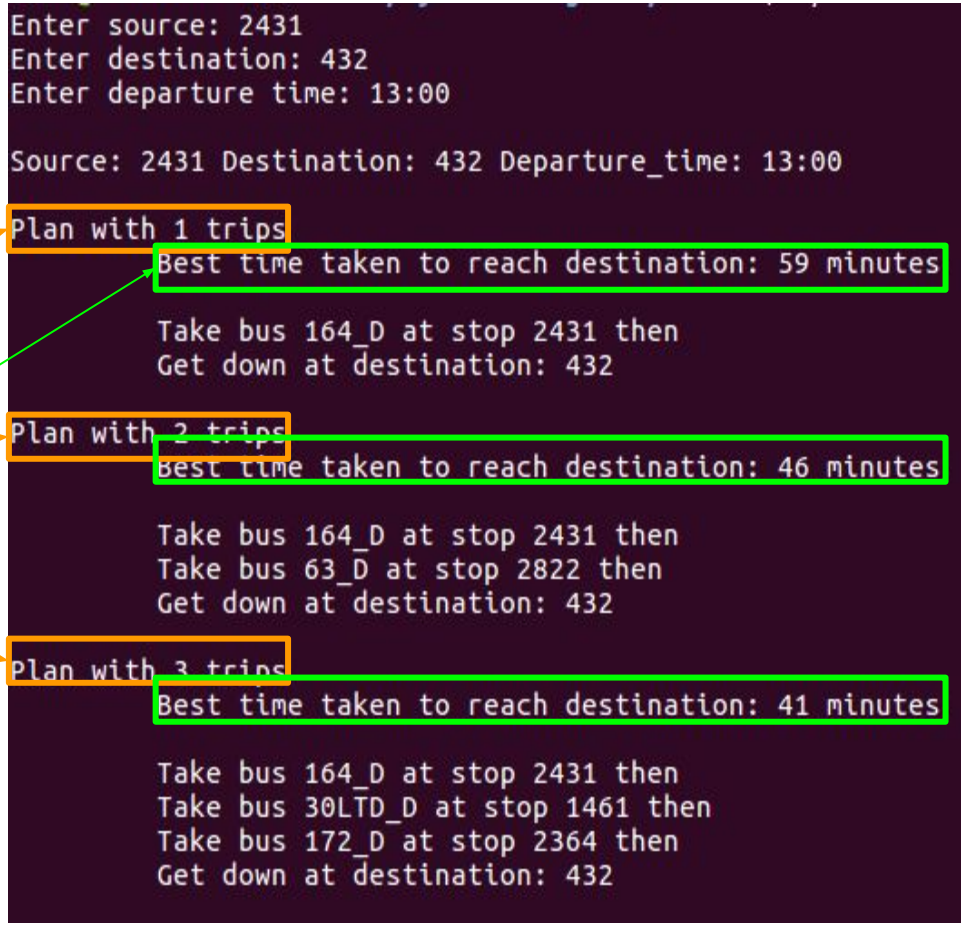
Best plans

source stage no. = 2431
destination stage no. = 432
Departure time = 13:00
K (max no. of transfers) = 2

Plan with 1 trip
Plan with 2 trips
Plan with 3 trips

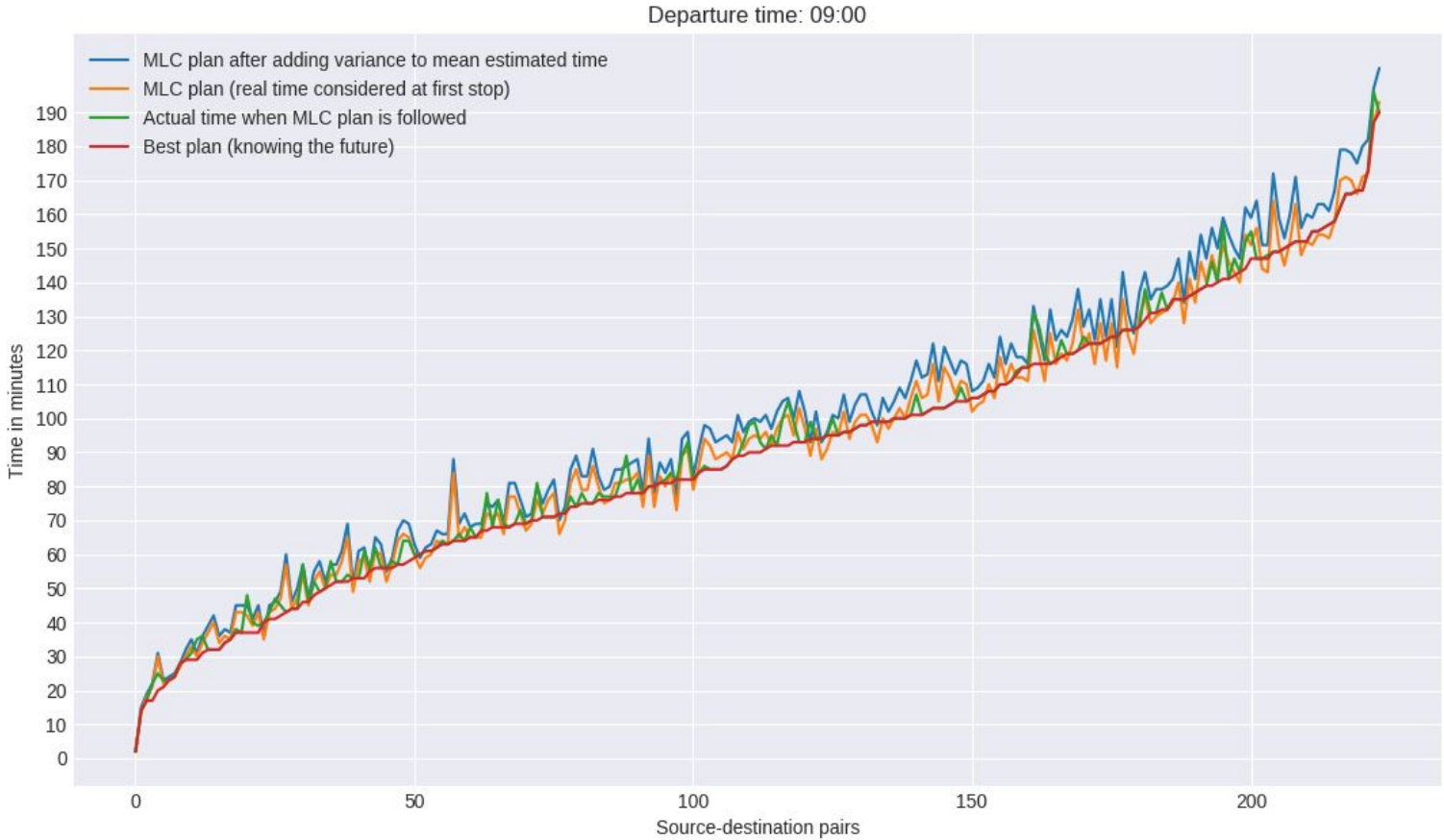
Plan i gives the **best time** taken to reach destination with i-1 transfers, along with the routes to take and stops to get down at.

