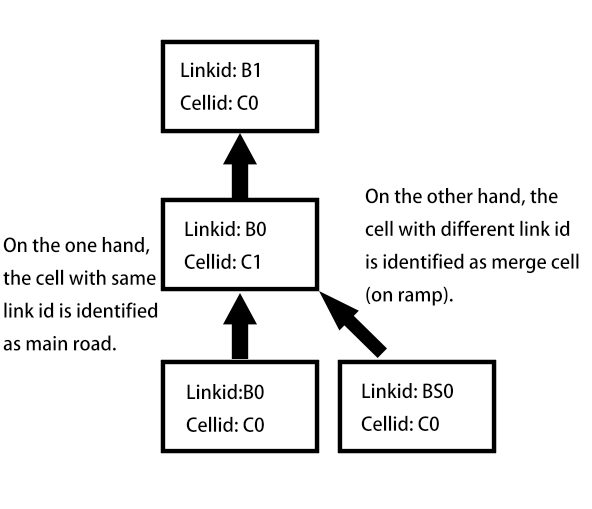
User guide and simple example

The core of cell transmission model python is Cell class, therefore it is necessary to explain Cell class first.

Attributes explanation:

Address: cellid, linkid and zoneid. These attributes represent different layer of traffic network and similar with IP address in Internet. A cell represents a road segment with about tens of meters to hundreds of meters. A link layer may contain several cells. A zone layer contains several links. Here, linkid is important because this attribute is used to identify main road and ramp. Figure below illustrates how the code works in such situation.



Other attributes are explained in table below.

|  |  |
| --- | --- |
| **Attributes** | **Meaning** |
| kjam | Jam density |
| vf | Free flow speed |
| arr\_rate | Arrival rate of traffic flow, will be used as inflow when there is no cell upstream. |
| dis\_rate | Discharge rate of traffic flow, will be used as outflow when there is no cell downstream. |
| w | Backwave speed |
| length | Length of a cell, unit is kilometer. |
| time\_sec | Time interval of simulation in second. |
| time\_hour | Time interval of simulation in hour. |
| k | Density of a cell at time interval t |
| oldk | Density of a cell at time interval t - 1 |
| updated | Flag bit of a cell that represents whether density of the cell has been updated. |
| cfrom | A list of upstream cells, objects of Cell class are saved in the list. |
| cto | A list of downstream cells, objects of Cell class are saved in the list. |

Modeling an arterial road with cell transmission model of this code can be easily conducted by following steps.

Step 1, create several Cell class objects. One may use quicklyCreateCells() method to create a series of homogeneous and connected cells. Number of cells and link layer id must be specified when use above method. If heterogeneous cells are needed, changing attributes of Cell class objects or initialize Cell class objects with different parameters.

For example, a road segment is divided into 2 cells, the first cell is named as “A0.B0.C0” and the second cell is named as “A0.B0.C1”. Parameters about id, that is cellid, linkid and zoneid must be specified. If other parameters are not given, default parameters will be used. Then, connections between cells should be created. Here, addConnection(sink) method should be used. This method is a method of Cell class and the object call this method is source (origin). Parameter of this method must be Cell class object, too, and the parameter will be identified as sink (destination). In order to delete connection between cells, method deleteConnection(sink) can be used and its usage is similar with addConnection.

**Code example:**

cell\_1 = Cell(‘C0’, ’B0’, ‘A0’) # Instantiate Cell class object with default parameters

cell\_2 = Cell(‘C1’, ‘B0’, ‘A0’, kjam=200, vf=80) # Instantiate Cell class object with specific parameters

cell\_1.addConnection(cell\_2)

Now, cell\_1 has connected with cell\_2. Traffic flow will first enter cell\_1, and then via cell\_1 flow into cell\_2. And usage of deleteConnection is given below.

cell\_1.deleteConnection(cell\_2) # Delete connection between cell\_1 and cell\_2

Step 2, for convenience, users are recommended to store all Cell class objects into a list.

