

用户指南（中文版）

一、简介

本程序基于 Daganzo 教授于 1994 年提出的 Cell Transmission Model 实现, 可以进行以 CTM 为理论框架的仿真。本程序可以阅读 GMNS 格式的路网, 同时支持输入时变的交通需求, 定义时变元胞通行能力 (即交通事故等事件), 定义初始元胞密度等。

本程序对所有元胞采用类似 IP 地址的格式命名, 命名规则为 `Zone_id.link_id.cell_id`, 即 ID 的第一位为 Zone 的 id, 第二位为 link 的 id, 第三位为元胞 id。程序默认会将区域 (Zone) id 设置为 A0, 路段 (link) id 从文件中获取, 元胞 (cell) id 则在建立元胞时从 C0 开始, 数字逐个递增。

二、输入输出

本程序的输入文件为 `link.csv`, `demand.csv` 和 `supply.csv`。

2.1 路网文件

2.1.1 综述

其中, `link.csv` 可由转换路网的程序, 从 OpenStreetMap 上生成标准的 GMNS 格式路网。而在 `link.csv` 当中, 如果您需要对匝道进行仿真, 则需要在 `link.csv` 中进行额外定义。定义匝道时, 您需要指定匝道所在的通道, 即 `corridor_id` 列; 匝道的 `from_node` 和 `to_node`, 尽管本程序不包含 `node` 文件, 但仍然需要您通过指定这两个值以确定匝道与哪段路段相连。例如: 一条主路的 ID 为 1, 它的终点为节点 10001, 这时如果您指定匝道同样与 10001 相连, 则程序认为您在路段 1 的末尾处, 添加了一个合流匝道。最重要

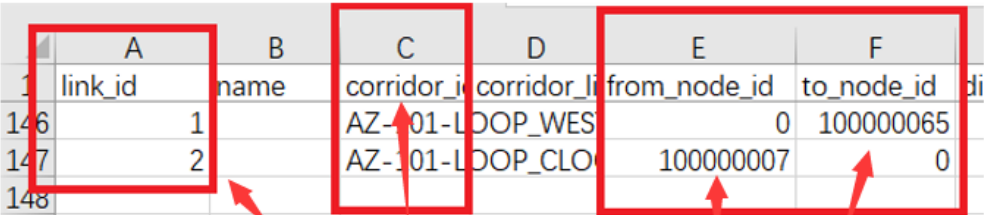
的，您需要在 ramp_flag 列指定标志位为 1，这样程序认为这条路段为一个匝道。除了上述的列以外，您只需指定一个与所有 link 的 id 均不重复的 id 给这段匝道即可，其余段皆可留空。具体地，您可参照下一小节的示例。

2.1.2 路网建立示例

当您把 link.csv 输入至程序中时，程序会自动地阅读整个文件，并建立路网。路网文件中的 corridor_link_order 列代表了一整个路段中，每条路段的顺序。程序会由低到高逐个建立相应的 CTM 路网，并依次相连。默认的元胞长度为 100 米，如果您想要更改这个设置，请在 simulation_main()函数中，传入 readNetwork()函数形参 cell_length，例如：readNetwork(cell_length=200)，此时则会以 200 米来建立元胞。本程序建立元胞的规则为：元胞个数 = int(路段长度/元胞长度) + 1。

当然，您可以通过直接调用 Cell 类来增加元胞，具体方法请参照第三章第二小节的 Cell 类方法。

下面将介绍如何建立匝道，图 1 (a) 和 (b) 展示了一个建立匝道的例子。



	A	B	C	D	E	F
	link_id	name	corridor_id	corridor_link_order	from_node_id	to_node_id
146	1		AZ-101-LOOP_WEST		0	100000065
147	2		AZ-101-LOOP_CLOVER		100000007	0
148						

定义匝道时，link_id、corridor_id和from_node_id或to_node_id必须被定义，其中link_id必须唯一且不重复，corridor_id必须与已有的一致，from_node_id或to_node_id必须被指定其中一个，另一个则设置为0。

(a)

J	K	L	M	N	O	P	Q	R
dir_flag	length	grade	facility_type	capacity	free_speed	lanes	ramp_flag	
1	200						1	