We assume to know the future for the travel day to get the best plans



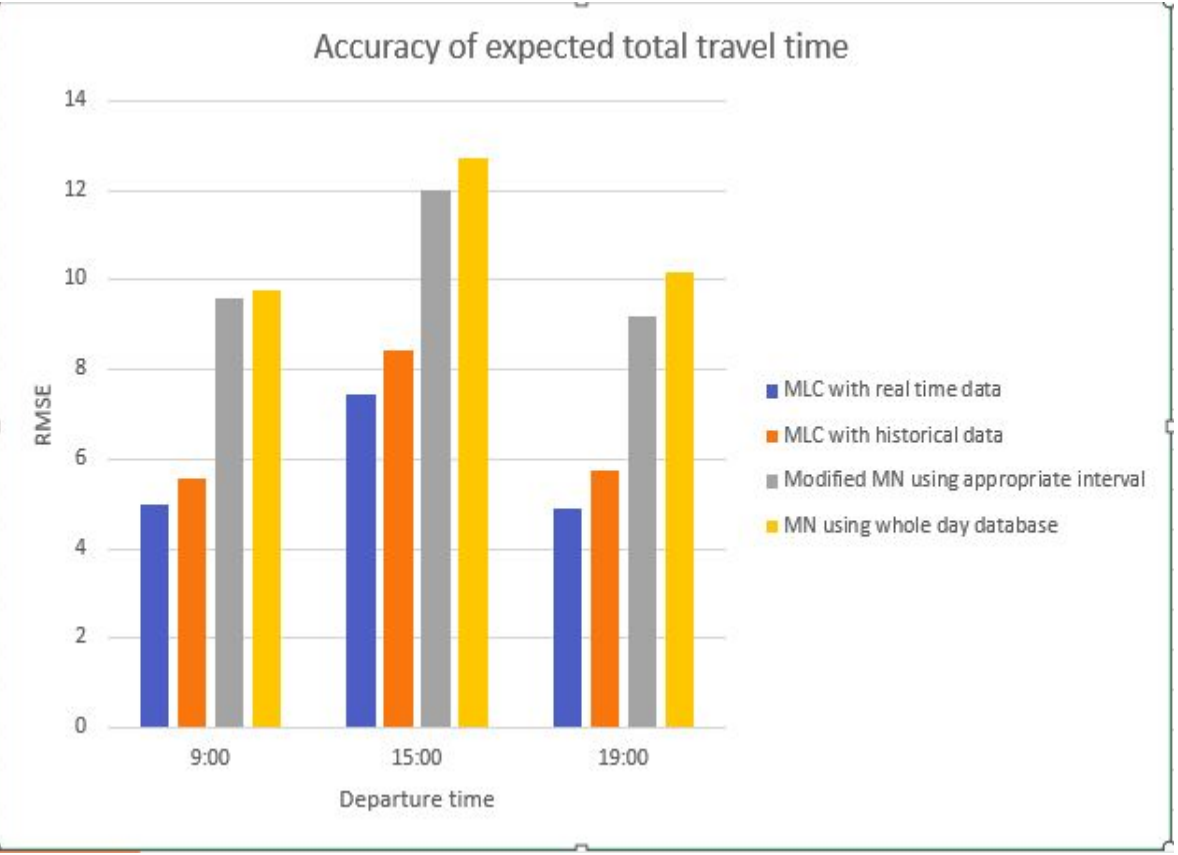
Comparison of Planners

Plans comparison with actual and best time



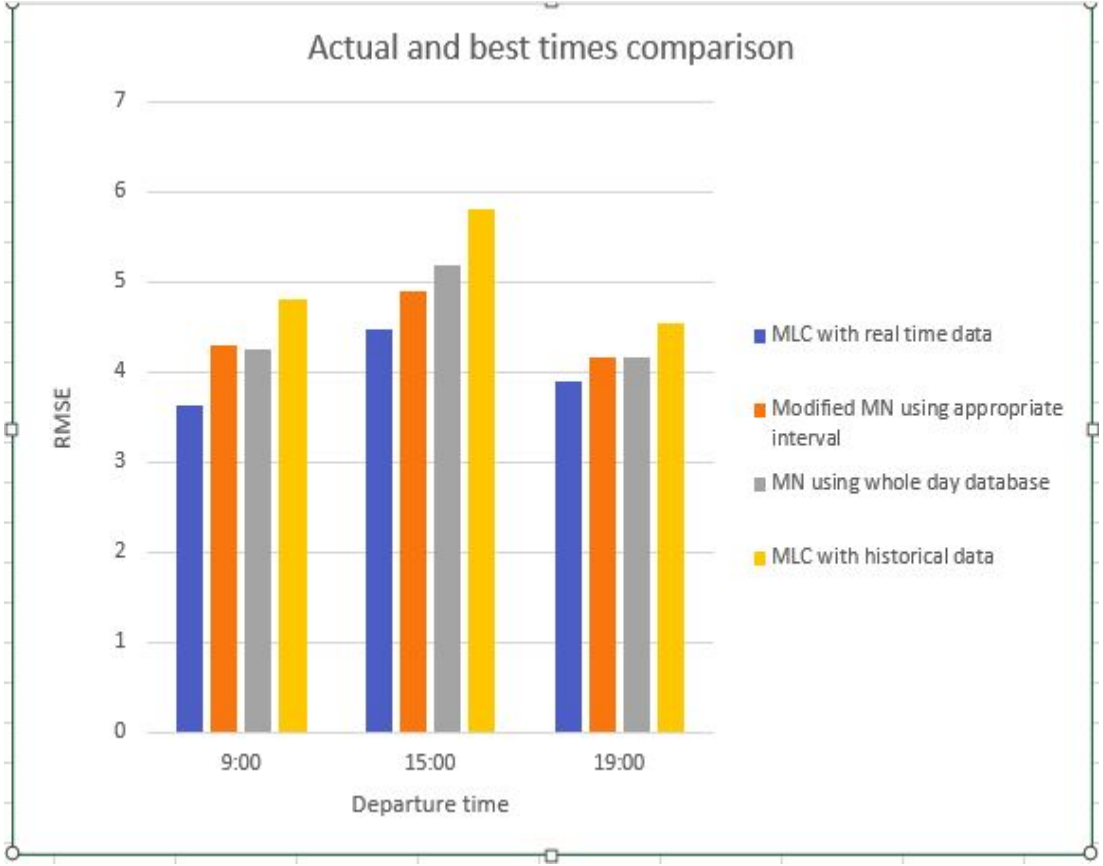
We see that the predicted time by MLC2 is slightly overestimated as compared to the actual time taken. The actual time plot and the best time plot overlaps for most of the part, suggesting that best plans are generated by MLC2 that uses real time data at first stop

Accuracy of expected total travel time



- Root mean squared error (RMSE) is used to measure the accuracy of all planners.
- We calculate the difference between the predicted time by the planners and actual time taken on travel day
- MLC with real time data is best in estimating the travel time
- Mumbai Navigator not considering departure time is worst in estimating the travel time

Are best plans generated?



