报名号 #1115

第三届“Science Word 杯”数学中国

数学建模网络挑战赛

承 诺 书

我们仔细阅读了第三届“Science Word 杯”数学中国数学建模网络挑战赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的, 如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们允许数学中国网站(www.madio.net)公布论文，以供网友之间学习交流，数学中国网站以非商业目的的论文交流不需要提前取得我们的同意。

我们的参赛报名号为：

参赛队员 (签名) ：

队员 1：张强

队员 2：张卓

队员 3：左名浩

参赛队教练员 (签名)：

参赛队伍组别：大学组

报名号 #1115

第三届“Science Word杯”数学中国

数学建模网络挑战赛

编 号 专 用 页

参赛队伍的参赛号码：（请各个参赛队提前填写好）：

1115

竞赛统一编号（由竞赛组委会送至评委团前编号）：

竞赛评阅编号（由竞赛评委团评阅前进行编号）：

报名号 #1115

2010年第三届“Science Word杯”数学中国数学建模网络挑战赛

题　　目　基于驾驶员决策的Braess悖论模型

——以北京二环内交通为例

关 键 词　Braess悖论 路径供需理论 网络流 城市路网

摘　　要：

1968年，Dietrich Braess提出，在一交通网中增加路径可能导致新网络中的均衡交通流的通行时间不降反升，产生与预期相反的结果。本文分析了Braess悖论的路径供需机理；以路径供需理论、城市路网模型等理论为出发点，建立道路节点的最小费用流模型、排队模型和单位流费用模型；应用计算机语言对北京二环内交通的实例进行模拟分析，来判断一个路网结构存在的拥堵是否来源于Braess悖论，得到了该交通系统存在Braess悖论的结论。并探究在信息充分且对等的前提下，局中人的决策对现有交通网络中存在的Braess悖论现象的影响，通过模型得到可量化这种影响的参考值。

参赛队号　1115

参赛密码 _____
（由组委会填写）

所选题目　B

Abstract

In 1968, Dietrich Braess proposed that increase in the traffic network path may lead to a rise of time in new network equilibrium traffic flow, rather than fall, and is expected to produce the opposite result. This paper analyzes the mechanism of supply and demand in Braess paradox; we set up modeling of minimum-cost flow node modeling, queuing modeling and unit flow costing modeling, basically basing on some network traffic modeling; with computer language we make simulation analysis with the case of Beijing's inner city transportation, to determine whether the existence of a congested road network structure is derived from the Braess paradox, which proves the existence of Braess paradox in the system. And we explore how the gamers' decision-making influence on the existing transport network in the presence of the Braess paradox phenomenon under the premise of equal in the information.

基于驾驶员决策的Braess悖论模型

--以北京二环内交通为例

一、 问题重述

根据北京统计信息网[1]数据显示，北京人拥堵时花在路上的时间是顺畅时的2.2倍。随着改革开放的不断深化，首都地区的经济得到了飞速的发展，但交通问题也越来越严重。截止2007年底，北京市城八区道路总里程约为4455公里，年均增长率超过10%。道路容量不断提高，但道路服务质量却不断下降，2000年的道路运行系统测试表明，二环内的中心区，道路平均速度不到 10km/h，在高峰时段，中心区一半以上的主干道发生交通阻塞。而在复杂的城市道路当中，存在着一种Braess悖论，即在一交通网中增加路径可能导致新网络中的均衡交通流的通行时间不降反升，产生与预期相反的结果。Braess悖论常常是造成实际交通效率显著下降的最终原因。

综上所述，我们将在本文中建立模型以解决如下问题：

（1） 判断在北京市二环路以内的路网中出现的交通拥堵，是否来源于Braess 悖论所描述的情况。

（2） 如果司机广泛使用可以反映当前交通拥堵情况的GPS导航系统，是否会缓解交通堵塞，并请估计其效果。

二、 问题分析

首先，我们要收集北京市二环内中心区的有关的大量的数据，通过对模型的简化，我们需要找到北京市二环内路网主干道路的数据。但由于时间和条件限制，我们无法找到齐全的所有指标的详尽资料，于是我们把通过适当的估计将问题数据要求合理的简化，再根据具体的路况状态定态分析校正。

第一问：

1、对 Braess 悖论所描述的具体情况进行深刻的认识，并把 2002 年北京二环修善定义为 Braess 悖论中的“增加路径或某条道路的通行能力”。我们将收集 2002 年前后的北京二环的路网数据，建立衡量路网效率的费用指标。通过前后费用的对比，验证北京二环的修善是否导致了 Braess 悖论中所描述的费用反向上升。我们将从这个观点出发，建立模型求解。