Then, create a dataframe of pandas package and let address of all cells as row index of the dataframe.

Next, users may use a loop to update density of cells in each loop iteration. As simulation\_main(endtime) method shown, parameter endtime means how many simulation steps want to be conducted. Within the loop, a temporary list named density is needed for storing calculated density into dataframe.

Once density of a cell is stored into the temporary list, density of this cell is identified as updated so that it is necessary to reset flag bit attribute updated as False.

Finally, after the main loop ends, users are recommended to use to\_csv method of dataframe to create a density profile as csv format.

**Code example:**

cells = [cell\_1, cell\_2]

dfindex = []

for elem in cells:

dfindex.append(elem.getCompleteAddress())

df = pd.DataFrame(index=dfindex)

for t in range(endtime):

density = []

for elem in cells:

elem.updateDensity()

for elem in cells:

density.append(elem.k)

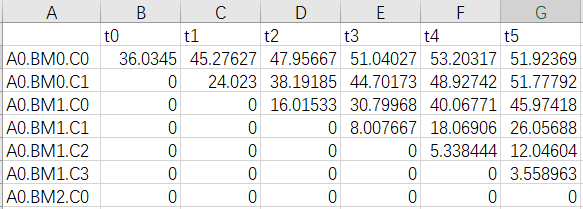
elem.updated = False

df["t%i"%t] = density

df = df.sort\_index()

return df # or use df.to\_csv(“Density profile.csv”)

Output:



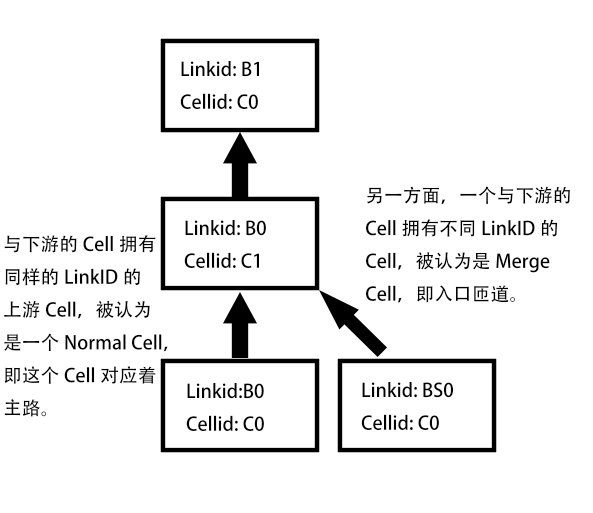
The horizontal axis means time step of simulation and vertical axis means cells. For instance, density of cell “A0.BM0.C0” in the 5th time step is 51.92369 veh/km.

用户指南以及简单示例

本程序的核心为Cell类，因此有必要首先解释Cell类的属性。

Cell类属性解释：

地址：cellid，linkid 和zoneid。这些属性用于区分不同层级的路网，与因特网中的IP地址的设计很相似，在下文中以地址来指代某个特定的cell。Cell层一般在几十米到几百米之间，而Link层一般包含数个Cell层，Zone层则包含数个Link层。这里Link层的id定义最为重要，因为在程序当中，如果遇到出入匝道，即merge和diverge cell的情况时，判断主路和匝道的依据就是Linkid。下图解释了程序如何判断主路和匝道。



其他属性在下表中解释：

|  |  |
| --- | --- |
| **属性名** | **含义** |
| kjam | Jam density，即阻塞密度 |
| vf | Free flow speed，即自由流速度 |
| arr\_rate | 交通流的到达流率，当这个cell类对象没有上游cell的时候，将它作为上游的demand。 |
| dis\_rate | 交通流的疏解流率，当这个cell类对象没有下游cell的时候，将它作为下游的通行能力。 |
| w | Backwave speed，交通波速度，即LWR模型中的反向波 |
| length | 单个cell的长度，单位是千米 |
| time\_sec | 以秒为单位的仿真时间步长。 |
| time\_hour | 以小时为单位的仿真时间步长。 |
| k | 在时间t时，cell内的交通流密度。 |
| oldk | 在时间t-1时刻，cell内的交通流密度。 |
| updated | 用于表示这个cell的密度是否被更新过的标志位。 |
| cfrom | 列表。存储这个cell上游的cell类对象。 |
| cto | 列表。存储这个cell下游的cell类对象。 |