- Compares difference between the actual time taken by plans and the best possible time
- Gives the ability of planners to generate best plans
- MLC with real time data gets the advantage of real time data
- Mumbai Navigator gets the advantage of adaptivity

Plans to pick in MLC

MLC 1 = MLC planner using only past data, generating multiple non-dominating plans

MLC 1a = MLC planner using only past data, generating minimum expected travel time plan

MLC 2 = MLC planner using real time data, generating multiple non-dominating plans

MLC 2a = MLC planner using real time data, generating minimum expected travel time plan

Mean absolute error

Results: Picking the plans with minimum expected total time amongst all non-dominating plans increases the error for almost all cases. Picking plans at random is better than picking minimum expected travel time plan.

RMSE vs MAE

Root mean squared error vs Mean absolute error

MLC 1 = MLC planner using only past data, generating multiple non-dominating plans

MLC 2 = MLC planner using real time data, generating multiple non-dominating plans

MN day = Mumbai Navigator independent of departure time

MN interval = Mumbai Navigator using interval of departure time

Planner -> Departure time↓	MLC 1 (RMSE | MAE)	MLC 2 (RMSE | MAE)	MN day (RMSE | MAE)	MN interval (RMSE | MAE)
09:00	4.82 | 2.25	3.64 | 1.7	4.26 | 1.69	4.29 | 1.67
15:00	5.81 | 2.90	4.39 | 2.11	5.19 | 2.07	4.90 | 2.07
19:00	4.55 | 2.03	3.80 | 1.50	4.17 | 1.56	4.16 | 1.57

Although the Mean absolute error (MAE) for MN is lower than MLC 2, the difference between RMSE and MAE is highest in MN. The greater difference between them, the greater the *variance* in the individual errors in the sample. RMSE penalizes large error more, which matters while comparing the generated plans with best plans

Conclusion

- Using real time data, estimates travel time better and generates best plans
- The adaptive nature of Mumbai Navigator generates a plan tree, which when followed, follows the best plan
- MLC planner generates multiple non dominating plans minimizing three criteria: travel time, variance and transfers. Picking the plan with min travel time always may not result in best possible arrival time at destination.

Future Work

- Using real time data at later stops to generate better adaptive plans
- Using actual bus data on all these planners and comparing the results

References

- Hoang Tam Vo.Enabling Smart Transit with Real-time Trip Planning.SAP Innovation Center, Singapore, 2015.
- J. Jariyasunant, D. Work, B. Kerkez, R. Sengupta, S. Glaser, and A. Bayen. Mobile Transit Trip Planning with Real–Time Data. Transportation Research Board 89th Annual Meeting, Washington, D.C., Jan. 10--14, 2010.
- M. Datar and A. Ranade. Commuting with delay prone buses. In proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, pages 22-29, 2000.
- Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. Route Planning in Transportation Networks, 2015.
- Delling, D., Pajor, T., Werneck, R.F. Round-based public transit routing. In Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX 2012), pp. 130–140. SIAM (2012).
- Justin Boyan, Michael Mitzenmacher. Improved Results for Route Planning in Stochastic Transportation Networks. In Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms Pages 895-902, 2001.
- Richard P. Guenthner and Kasimin Hamat. Distribution of Bus Transit On-Time Performance. Transportation Research Record 1202, Transit Issues and Recent Advances in Planning and Operations Technique, 1988.
- https://www.tutorialspoint.com/cplusplus/cpp_date_time.htm
- Peng Ni, Hoang Tam Vo, Daniel Dahlmeier. DEPART: Dynamic Route Planning in Stochastic Time-Dependent Public Transit Networks. IEEE 18th International Conference on Intelligent Transportation Systems, 2015.

THANK YOU

BACKGROUND SLIDES

BEST tickets data

- 52 million ticket records of size 14 GB
- Each ticket record has 35 fields
- *Issue:* Fields identification

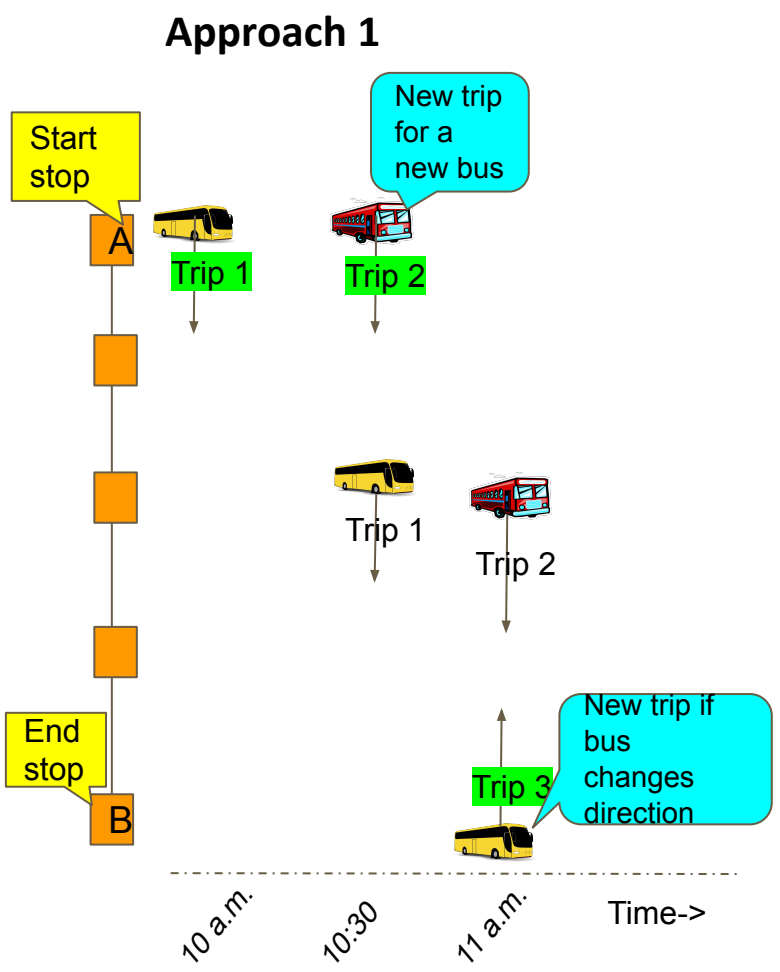
single ticket cost	total cost	tickets bought	route no	bus no	ticket date	ticket time	“from” stage no	“from” stage name	“to” stage name	“to” stage no	trip direction
8	16	2	440	196	01/07/2017	07:00	710	SHRAWAN YESHWANT CHOWK	LOWER PAREL STATION	56	U

- 514 routes and 986 stage-stops identified in the data
- 0.3% data with empty field and out of range data was removed
- **Aim:** Convert this data to a much smaller “Trip data”. Leads to easy querying.
- Conversion requirements: Grouping tickets into trips, Calculating occupancy and arrival time of trips at each stop

Assigning Trip ID

- **Aim:** Identify trips for each ticket. Assign unique trip ID to each trip
- **Approach 1** : A trip is created on a route on a day when a new bus (identified by bus no.) starts or a bus changes its direction.
  - Trip ID = *routeno_ticketdate_tripoftheday*
  - **Problem:** Incorrect for circular routes, as no direction change happens when a new trip starts.
  - **Approach 2 :** Identified a new field in the data “tripno”, s.t. combination of above Trip ID with “bus no” and “tripno” created unique Trip ID’s

Note: same “tripno” was used in data for different trips of few routes, thus combining both the approaches was required.