2、由于“人的行为决策”很难把握，为建立模型，本文简化人的心理活动，即：驾驶员总以他个人所知的可以到达目标的最小费用路径行驶。这样，我们可

以修改最小费用的定义来模拟人的决策方向。

3、我们假设人的路径选择始终以费用最小为准则，便可以建立最大流量最小费用的图论模型分析北京二环内的主干道情况，同时简化路网，为寻优带来方便。生成模型所需数据：最大流量矩阵、费用矩阵，我们运用了 Smeed[2] 的城市中心路网模型解决了流量数据不完整的问题，建立交通节点的排队模型描述路径选择的成本。分别对北京二环路修善前后路网成本进行描述，并通过比较得到我们的结论。

第二问：

根据问题一中所建立的模型，我们修改路径费用函数，对于拥有了 GPS 的驾驶员，我们认为他具有完全的信息，即其将对堵车情况和时间花费进行衡量，做出决策。在我们的费用函数中我们模拟完全信息的驾驶员，进一步对路网费用进行衡量，从而，得到较好的结果。我们还将向驾驶员以及政府部门提出相应建议，妥善解决北京二环内的交通拥堵。

三、 模型的基本假设

1) 每个驾驶员个体均按照费用最小原则进行路线选择，费用函数能够合理描述驾驶员的决策行为。
2) 交通系统处于理想的路网容量状态，路网状态可以被驾驶员时间费用总和进行描述。
3) 每个驾驶员掌握的信息都是对等和及时的，加入 GPS 系统前他们不具有完全的信息，仅了解每条道路的通行时间（通行费用）；加入后，他们的信息是完全的，他们能够了解到当前每条道路的路况，即其他局中人的中人的决策信息。
4) 在较长的一段时间内，进入二环和驶出二环的车辆数是相似且近似常量的，即考察域内车辆数目一定。
5) 根据最大效用原则，在道路限速允许的前提下，驾驶员将尽可能的快行。
6) 为便于思考，模型考虑的是单位时间内的路网情况。
7) 道路的自然情况良好，无事故，各道路绿信比相等。

四、 模型准备与概念引入

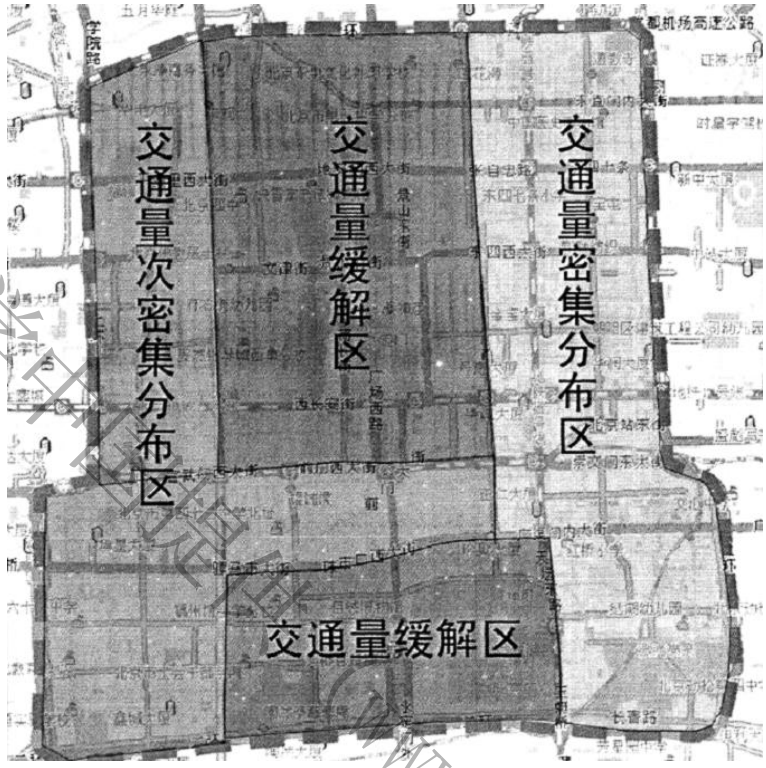
1、路网分析

北京市二环路网地处北京市中心区，路网的有效性受到以下几个因素的影响：

- 历史古迹的影响。城市中心区的历史古迹不得有城市道路穿过，因此，道路网从北到南存在几部分道路网无法覆盖的空白区域，这边增加了其周边道路系统的压力。
- 道路容量和密度的影响。除道路的长度和路宽以外，快速路、主干路、次干

