Technical Report for Intermediate Stop TOD Model

Siyu LI

June 2016

Model Description

This model will predict the stop arrival time for stops on the first half-tour, stop departure time for stops on the second half-tour. Dummy variable is used to distinguish stops on the first half-tour and on the second half-tour. The intermediate stop tod model is different from tour tod model because it doesn't need to predict both stop arrival time and departure time. The departure time of stops on the first half-tour, and arrival time of stops on the second half-tour can be derived.

Choice Set

There are 48 alternatives. 3:30-3:30, 3:30-4:00,. . . , 26:30-27:00

Model Structure

The mode/destination Model is a MNL model. In the utility function, we use trigonometric series alone as time-dependent constant and trigonometric series times dummy variables to reflect a time-depdent effect of that dummy variables. A trigonometric series is defined as: $\sum_{i=1}^{k} \sin(2i\pi t/24) + \cos(2i\pi t/24)$. Different from tour tod model, this model will consider the effect of travel time and travel cost.Since the model structure is similar to tour tod, refer to specification file for details.

Variables

This section will be dedicated to the calculation of travel time and travel cost for each of the 48 alternatives.

1. Selection of origin and destination: correct origin and desination must be selected. For stops on the first half-tour, the origin is the stop location, the destination is the location of the next (remember for stops on the first half-tour, the stops are generated in counterclock-wise, the next stop is the one closer to home) stop. If the stop being model is the last stop (closest to home), the destination is home. For stops on the second half-tour, selection of origin and destination is the same.
2. Travel time: with origin and destination selected, getting travel time is trivial.

```
for (i in 1:48)
{
  print(i)
  sql_add_TT=
  paste("ALTER TABLE `hits`.`stops` ADD COLUMN `TT_",i,"` double default 999;",sep="")


  sql_attach_TT_1=paste("UPDATE `hits`.`stops`,`zone`.`Tcost`
  SET `hits`.`stops`.`TT_",i,"`=`zone`.`Tcost`.`TT_car_arrival_",i,"`",
  " where `hits`.`stops`.`origin`=`zone`.`Tcost`.`origin` and
  `hits`.`stops`.`destination`=`zone`.`Tcost`.`destination`
  and `hits`.`stops`.`mode`>=4 and `hits`.`stops`.`first_bound`=1;",sep="")

  sql_attach_TT_2=paste("UPDATE `hits`.`stops`,`zone`.`Tcost`
  SET `hits`.`stops`.`TT_",i,"`=`zone`.`Tcost`.`TT_bus_arrival_",i,"`",
  " where `hits`.`stops`.`origin`=`zone`.`Tcost`.`origin` and
  `hits`.`stops`.`destination`=`zone`.`Tcost`.`destination`
  and `hits`.`stops`.`mode`<=3 and `hits`.`stops`.`first_bound`=1;",sep="")

  sql_attach_TT_3=paste("UPDATE `hits`.`stops`,`zone`.`Tcost`
  SET `hits`.`stops`.`TT_",i,"`=`zone`.`Tcost`.`TT_car_departure_",i,"`",
  " where `hits`.`stops`.`origin`=`zone`.`Tcost`.`origin` and
  `hits`.`stops`.`destination`=`zone`.`Tcost`.`destination`
  and `hits`.`stops`.`mode`>=4 and `hits`.`stops`.`first_bound`=0;",sep="")

  sql_attach_TT_4=paste("UPDATE `hits`.`stops`,`zone`.`Tcost`
  SET `hits`.`stops`.`TT_",i,"`=`zone`.`Tcost`.`TT_bus_departure_",i,"`",
  " where `hits`.`stops`.`origin`=`zone`.`Tcost`.`origin` and
  `hits`.`stops`.`destination`=`zone`.`Tcost`.`destination`
```

```
  and `hits`.`stops`.`mode`<=3 and `hits`.`stops`.`first_bound`=0;",sep="")

  sql_attach_TT_5=paste("UPDATE `hits`.`stops`,`zone`.`Tcost`
  SET `hits`.`stops`.`TT_",i,"`=`zone`.`Tcost`.`distance_min`/5.0
  where `hits`.`stops`.`origin`=`zone`.`Tcost`.`origin`
  and `hits`.`stops`.`destination`=`zone`.`Tcost`.`destination`
  and `hits`.`stops`.`mode`=8;",sep="")


  dbGetQuery(con,sql_add_TT)
  dbGetQuery(con,sql_attach_TT_1)
  dbGetQuery(con,sql_attach_TT_2)
  dbGetQuery(con,sql_attach_TT_3)
  dbGetQuery(con,sql_attach_TT_4)
  dbGetQuery(con,sql_attach_TT_5)}
```