Occupancy of trips at stops

- User may want to board a bus where he can find a seat
- Group tickets based on trip ID “tid”
- For each trip, create a table =>
  - n = number of stops in the route
  - rowsum(p) = number of people boarded at stop p
  - colsum(p) = number of people alighted at stop p

	stop 1	. . .	stop n
stop 1	0		# tickets bought from stop 1 to stop n
. . .			
stop n	0		0

occupancy(tid, p) = occupancy(tid, p-1) + rowsum(p) - colsum(p)

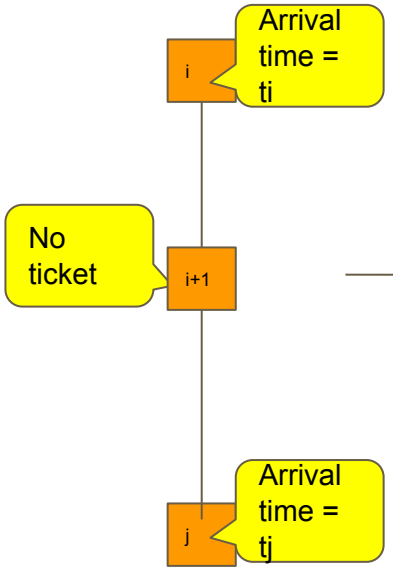
So far …
Next: Finding arrival time of trips at stops.

Trip ID	Stop number	Occupancy	Trip direction
4941_1/7/17_11	333	2	U
4941_1/7/17_11	254	6	U
. . .	. . .	. . .	
4941_2/7/17_13	412	23	D

Arrival time of trips at stops

- **Assumption:** Tickets were issued immediately at stops
- Arrival time of a trip t at a stop p = “ticket time” of first ticket issued at p on t
- **Problem** : No ticket issued at stop p on trip t
- **Solution :** Find the average time taken between stop p-1 and stop p on given route in a particular time interval using information from all past trips of the route

*Time interval matters because buses may take 8 minutes to travel between stop p-1 and p in peak hours and 3 minutes in non peak hour.



Use calculated mean times to find ratio of time taken between stop i and stop i+1 to time taken between i+1 and j in that time window.

Divide the time difference (tj - ti) to get the arrival time at stop i+1 accordingly

Arrival time at stop i+1 on trip t, with no ticket

Querying Trip dataset

- **Queries on Trip data set:**
  - What is the average occupancy of buses on route 3961 at I.I.T Market at 8:00 a.m.?
  - What is the average occupancy of buses on route 3961 at all stops at all times?
  - What is the average occupancy of all buses passing through I.I.T Market on different routes?
  - Which buses run empty?
  - Bus starting at 09:00 a.m. from Thane reaches I.I.T Market at what time?
  - What is the frequency of buses on a particular route?

Trip data set: size 460 MB

Trip ID	Stop number	Arrival time	Occ	Trip direction
4941_1_11	333	08:45	2	U
4941_1_11	254	08:52	6	U
. . .	. . .	. . .	. . .	
4941_2_13	412	09:54	23	D

* Trip ID was a combination of 5 fields. Reduced its size by replacing last three fields with a unique no.

Some outputs

```
Enter $ to quit
1. Route-wise Average occupancy
2. Stop-wise Average occupancy
3. Quit
1
1. One route, One stop at a given time
2. One route, All stops at a given time
3. One route, All stops at all times
1
Enter route number: 3961
Enter stop number: 1179
Enter time1: 07:00
Enter time2: 12:00

stageno = 1179
avg_occupancy = 24.0
```

Avg occupancy of buses on route 3961 from 07:00 to 12:00 pm at I.I.T Market

```
1. Route-wise Average occupancy
2. Stop-wise Average occupancy
3. Quit
1
1. One route, One stop at a given time
2. One route, All stops at a given time
3. One route, All stops at all times
2
Enter route number: 3961
Enter time1: 16:00
Enter time2: 23:00
stageno = 76
avg_occupancy = 4.0

stageno = 363
avg_occupancy = 28.0

stageno = 365
avg_occupancy = 26.0

stageno = 370
avg_occupancy = 31.0

stageno = 372
avg_occupancy = 31.0

stageno = 387
avg_occupancy = 29.0

stageno = 407
avg_occupancy = 24.0

stageno = 410
avg_occupancy = 18.0

stageno = 412
avg_occupancy = 18.0

stageno = 413
avg_occupancy = 17.0
```

```
stageno = 430
avg_occupancy = 9.0

stageno = 434
avg_occupancy = 6.0

stageno = 476
avg_occupancy = 17.0

stageno = 491
avg_occupancy = 16.0

stageno = 624
avg_occupancy = 22.0

stageno = 627
avg_occupancy = 25.0

stageno = 1167
avg_occupancy = 27.0

stageno = 1179
avg_occupancy = 29.0

stageno = 1181
avg_occupancy = 32.0

stageno = 1244
avg_occupancy = 2.0

stageno = 1256
avg_occupancy = 13.0

stageno = 2458
avg_occupancy = 15.0

stageno = 2476
avg_occupancy = 28.0
```

Avg occupancy of buses on route 3961 from 16:00 to 23:00

```
1. Route-wise Average occupancy
2. Stop-wise Average occupancy
3. Quit
2
1. One stop, All routes at a given time
2. One stop, All routes at all times
2
Enter stop number: 1179
RouteNO = 1850
stageno = 1179
avg_occupancy = 5.0
time1 = 23:00:00
time2 = 23:59:00

RouteNO = 1850
stageno = 1179
avg_occupancy = 7.0
time1 = 05:00:00
time2 = 05:59:00

RouteNO = 1850
stageno = 1179
avg_occupancy = 9.0
time1 = 22:00:00
time2 = 22:59:00

RouteNO = 1850
stageno = 1179
avg_occupancy = 13.0
time1 = 14:00:00
time2 = 14:59:00

RouteNO = 1850
stageno = 1179
avg_occupancy = 13.0
time1 = 16:00:00
time2 = 16:59:00

RouteNO = 1850
stageno = 1179
avg_occupancy = 13.0
```