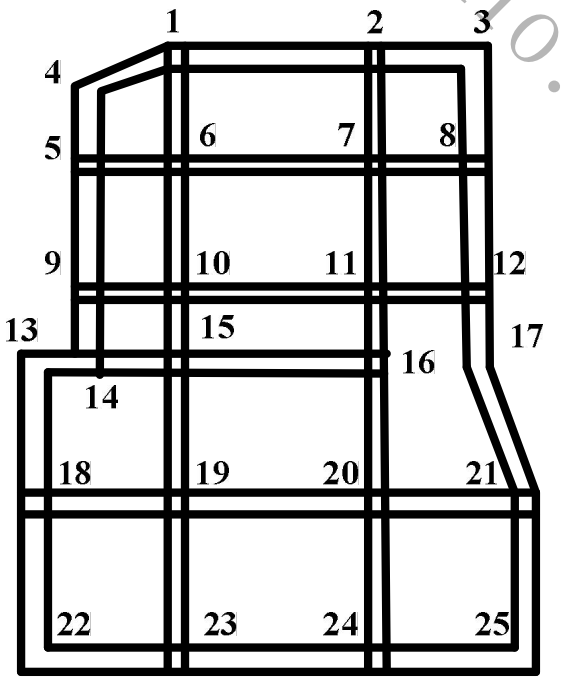
路和支路等的配比和分布很大程度上决定了其路网效率。从北京的实际情况看，北京二环内的主干道和次干路比例不均。而且南北朝向表现尤为严重，这也影响了二环内路网的效率。