除了id以外，您也必须指定匝道的长度，长度不能小于最小元胞长度；然后，您必须将表示匝道的标志位ramp_flag置1

(b)

图 1 如何添加匝道

当 from_node_id 被指定时，则认为是合流匝道；反之则是分流匝道。默认的匝道输入流量为 0，通行比例为主：副=3：1。如果您想更改这个比例，您需要对和匝道直接相连的 Cell 中的 pk 属性进行修改，例如：Cell.getCell('A0.1.C1').pk = 0.5。在读取 link.csv 时，程序会自动地将合流匝道的最后一个元胞与主路的第一个元胞相连接；相对地，分流匝道的第一个元胞会与对应主路的最后一个元胞连接。

Cell.getCell('A0.1000001.C0').pk = 0.6

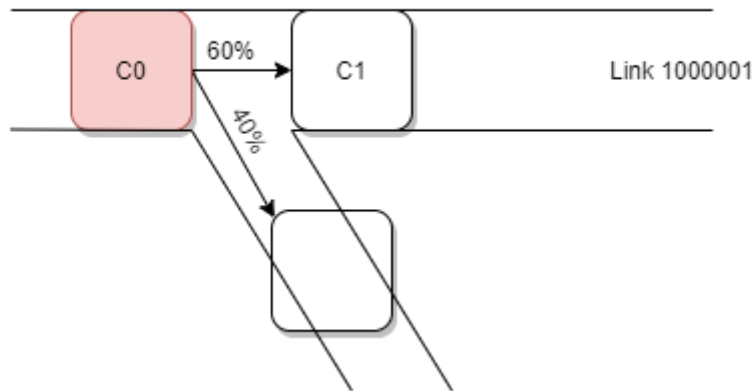


图 2 匝道比例设定示例

2.2 交通需求文件 demand.csv

定义交通需求的 demand.csv，包含了 time、corridor_id 和 demand 三列。其中，time 列为时间周期，每个周期会读取对应行的 demand，然后修改对应 corridor 的第一个 link 上第一个 cell 的 arr_rate 属性。更新 demand 的时间周期，您可以通过修改 simulation_main()

函数中的 `time_to_update_demand` 变量来实现。如果您设定了一个较小的时间周期而总的仿真时间较长，则程序会在读完 `demand` 文件中定义的需求量后，取文件中对应 `corridor` 最后一行作为此后的交通需求量。因此，如果您希望某个时刻以后交通需求为 0，您需要保证 `demand.csv` 中同一 `corridor` 的最后一行中 `demand` 值设定为 0。

2.3 辅助信息文件 `supply.csv`

最后，介绍 `supply.csv`。`supply.csv` 中，`time_period` 列需要您保证以以下格式书写：“`xxxx_yyyy`”，其中 `xxxx` 为周期开始时间，`yyyy` 为结束时间。程序会阅读整个 `corridor` 上每条 `link` 的 `time_period` 列，并会将同一 `link` 上最后一列的时间，减去第一列的第一个时间作为总的仿真时间。例如：

Corridor_id	corridor_link_order	Time_period
11	1	0600_0615
11	1	0615_0630
11	1	0630_0645
11	1	0645_0600

此时程序认为，`corrido_id` 为 11 的第一条路段上，每 15 分钟更新一次，总时间为 45 分钟。为了保证程序不出错误，您需要保证 `supply.csv` 中每一个 `link` 上的 `time_period` 为一致的。

`Supply.csv` 中，还有您还可以通过在 `volume` 列定义数值，改变每个元胞流出时的最大通行能力。例如，您可以在 `time_period` 为 0615_0630 时设置 `volume` 为 0，则此时刻该元胞被堵死，认为这个元胞对应的路段上出现了交通事故等情况导致路面被封闭。同样地，您可以通过定义 `density` 列来指定每个元胞上的初始交通密度。当然，您只需对第