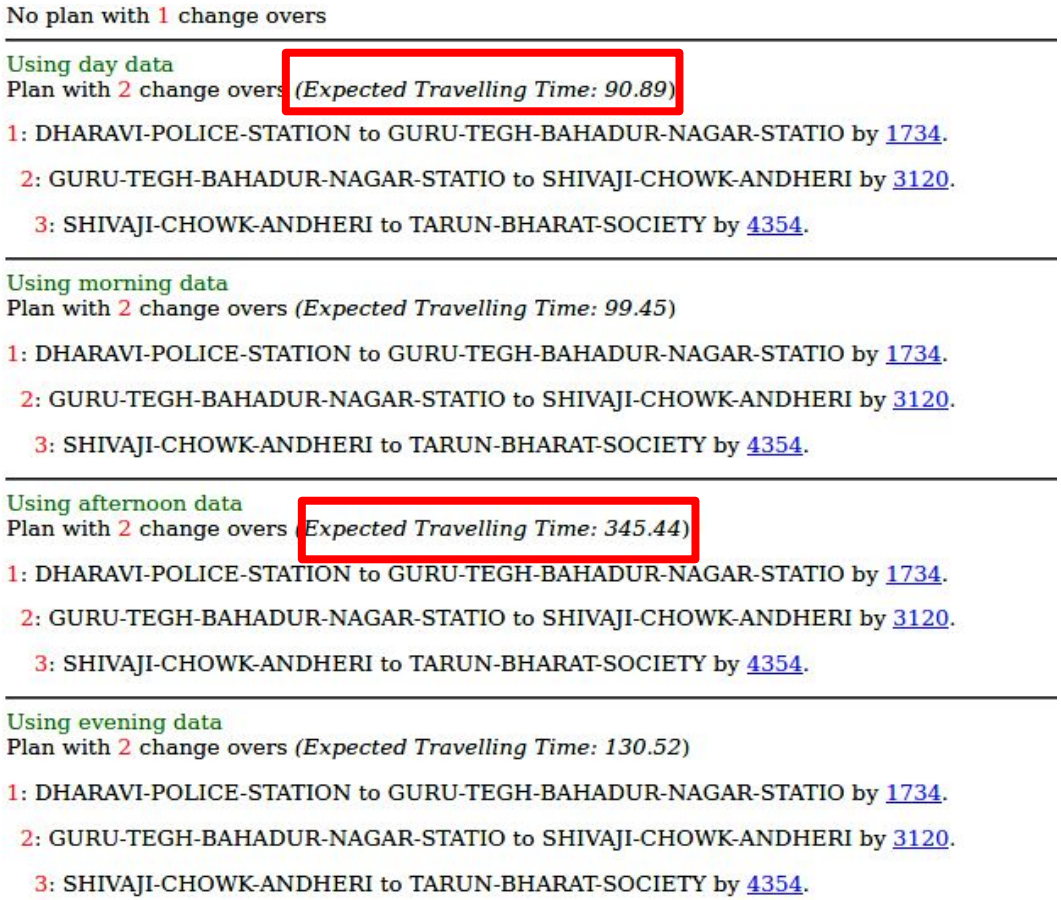
```
RouteNO = 3070
stageno = 1179
avg_occupancy = 22.0
time1 = 15:00:00
time2 = 15:59:00

RouteNO = 3070
stageno = 1179
avg_occupancy = 23.0
time1 = 13:00:00
time2 = 13:59:00

RouteNO = 3070
stageno = 1179
avg_occupancy = 23.0
time1 = 14:00:00
time2 = 14:59:00

RouteNO = 3070
stageno = 1179
avg_occupancy = 24.0
time1 = 10:00:00
time2 = 10:59:00

RouteNO = 3070
stageno = 1179
avg_occupancy = 25.0
time1 = 16:00:00
time2 = 16:59:00

RouteNO = 3070
stageno = 1179
avg_occupancy = 26.0
time1 = 09:00:00
time2 = 09:59:00

RouteNO = 3070
stageno = 1179
avg_occupancy = 26.0
time1 = 11:00:00
time2 = 11:59:00
```

```
RouteNO = 4222
stageno = 1179
avg_occupancy = 34.0
time1 = 19:00:00
time2 = 19:59:00

RouteNO = 4222
stageno = 1179
avg_occupancy = 35.0
time1 = 08:00:00
time2 = 08:59:00

RouteNO = 4222
stageno = 1179
avg_occupancy = 35.0
time1 = 20:00:00
time2 = 20:59:00

RouteNO = 4240
stageno = 1179
avg_occupancy = 9.0
time1 = 23:00:00
time2 = 23:59:00

RouteNO = 4240
stageno = 1179
avg_occupancy = 10.0
time1 = 06:00:00
time2 = 06:59:00

RouteNO = 4240
stageno = 1179
avg_occupancy = 14.0
time1 = 22:00:00
time2 = 22:59:00

RouteNO = 4240
stageno = 1179
avg_occupancy = 18.0
time1 = 21:00:00
```

Avg occupancy of buses at I.I.T Market on all routes at all times

Errors in BEST data

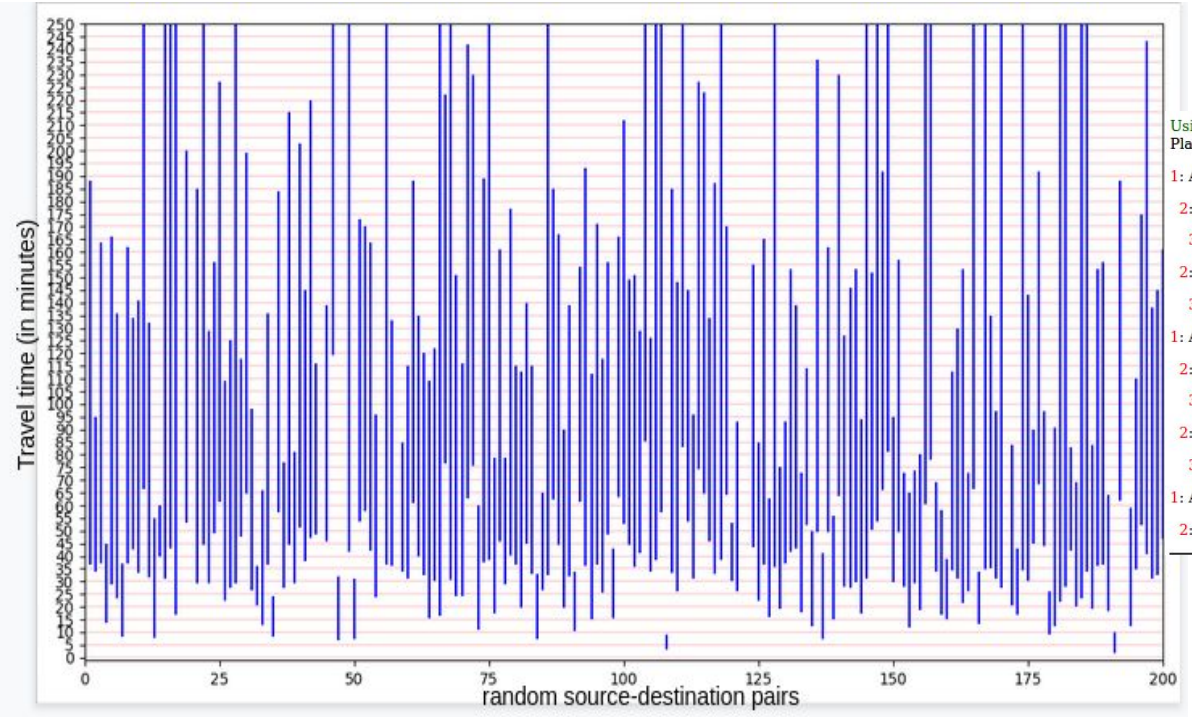
- Incomplete and noisy
- tripno field in ticketing data is not unique for a given route, date, vehicle
- wrong routeno written with tickets
- trip direction error for single directional routes
- In 11% of the trips, either incorrect time was written with the ticket or ticket was issued much later (led to earlier arrival times for later stops)
- **Example 1:** route 4354 had no buses running in afternoon, which gave large waiting time and hence large expected travelling time