因此，我们根据北京二环内部的交通量分布，选取了以积水潭桥为出发点，左安门桥为目的地的寻优端点。



2、路网简化

将北京二环路以内（包含二环路）的交通线路较为复杂，不便于研究，因此，我们将主干路罗列如下：

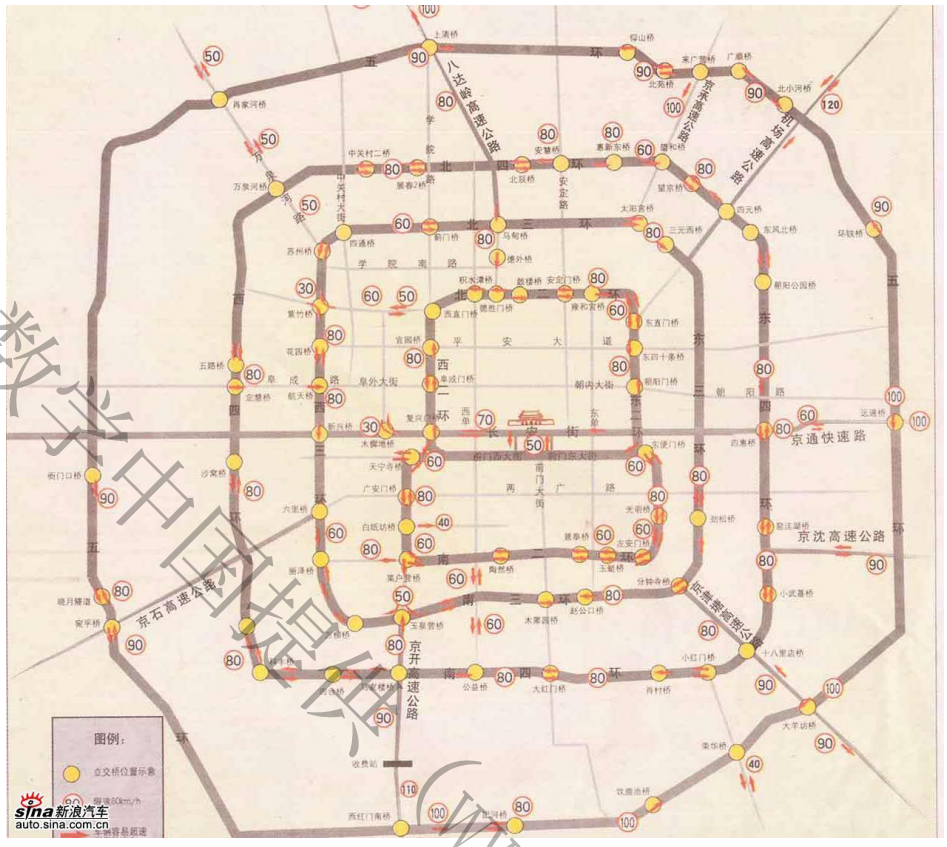


3、费用矩阵

- 将各个节点(即路口)编号为1~25,其中起点为1,终点为25利用Google Map的测距功能测出各相邻节点的距离，距离列表如下表：

I	J	S(km)	I	J	S(km)
1	2	3.80	13	14	0.70
1	4	1.60	13	18	0.90
1	6	1.70	14	9	0.90
2	1	3.80	14	13	0.70
2	3	1.50	14	15	1.30
2	7	1.60	15	10	0.75
3	2	1.50	15	14	1.30
3	8	1.60	15	16	3.80
4	1	1.60	15	19	1.00
4	5	1.10	16	11	0.75
5	4	1.10	16	15	3.80
5	6	1.50	16	20	0.85
5	9	2.70	17	12	0.75
6	1	1.70	17	21	1.30
6	5	1.50	18	13	0.90
6	7	3.70	18	19	2.00
6	10	2.80	18	22	2.30
7	2	1.60	19	15	1.00
7	6	3.70	19	18	2.00
7	8	1.40	19	20	3.60
7	11	4.30	19	23	1.90
8	3	1.60	20	16	0.85
8	7	1.40	20	19	3.60
8	12	2.60	20	21	2.10
9	5	2.70	20	24	2.30
9	10	1.40	21	17	1.30
9	14	0.90	21	20	2.10
10	6	2.80	21	25	2.40
10	9	1.40	22	18	2.30
10	11	3.70	22	23	2.20
10	15	0.75	23	19	1.90
11	7	4.30	23	22	2.20
11	10	3.70	23	24	3.90
11	12	1.50	24	20	2.30
11	16	0.75	24	23	3.90
12	8	2.60	24	25	1.80
12	11	1.50	25	21	2.40
12	17	0.75	25	24	1.80

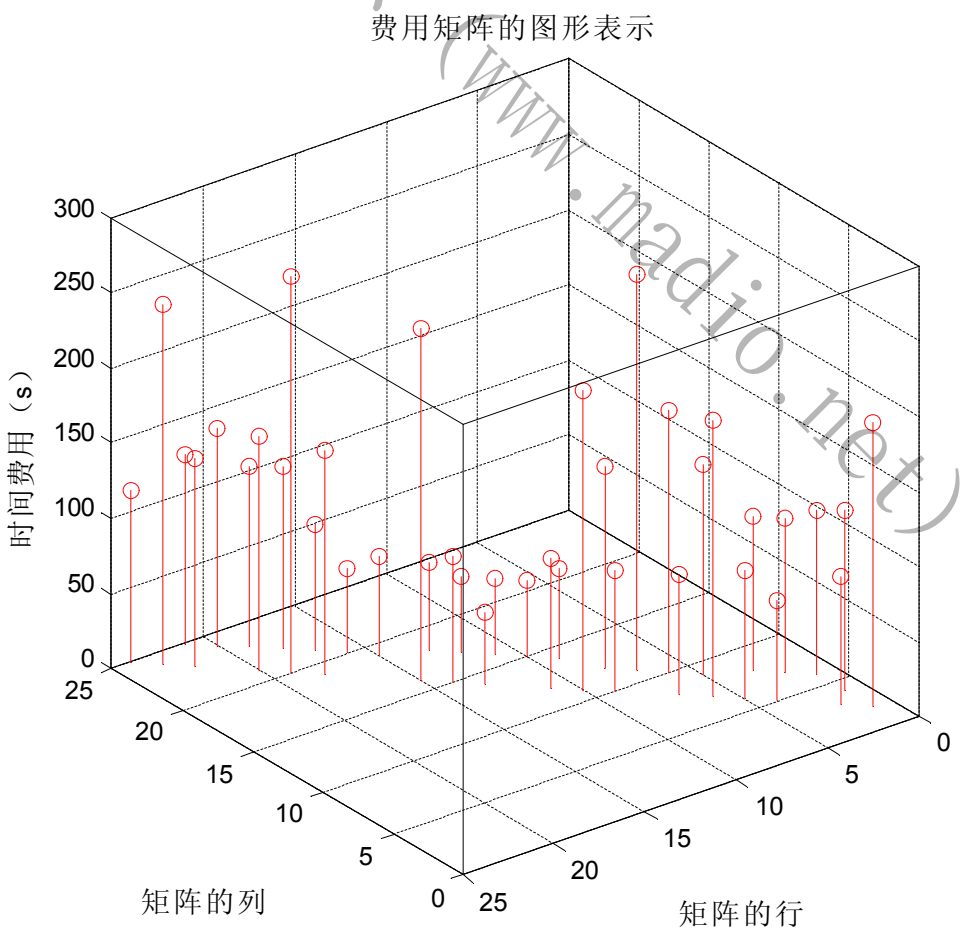
- 利用网络搜索所得北京市的限速图片，标定了各条线路的限速，图片及列表如下：[1]



I	J	V(km/h)	I	J	V(km/h)
1	2	80	13	14	60
1	4	50	13	18	80
1	6	60	14	9	60
2	1	80	14	13	60
2	3	80	14	15	60
2	7	60	15	10	60
3	2	80	15	14	60
3	8	60	15	16	60
4	1	50	15	19	60
4	5	80	16	11	60
5	4	80	16	15	60
5	6	80	16	20	60
5	9	80	17	12	60
6	1	60	17	21	60
6	5	80	18	13	80
6	7	80	18	19	50
6	10	60	18	22	80
7	2	60	19	15	60
7	6	80	19	18	50