使用本程序对主干道的交通流进行仿真，可以通过以下步骤实现：

第一步，创建数个cell类的对象实例。用户可以使用quicklyCreateCells()方法实现快速的，已经相互连接的，同质的cell的创建，需要在形参中指定创建cell的个数和link层的id。如果想创建异质的cell，则需要改变cell类对象实例的属性，或者在初始化的时候即指定好不相应的属性。

以下面的代码为例，假设一段道路被分成了两个cell，第一个cell的地址为“A0.B0.C0”，第二个cell的地址为“A0.B0.C1”。实例化一个cell类对象必须在形参中指定cellid，linkid和zoneid三个属性。其他的有关交通流的属性如果不事先指定，则将使用默认的值。接着，使用addConnection(sink)方法来建立两个cell之间的连接。作为起点的cell类对象调用addConnection方法，而参数则是作为终点的cell类对象。如果想删除两个cell之间的连接，用类似的方式调用deleteConnection(sink)方法即可。

**代码示例:**

cell\_1 = Cell(‘C0’, ’B0’, ‘A0’) # 以默认参数实例化cell类对象

cell\_2 = Cell(‘C1’, ‘B0’, ‘A0’, kjam=200, vf=80) # 在实例化cell类对象的时候指定部分参数

cell\_1.addConnection(cell\_2)

现在，cell\_1已经和cell\_2连接在了一起，交通流将先进入cell\_1，再通过cell\_1流入cell\_2。下面是删除两个cell之间连接的代码示例。

cell\_1.deleteConnection(cell\_2) # 删除cell\_1 和 cell\_2之间的连接

第二步，执行仿真，更新每个cell内的密度。为了便利，推荐用户将每一个cell类的实例都保存在一个列表中。然后，用pandas包创建一个dataframe，把所有cell的地址作为dataframe的行索引传入。接着，定义一个循环来迭代更新cell内的密度。比如，这里可以参考 simulation\_main(endtime)方法，它的参数endtime代表有多少步的仿真要执行。在循环中，要定义一个临时的列表，存储这一时刻的每个cell的密度，并在每一次循环结束时保存到dataframe中。一旦一个cell中的密度被存储到了临时的列表中，就认为这个cell的密度已经更新过了，则将updated标志位复位至False。最后，在循环结束后，使用pandas的dataframe自带的to\_csv方法输出一个csv格式的密度表。

**代码示例:**

cells = [cell\_1, cell\_2]

dfindex = []

for elem in cells:

dfindex.append(elem.getCompleteAddress())

df = pd.DataFrame(index=dfindex)

for t in range(endtime):

density = []

for elem in cells:

elem.updateDensity()

for elem in cells:

density.append(elem.k)

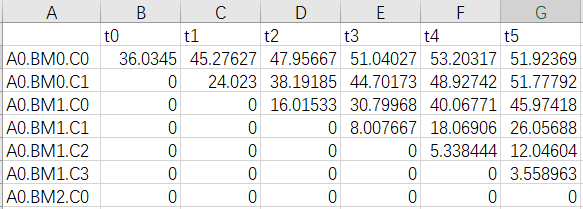
elem.updated = False

df["t%i"%t] = density

df = df.sort\_index()

return df # or use df.to\_csv(“Density profile.csv”)

输出示例:



横轴代表时间，纵轴代表不同的cell。例如，地址（ID）为 “A0.BM0.C0” 的cell在第五个时间周期的密度为 51.92369 veh/km.