一个时间周期内的 density 进行定义，其他时间周期上的 density 定义均是无效的。

F	G	H	I	J	K	L	M	
time_peri	date	geometry	volume	travel_time	speed	reference	density	cue
0600_0615	2019/1/1	LINESTRIN	1800	Capacity (Time varying)	65	Initial density	10	
0615_0630	2019/1/1	LINESTRIN	0		65		0	
0630_0645	2019/1/1	LINESTRIN	0		66		0	
0645_0700	2019/1/1	LINESTRIN	0		68		0	
0700_0715	2019/1/1	LINESTRIN	0		65		0	
0715_0730	2019/1/1	LINESTRIN	0		66		0	
0730_0745	2019/1/1	LINESTRIN	800		66		0	
0745_0800	2019/1/1	LINESTRIN	800		68		0	
0800_0815	2019/1/1	LINESTRIN	800		69		0	

图 3 通行能力和初始密度设定示例

2.4 输出文件

输出每个 corrido 上的密度和流量信息，纵轴为路段，横轴为时间。

三、类定义及方法说明

本程序的核心类为 Cell 类，用户可以通过直接调用 Cell 类来建立元胞。本章对 Cell 类的属性以及方法进行介绍。

3.1 属性

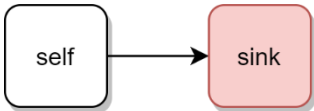
表 1 列出了程序中属性的含义。

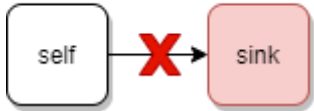
表 1 属性表

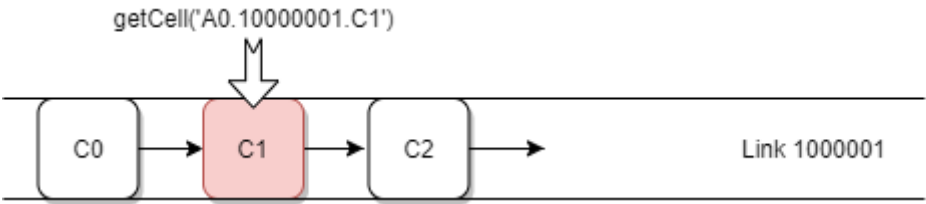
属性名	含义	默认值
Kjam	阻塞密度	220 veh/km
Qmax	通行能力	2160 veh/h
Vf	自由流速度	60 km/h
w	冲击波速度	12 km/h
cellid	元胞 id	N/A
linkid	路段 id	N/A
zoneid	区域 id	N/A
updated	表示是否更新过密度	False

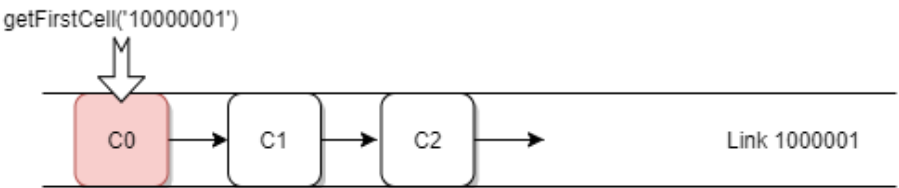
Time_interval	仿真时间区间长度	6 seconds
Arr_rate	元胞的到达流流量	0 veh/h
Dis_rate	元胞的疏解流流量	2160 veh/h
Length	长度	M
pk	主路的合流或分流比例	0.75
Ramp_flag	匝道标志位	0