7	8	80	19	20	50
7	11	60	19	23	60
8	3	60	20	16	60
8	7	80	20	19	50
8	12	80	20	21	50
9	5	80	20	24	60
9	10	70	21	17	60
9	14	60	21	20	50
10	6	60	21	25	80
10	9	70	22	18	80
10	11	70	22	23	60
10	15	60	23	19	60
11	7	60	23	22	60
11	10	70	23	24	60
11	12	70	24	20	60
11	16	60	24	23	60
12	8	80	24	25	60
12	11	70	25	21	80
12	17	60	25	24	60

- 利用费用矩阵的定义，我们有在不完全信息下的费用矩阵如下图：



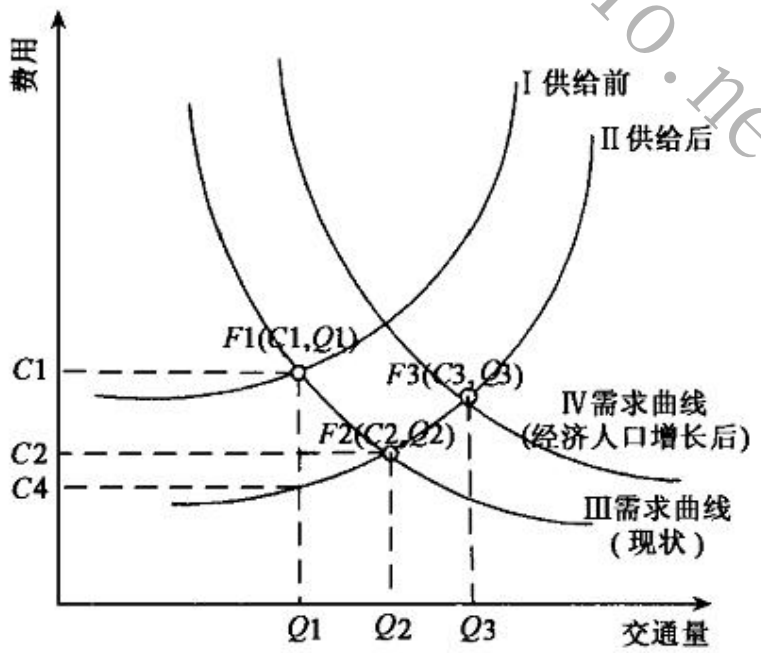
● 流量矩阵

根据《公路工程设计标准》的数据准则，我们将所得线路的设计流量作为衡量道路通行能力的标准，有下表：

i	j	公路设计流量	i	j	公路设计流量
1	2	1600	11	16	1200
2	3	1600	12	17	1200
1	4	1000	13	14	1200
4	5	1600	14	15	1200
1	6	1200	15	16	1200
2	7	1200	13	18	1600
3	8	1200	15	19	1200
5	6	1600	16	20	1200
6	7	1600	17	21	1200
7	8	1600	18	19	1000
5	9	1600	19	20	1000
6	10	1200	20	21	1000
7	11	1200	18	22	1600
8	12	1600	19	23	1200
9	10	1400	20	24	1200
10	11	1400	21	25	1600
11	12	1400	22	23	1200
9	14	1200	23	24	1200
10	15	1200	24	25	1200

4、Braess 的路径供需理论与原理[3]：

对于路径的选择，我们根据经济学的供需平衡理论，道路容量的增加会降低出行费用，出行费用的降低会刺激需求的增长，从而达到新的平衡。如下图：



5、Smeed 城市路网模型[2]

1966 年，Smeed 提出计算城市中心区路网容量的方法。定义 N 为单位 时间内有效进入城市中心区的车辆数。对于整个路网来说 ，它的容量被整个路网最先达到饱和的路段的通行能力所限制， 因此 N 的值取决于路网形态， 包括道路宽度交叉口控制类型 交通分布和车辆类型等建立模型如下。

$$N = \alpha f C\sqrt{A}$$

式中：α 为常数，取决于道路 网的结构形式，对于环状路网通常取 0.627，对于放射线和放射弧线 路网其值通常为 1.0；而不含环状的路网其值大于 2.2；为道路面积率，$\sqrt{A}$ 建成区内道路面积与建成区面积之比；C 为单位时间单位道路宽度通过的车辆数 $pcu/h\cdot m$； A 为建成区面积 。