3. travel cost (`low_tod` and `high_tod` are the lower bound and higher bound to determine the availibility of 48 alternatives. Introduction in next section)

```
for row in am:
    AM[(int(row['origin']),int(row['destin']))]=\
    {'distance': row['AM2dis'], \
    'car_ivt' : row['AM2Tim']/60,\
    'pub_ivt':  row['AM2ivt']/60, \
    'pub_out':  (row['AM2aux']+row['AM2wtt'])/60, \
    'car_cost_erp': row['AM2ERP']/100, \
    'pub_cost': row['AM2cos']/100}

for row in pm:
    PM[(int(row['origin']),int(row['destin']))]=\
    {'distance': row['PM2dis'], \
    'car_ivt':   row['PM2Tim']/60,\
    'pub_ivt':  row['PM2ivt']/60, \
    'pub_out':  (row['PM2aux']+row['PM2wtt'])/60, \
    'car_cost_erp': row['PM2ERP']/100, \
    'pub_cost': row['PM2cos']/100}

for row in op:
    OP[(int(row['origin']),int(row['destin']))]=\
    {'distance': row['OPdis'], \
    'car_ivt':   row['OPTim']/60,\
    'pub_ivt':  row['OPivt']/60, \
    'pub_out':  (row['OPaux']+row['OPwtt'])/60, \
    'car_cost_erp': row['OPERP']/100, \
```

```
    'pub_cost': row['OPcos']/100}

am_alternative=range(10,14)
pm_alternative=range(30,34)
op_alternative=range(1,10)+range(14,30)+range(34,49)

for i in range(1,49):
        if mode in [4,5,6,7,9]:
            duration=int(row['first_bound'])*(int(row['high_tod'])-i+1)+
            int(row['second_bound'])*(i-int(row['low_tod'])+1)
            duration=0.25+(duration-1)*0.5
            parking_rate=ZONE[origin_tod]['parking_rate']
            cost_car_parking=(8*(duration>8)+duration*(duration<=8))*parking_rate
            if i in am_alternative:
                cost_car_ERP=AM[(origin_tod,destination_tod)]['car_cost_erp']
                cost_car_OP=AM[(origin_tod,destination_tod)]['distance']*0.147
                walk_distance=AM[(origin_tod,destination_tod)]['distance']
            elif i in pm_alternative:
                cost_car_ERP=PM[(origin_tod,destination_tod)]['car_cost_erp']
                cost_car_OP=PM[(origin_tod,destination_tod)]['distance']*0.147
                walk_distance=PM[(origin_tod,destination_tod)]['distance']
            else:
                cost_car_ERP=OP[(origin_tod,destination_tod)]['car_cost_erp']
                cost_car_OP=OP[(origin_tod,destination_tod)]['distance']*0.147
                walk_distance=OP[(origin_tod,destination_tod)]['distance']
            if mode in [4,5,6]:#drive1 shared2 shared3
                cost[i-1]=(cost_car_parking+cost_car_OP+cost_car_ERP)/(mode-3.0)
            elif mode==7:#motor
                cost[i-1]=0.5*cost_car_ERP+0.5*cost_car_OP+0.65*cost_car_parking
            else:#taxi
                cost_taxi=3.4+cost_car_ERP+3*central_dummy+\
                ((walk_distance*(walk_distance>10)-10*(walk_distance>10))/0.35+\
                (walk_distance*(walk_distance<=10)+10*(walk_distance>10))/0.4)*0.22
                cost[i-1]=cost_taxi
        elif mode in [1,2,3]:
            if i in am_alternative:
                cost[i-1]=AM[(origin_tod,destination_tod)]['pub_cost']
            elif i in pm_alternative:
                cost[i-1]=PM[(origin_tod,destination_tod)]['pub_cost']
            else:
                cost[i-1]=OP[(origin_tod,destination_tod)]['pub_cost']
        else:
            cost[i-1]=0
        if i in range(low,high+1):
            avail[i-1]=1
        else:
```

```
            avail[i-1]=0
    for i in range(0,48):
        fout.write('%s\t' % (cost[i]))
```

Availability of Alternatives

The availability of all 48 alternatives is determined by `low_tod` and `high_tod`. any alternative `n` satisfying `low_tod<=n<=high_tod` is available for a given stop.

- For stops on the first half-tour, the `high_tod` is the departure time of the stop, which can be derived. `low_tod` for first half-tour is bounded by time of arrival at home of previous tour, or 3 am.
- For stops on the second half-tour, the `low_tod` is the arrival time of the stop, which can be derived. `high_tod` for second half-tour is bouned by the minimum of tour primary activity arrival time of other tours.