Example 1

Errors in BEST data (2)

Difference between expected time given by the day plan and actual time taken on July 25th at 09 am



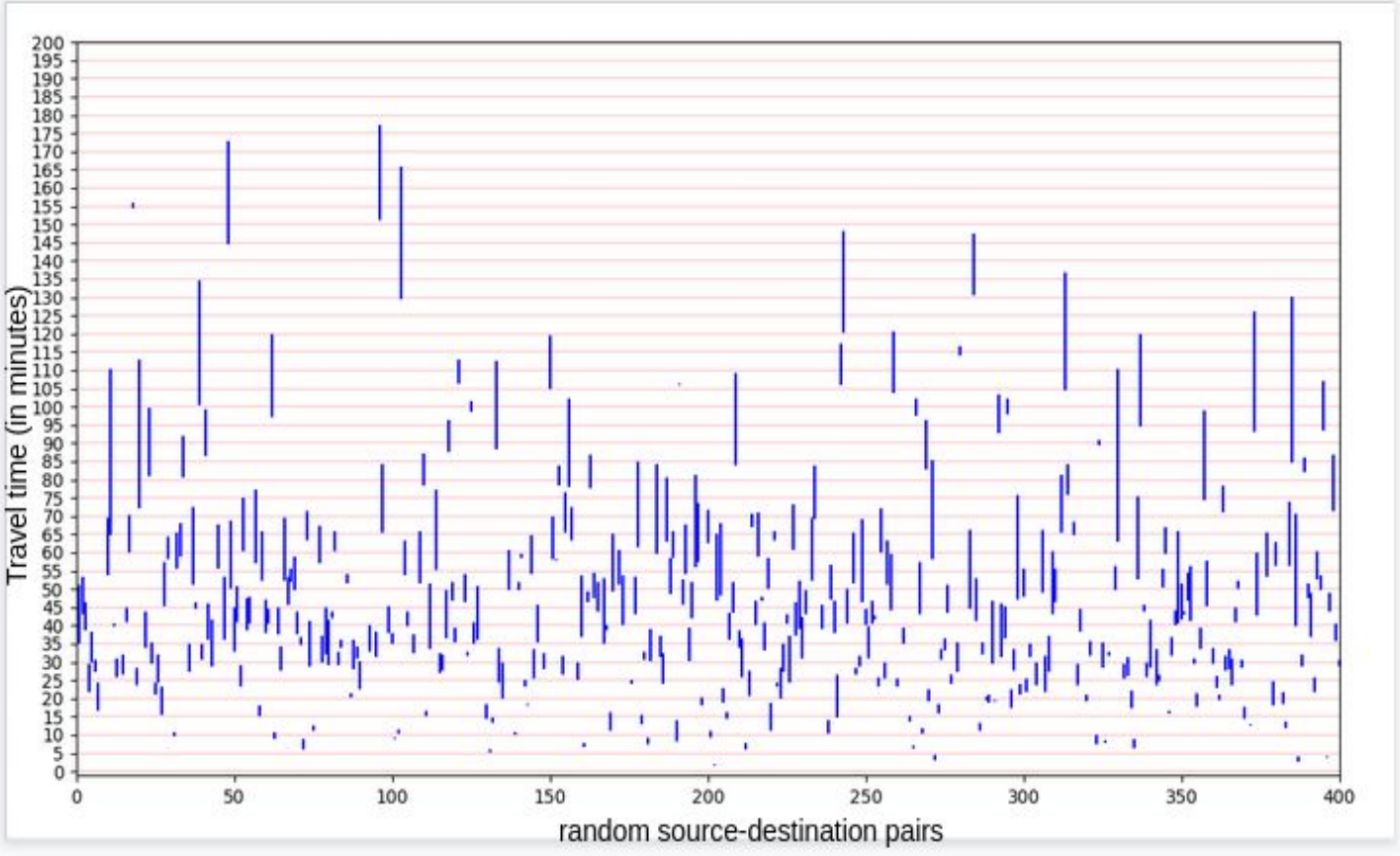
Travel plans from A.H.ANSARI-CHOWK to DR.BHADKAMKAR-ROAD

Using day data
Plan with 2 change overs *(Expected Travelling Time: 6.59)*

1: A.H.ANSARI-CHOWK to DR.M.IQBAL-CHOWK by 6[illegible],150,251,140,111,90,63.
2: DR.M.IQBAL-CHOWK to MUMBAI-CENTRAL-STATION by 1240.
3: MUMBAI-CENTRAL-STATION to DR.BHADKAMKAR-ROAD by 661.
2: DR.M.IQBAL-CHOWK to NANA-CHOWK by 1352,1350.
3: NANA-CHOWK to DR.BHADKAMKAR-ROAD by 424.
1: A.H.ANSARI-CHOWK to MUMBAI-CENTRAL-DEPOT by 620.
2: MUMBAI-CENTRAL-DEPOT to MUMBAI-CENTRAL-STATION by 1240,3510,3570,1250,630.
3: MUMBAI-CENTRAL-STATION to DR.BHADKAMKAR-ROAD by 661.
2: MUMBAI-CENTRAL-DEPOT to NANA-CHOWK by 670.
3: NANA-CHOWK to DR.BHADKAMKAR-ROAD by 424.
1: A.H.ANSARI-CHOWK to NANA-CHOWK by 670.
2: NANA-CHOWK to DR.BHADKAMKAR-ROAD by 424.

According to tickets data, this bus ran only in the night on July 25th

Travel plans comparison



- Difference between expected times of plans with 2 changeovers using day and morning database
- 400 source-destination pairs
- Different plans might be generated for different interval
- Maximum difference = 45 minutes

Plans to pick in MLC

MLC 1 = MLC planner using only past data, generating multiple non-dominating plans
MLC 1a = MLC planner using only past data, generating minimum expected travel time plan
MLC 2 = MLC planner using real time data, generating multiple non-dominating plans
MLC 2a = MLC planner using real time data, generating minimum expected travel time plan

Planner -> Departure time↓	MLC 1 (RMSE | MAE)	MLC 1a	MLC 2	MLC 2a
09:00	4.82 | 2.25	4.87 | 2.39	3.64 | 1.7	3.76 | 1.82
15:00	5.81 | 2.90	6.28 | 3.23	4.39 | 2.11	4.40 | 2.06
19:00	4.55 | 2.03	4.97 | 2.33	3.80 | 1.50	3.59 | 1.50

Results: Picking the plans with minimum expected total time amongst all non-dominating plans increases the error for almost all cases. Picking plans at random is better than picking minimum expected travel time plan.