五、 模型建立与求解

I.构建最大饱和流与路网的最大流量的相关模型：

为求得路网的最大流量，我们使用的模型为考虑了平均道路宽度和交通信号控制间距的 Wardrop 速度—流量模型【4】：

$$\frac{1}{v} = \frac{1}{v_r} + kd \qquad (1)$$

其中，平均速度 v 表示行程中的平均速度，其中包含了停车时间；行驶速度 v_r 定义为车辆行驶过程中的平均速度。k 为交通信号的分布； d 为交叉口的平均延误。

并令：

$$v_r = a(1 - q/Q) \qquad (2)$$

$$d = b/(1 - q/\lambda S) \qquad (3)$$

式中：a，b 为参数，可通过实际数据测得，q 为平均流量，Q 为设计通行能力；λ 为有效绿灯时间于信号周期之比，S 为饱和流率。综合考察后有：

$$\frac{1}{v} = \frac{1}{a(1 - q/Q)} + \frac{kb}{1 - \frac{q}{\lambda S}} \qquad (4)$$

陈春妹王雪等【3】经过实证确定二环内的数据 Q=4722(pcu / h)，kb=0.00525(经验估计值)则:

$$\frac{1}{v} = \frac{1}{51.89 - 0.00576q} + \frac{0.00525}{1 - \frac{q}{4722}} \qquad (5)$$

解出 q= 2738 pcu/h

根据Smeed[2] 提出的计算城市中心区路网容量的模型：

$$N=\frac{\alpha f q\sqrt{A}}{d} \qquad (6)$$

其中 d 为路宽,m；A 为建成区面积，m^2；α 为常数，取决于道路网的结构形式，环状路网的取值为 0.627，放射性路网为 0.1，不含环路的网大于 2.2；f 为道路覆盖面积率。

最后解得单位时间有效进入城市中心的车辆数 N=52800 辆。

II. 道路节点的排队模型

为了研究路径选择的成本，我们建立了道路节点的排队模型，车辆相继到达的时间间隔被看作服从参数为 λ 的负指数分布。车辆到达时，若有空闲车位驶出，否则选取车道排队数较少的车道等待。因此建立 n 个服务台的排队模型（这里我们估计并固定车驶出的时间为 0.5 辆每秒钟）：

$$L=\frac{c(s,\rho)\rho_s}{1-\rho_s} \qquad (7)$$

其中，L_q 表示平均队长；$c(s,\rho)$ 为车到达信号灯处需要等待的概率；ρ_s 为排队系统能否达到平衡的指标描述，$\rho_s<1$ 时系统可以达到平衡，否则将引起堵车。

对于排队系统，由 Little 公式可以求解出平均等待时间：

$$t_w=\frac{L}{\lambda_{ij}}=\frac{1}{\mu_{ij}-\lambda_{ij}} \qquad (8)$$

III. 单位流的费用确定：

根据假设，驾驶员的决策将按照最短时间来选择路径。因此我们定义费用函数：

$$f(e)=t=t_r+t_w=\frac{L}{v_r}+t_w \qquad (9)$$

来表示驾驶员选择本路径的成本。其中，L 为节点间路程，v_r 为行驶速度，t_r 为行驶时间，t_w 为停车等待时间。我们放松了 v_r 的限制，以二环内的道路限速[6] 作为建模数据。

对于线路的平均等待时间，我们将建立交通路口的排队等待模型：

在高峰期间的十字路口，我们采用 M/M/1 的排队模型研究较具有代表性的广渠门大街：根据调查的结果单方向上车流量为 1639 辆/h，平均 2s 通过一辆汽车，假设车辆到达是随机的，则排队系统中的车辆数计算如下：

$$\lambda=4/9(\text{辆}/s)，\ \mu=1/2(\text{辆}/s)$$

根据公式：$\rho=\lambda/\mu=0.89<1$，排队系统是稳定的。

排队系统中的平均车辆数：$\vec{n}=\frac{\rho}{1-\rho}=\frac{0.89}{1-0.89}=8(\text{辆})$

$$\lambda_{ij} = c_{ij} = CL_{ij}，\ \mu_{ij} = 1/2 \text{（辆/秒）}$$

对于系统中排队的逗留时间，可以证明其服从参数为的负指数分布，

$$P(T > t) = e^{-(\mu-\lambda)t}，\ t > 0 \qquad (10)$$

因此平均逗留时间为：

$$t_w = \frac{1}{\mu_{ij} - \lambda_{ij}} \qquad (11)$$

对于排队分析的结果的可得：广渠门排队系统中将有 8 辆汽车，说明尽管车辆通过界面的数量大于车辆到达的数量，系统可以达到稳定状态。然而，任然存在缓慢行驶甚至停止的汽车，同时考虑到队长于路长的关系，可知当路长小于队长时，排队车辆将占用路口空间导致堵塞。