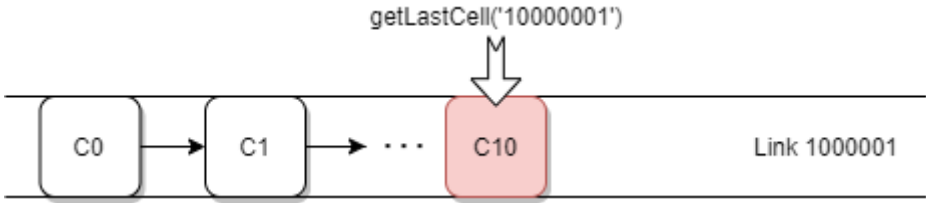
3.2 方法

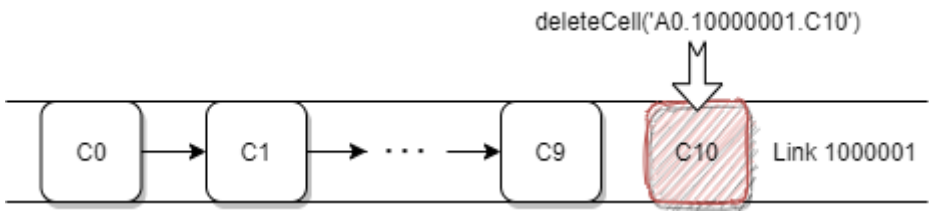
方法名	addConnection(self, sink)
传入参数	Cell 类实例
返回值	无
对调用该方法的 Cell 对象实例，与形参传入的另一个 Cell 类实例间建立一个连接。	
<p style="text-align: center;">addConnection</p>  <pre>graph LR; self --> sink</pre>	

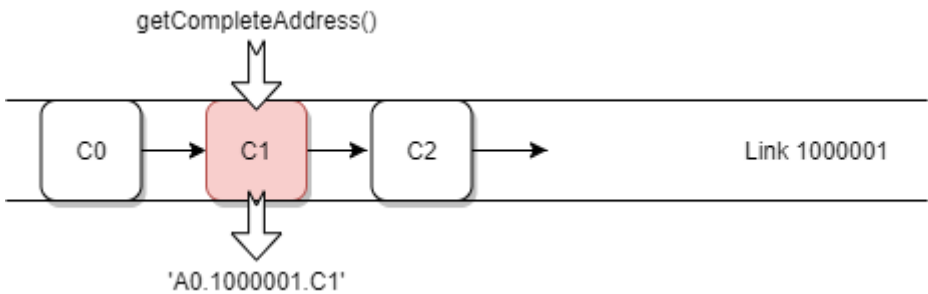
方法名	deleteConnection(self, sink)
传入参数	Cell 类实例
返回值	无
对调用该方法的 Cell 对象实例，删除与形参传入的另一个 Cell 类实例间的连接。	
<p style="text-align: center;">deleteConnection</p>  <pre>graph LR; self --X--> sink</pre>	

方法名	getCell(cid)
传入参数	字符串 cid, 传入完整的元胞 id
返回值	Cell 类实例
根据完整的元胞 ID, 返回对应的 Cell 类实例。	
 <pre> graph LR C0 --> C1 C1 --> C2 C2 --> Link[Link 1000001] style C1 fill:#f99 </pre>	

方法名	getFirstCell(linkid)
传入参数	字符串 linkid, 路段的 id
返回值	Cell 类实例
返回输入路段 ID 的第一个元胞对应的 Cell 类实例。	
 <pre> graph LR C0 --> C1 C1 --> C2 C2 --> Link[Link 1000001] style C0 fill:#f99 </pre>	

方法名	getLastCell(linkid)
传入参数	字符串 linkid, 路段的 id
返回值	Cell 类实例
返回输入路段 ID 的最后一个元胞对应的 Cell 类实例。	
 <pre> graph LR C0 --> C1 C1 --> Dots[...] Dots --> C10 C10 --> Link[Link 1000001] style C10 fill:#f99 </pre>	

方法名	deleteCell(cid)
传入参数	字符串 cid，传入完整的元胞 id
返回值	无
删除对应 ID 的 Cell 类实例。	
	

方法名	getCompleteAddress(self)
传入参数	无，Cell 类实例自身可调用
返回值	字符串，返回一个完整的元胞的 ID
对于调用该方法的 Cell 类实例，返回完整的元胞 ID。	
	

四、FAQ

Q：为什么我输入路段 ID 却没法获得到 Cell？

A：如果您输入的路段 ID 为纯数字，则程序会以 numpy.float64 格式读取并强制转化为字符串，这时，您必须指定 numpy.float64 格式的字符串，才能调用对应的方法。例如：路段 ID 为 1000001，程序里存储的路段 ID 为 '1000001.0'。因此，如果您使用 '1000001' 调用如 Cell.getFirstCell 方法，则程序会报错。

Q：为什么执行 addConnection 时报错？

A：根据 Daganzo 教授的假设，一个 Cell 不能有两个以上的连接同时与它相连。此外，

一个合流元胞不能同时是一个分流元胞，即对于同一个元胞，不能既有两个元胞流入它同时它还将流出至两个元胞。