MLC algorithm

Algorithm 8 MLC [1]

```
Input: source ns, destination nd, departure time t

Initialize priorityQueue pq, predecessorMap pm, nodeCostsList cl to null
▷ Create a label for source node, with cost as 0. Cost to reach source from source is 0
label ls = createLabel (ns , cs = 0)
▷ Insert this label into priority queue
pq.insert(ls)
▷ Main loop, runs till priority queue is empty
while pq != null do
    ▷ pops label with minimum travel time to reach from source
    label lu = pq.pop()
    node nu = lu.getNode()                                  ▷ find node corresponding to this label
    for each outgoing edge e(u,v) do
        ▷ find if edge is waiting edge or travel edge
        ▷ get the edge cost using time interval of the day
        ▷ find cost for node nv by adding cost of label lu with edge cost
        cost cv = getAccurateCost (lu , e(u,v), t)
        ▷ get all the existing costs obtained till now for node nv
        List <cost> costs = cl.getCosts(nv)
        if cv is dominated by costs then
            drop cv
        else
            cl.put(nv,cv)
            cl.removeDominated()                            ▷ Remove all dominated costs from node
            ▷ Create new label and insert into priority queue
            Label lv = createLabel(nv,cv)
            pm.put(lv,lu)
            pq.insert(lv)
        end if
    end for
end while
```

A priority queue pq is maintained to determine which node and label to explore next. It is sorted based on the mean travel time of the multi-dimensional cost. A predecessor map pm keeps track of the predecessors of the labels. For every node, there is a node cost list cl. This list stores all non-dominated costs. Initially priority queue, predecessor map, node cost list is empty. A label with node as source node and cost as 0 is created and inserted into priority queue. The main loop in the algorithm runs until the priority queue is empty and there are no more labels left to visit in the priority queue. In each iteration, the label with the lowest cost is retrieved and all of its outgoing edges are examined. Here, we find the correct cost (of the incoming node) depending on the edge type:

Once we get the cost, it is compared with existing costs of the same node. If it is better than any, a new label is created and inserted into the priority queue, and dominated costs are removed from the node. The algorithm stops when all labels are processed and the queue becomes empty.

RAPTOR algorithm

Algorithm 9 RAPTOR [5]

Input: source ps, destination pd, departure time t

Initialize multi label values for each stop to infinity
▷ Fastest way to reach source with 0 trips
$\tau_0(ps) = t$ ▷ t is the departure time
for k = 1,2 ... K **do**
  Assign $\tau_k(p)$ = Infinity
  ▷ Process each route in each round
  for each route r in R **do**
    Find first stop p' in r where $\tau_{k-1}(p')$ is defined
    current_trip = earliest trip t in r that we can catch at p' ▷ s.t. departure time of t at p' is > $\tau_{k-1}(p')$
    for each stop p_i of r beginning with p' **do**
      ▷ Update $\tau_k(pi)$ to the arrival time of current_trip at p_i if it is better
      if $\tau_{arr}(current\_trip, p_i) < \tau_k(p_i)$ **then**
        $\tau_k(p_i) = \tau_{arr}(current\_trip, p_i)$
      end if
      If we can catch earlier trip at p_i , update the current_trip
    end for
  end for
end for

In each round, each route is processed exactly once. When processing route r in round k, we consider journeys where the last (k'th) trip taken is on route r.
At each stop of route r , we find the first trip that we can catch on route r whose departure time from stop p is after the earliest arrival time at stop p with k-1 trips and update the arrival time (for round k) for next stop if it is better than the existing arrival time. This step is basically taking the solutions from the previous round and extending them by one trip if it is improving the arrival time at stops. After processing n number of routes in round k, the best arrival times at stops is either not defined or is better than round k-1, improved by taking the last trip of the journey on any of the n processed routes.
At the end of the algorithm, at each stop, for each k (from 1 to K = max transfers) we have best arrival times obtained with k trips. Parent pointers are associated with each taken trip to obtain a full travel plan.

$$O\left(K\left(\left(\sum_{r \in R} |r|\right) + |T| + |F|\right)\right.$$

RMSE vs MAE

CASE 1: Evenly distributed errors

ID	Error	|Error|	Error^2
1	2	2	4
2	2	2	4
3	2	2	4
4	2	2	4
5	2	2	4
6	2	2	4
7	2	2	4
8	2	2	4
9	2	2	4
10	2	2	4

MAE	RMSE
2.000	2.000

CASE 2: Small variance in errors

ID	Error	|Error|	Error^2
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	3	3	9
7	3	3	9
8	3	3	9
9	3	3	9
10	3	3	9

MAE	RMSE
2.000	2.236

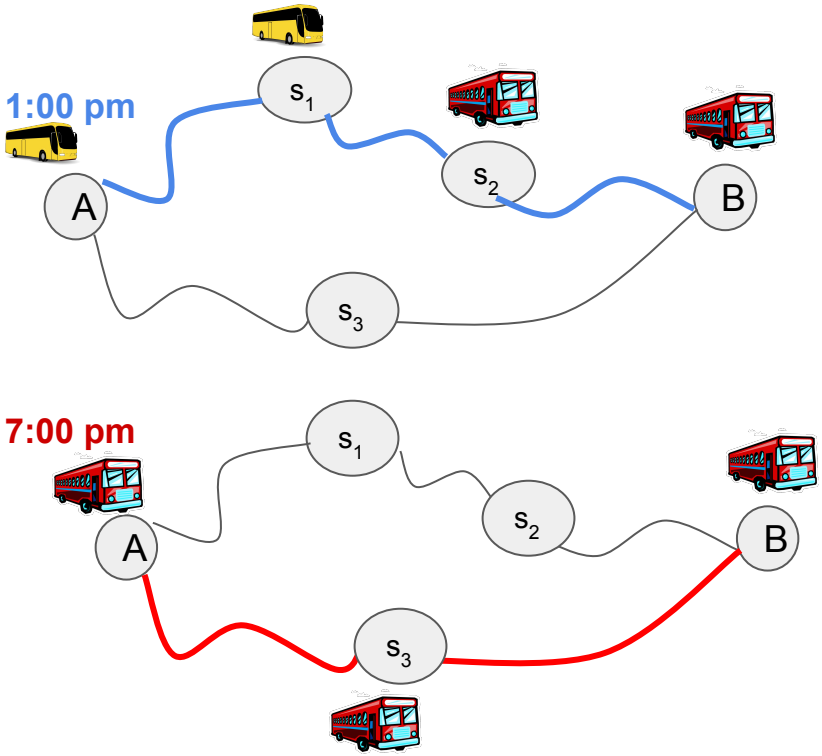
CASE 3: Large error outlier

ID	Error	|Error|	Error^2
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	20	20	400