从模型中可以看出，道路情况（路宽，车道数，路长）和车流情况（车流量、停车时间）将影响系统平衡的到达。

IV. 驾驶费用的定义

- 驾驶员的决策行为：

驾驶员感知外界道路环境信息和汽车本身的状态信息。包括通过视觉获得前方道路宽窄及曲率变化、前方有无障碍物及车辆等, 通过路感或车速表等获得汽车加速度、速度等状态信息。驾驶员根据信息加之自身考虑因素。如驾驶经验的多少、心理和生理的限制和疲劳程度等)及有关道路交通法规等, 从中决策出具体路径的选择。

- 费用函数的表示：

用 E 表示路网中可供选择的路径的集合，i，j 分别表示相邻节点。Z，G 分别体现驾驶员各种费用的目标函数向量和约束函数向量。则寻求最小费用的驾驶员选择的最优路径是多目标约束的最优化问题：

$$\min Z(e)$$

$$\begin{cases} s.t.G(e) \geq 0 \\ \forall e \in E \end{cases} \qquad (12)$$

在目标函数中存在驾驶员驾车体验的相关因素难以量化也并非重要因素，因此对于此项我们将其忽略。

对于驾驶员复杂的决策行为，我们认为主要因素为驾驶员对路况信息的判断，即在路径畅通的条件下，他总会选择时间最小的路径行驶。则有：

$$\min T(e)$$

$$\begin{cases} Q(e) \geq 0 \\ \forall e \in E \end{cases} \qquad (13)$$

其中 Q 为流量要求的约束条件，T 为我们将以此模拟驾驶员的行为决策。

- 总体费用函数的定义

$$F(e)=\sum_{Q}T_i(e) \qquad (14)$$

总体费用函数定义对于流 Q 所产生的驾驶费用的总和。其中 F 为费用加总；Q 为单位时间流量；T 为驾驶员的时间费用。

V. 驾驶模拟

我们建立了北京市二环内的简化地图取得其中 25 个节点，对于容量网络 $G=(V,E,C,t)$，单位流量的费用 $t=T(e)$，给定流 Q 时，得到可行流 $f=\{f_{ij}\}$ 使得流量为 Q，总费用最小：

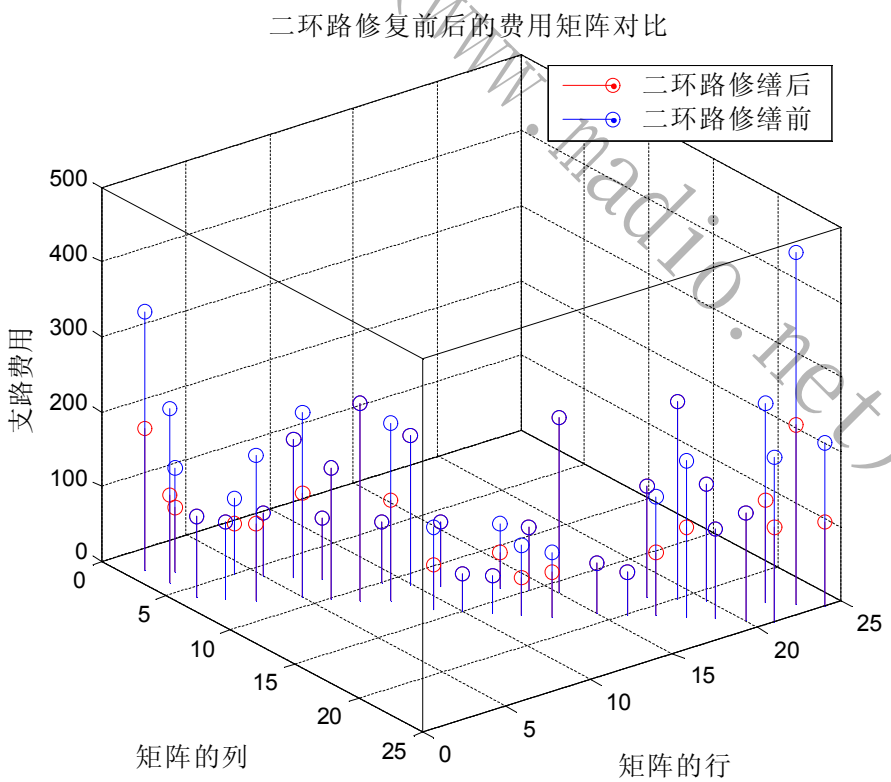
$$F(Q)=\sum_{(i,j)\in E}t_{ij}f_{ij} \qquad (15)$$