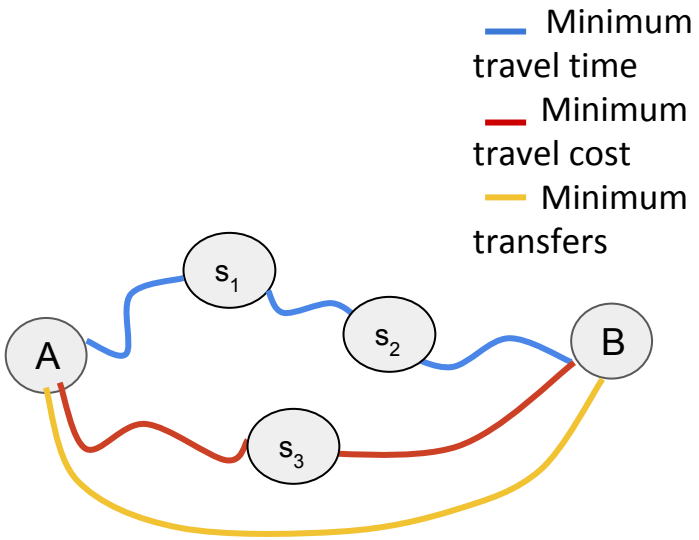
MAE	RMSE
2.000	6.325

MAE and RMSE for cases of increasing error variance

Why simple approaches don’t work?



Time dependent Nature



MultiCriteria Nature

Variants of Trip Planning

1. **Earliest Arrival Problem**: minimizes travel time
2. **Range Problem**: given range of departure time, minimizes the travel time
3. **Multicriteria Problem**: multiple criteria to optimize, returns non-dominating set of plans

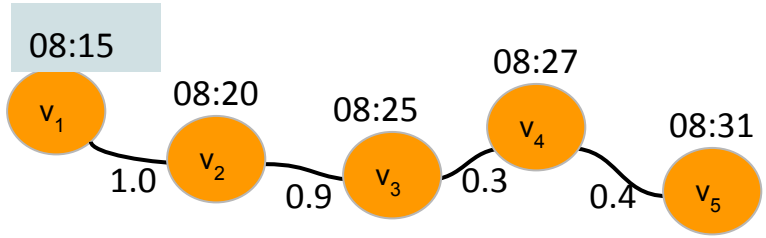
Non dominating set of plans:

Optimizing criteria (travel time, number of transfers)
Plan1: (30 minutes, 1 transfer) ✔️
Plan2: (32 minutes, 2 transfers) ❌
Plan3: (25 minutes, 2 transfers) ✔️

Network Description file stores routes, stages and stage to stage distances

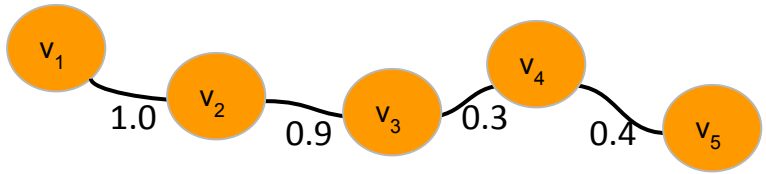
Schedule Generator generates trips for each route

Trip i on route 1_U



Trip i+1 on route 1_U

08:15 + 9.6 + 2.4



Frequency function is called whenever

Frequency Modification file

routeno	Base freq	07:00 to 08:00	08:00 to 09:00	. . .
1_U	9.6	0.5	2.4	. .
. .	. .	. .	. .	. .

*Frequency function is called every time we want to start a new trip

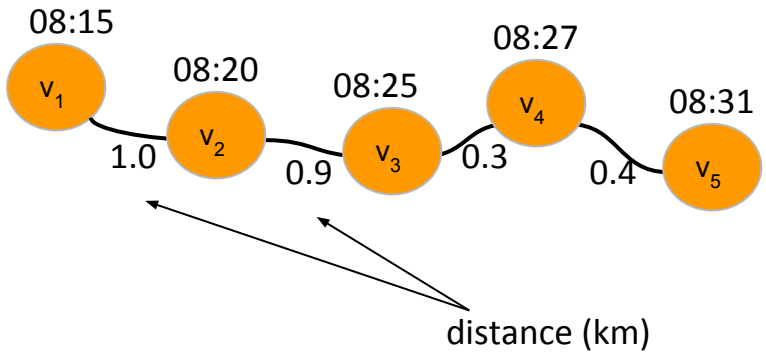
*It takes route no. and time interval as parameters

frequency (route,time interval) = base_freq+dv

Frequency (1_U, 8 to 9) = 9.6 + 2.4 = 12

*to the last trip of the route, we add the frequency as returned by the function to get the start time of the new trip on a route

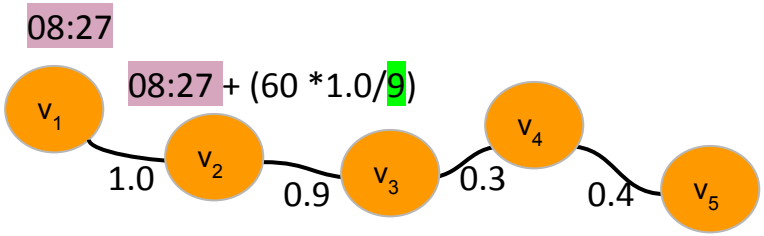
Trip i on route 1_U



Speed Modification file

stage	Base speed	07:00 to 08:00	08:00 to 09:00	. . .
v_1	10	. . .	-1	
v_2	12	. . .	-3	
v_3	11	. .	-0.5	
. .	. .	. .	. .	. .

Trip i+1 on route 1_U



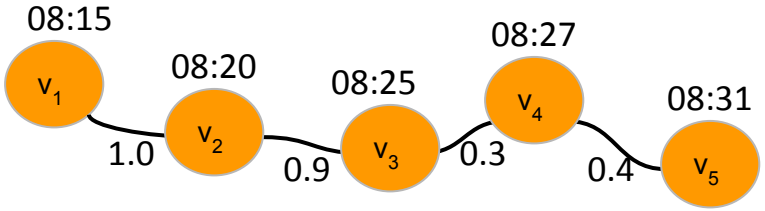
*Speed function is called to predict the arrival time of trip at next stage stop
*Average of speed of buses at current and next stage stop in the time interval and the distance between stages is used to calculate travel time between two stages

Average speed to go from v_1 to v_2= (Speed at v_1+ Speed at v_2) /2 = 9 kmph
Speed at v_1 = 10 - 1 = 9
Speed at v_2 = 12 - 3 = 9

speed(stage,time interval) = base_speed+dv

Similarly we calculate arrival time of trip at all stops and generate all trips for all routes

Trip i on route 1_U



Trip i+1 on route 1_U