根据我们对驾驶员决策的假定，驾驶员将按照最小费用在图中寻优，等到其最适路径。

六、 模型验证与结论

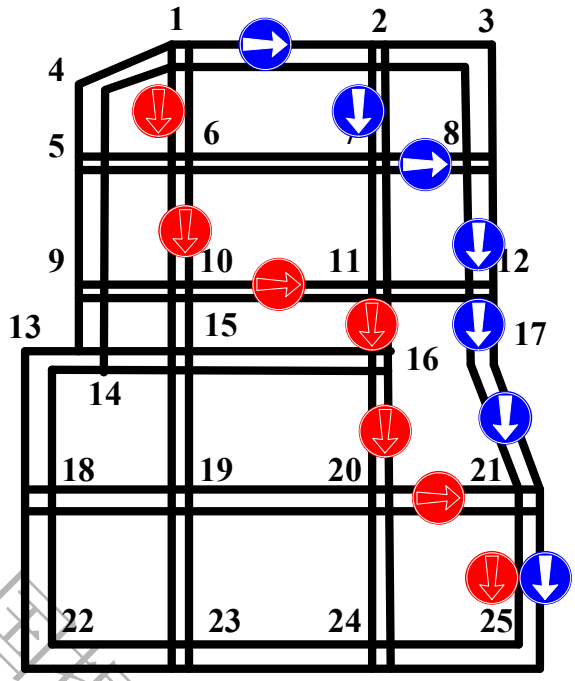
问题一：

我们利用 C++编程实现最小流算法模仿驾驶员的路径选择，对 2002 年以前和 2002 年以后的路网费用进行估计，得到两个矩阵的对比。很遗憾我们未能得到 2002 年以前的数据，我们将二环路的设计流量进行了估计行的修改得到了 2002 年以前的相关情况。这里，我们认为流量是修善后的 50%。



由此可见，修复后的二环路具有更少的费用，因此对于驾驶者具有更高的吸引力。那么，将有更多的驾驶员选择经过二环路行使，对于驾驶员的最优路径如

下图：



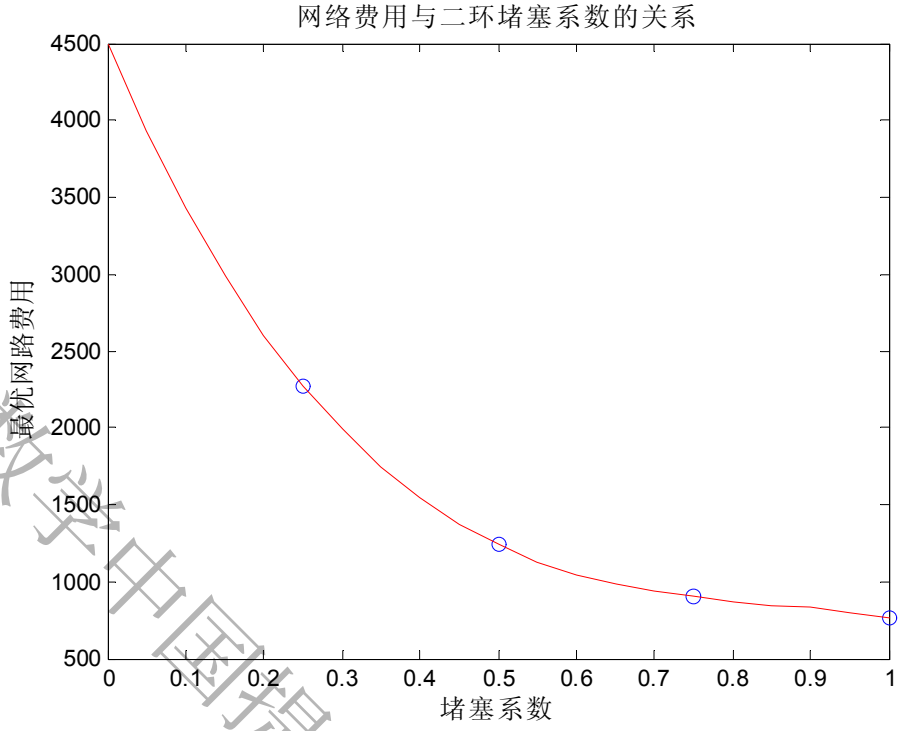
其中红色为二环修复后的驾驶员最优路线，其总费用为 952；蓝色为修复后的驾驶员最优路线，其总费用为 769。可见，驾驶员在修复后将选择经过二环，这将导致二环的流量增加，这无异于形成二环修善前的情况。对于修缮前的路网，驾驶员最优的决策费用为 965；驾驶员现在的决策在拥堵情况的费用为 1239。显然，在拥堵情况选择行驶于二环并不明智，修复后的二环反而使费用增加，形成了 Braess 悖论所描述的情况。

在选择经过二环的路径时，我们定义交通拥堵率来描述可通过流的衰减，交通拥堵率与费用的关系如下：

在流量 $Q=Q_0\cdot\theta$ 情况下的费用：

堵塞系数	0.25	0.5	0.75	1
最优费用	2272	1239	901	769

我们根据点的样条差值得到：



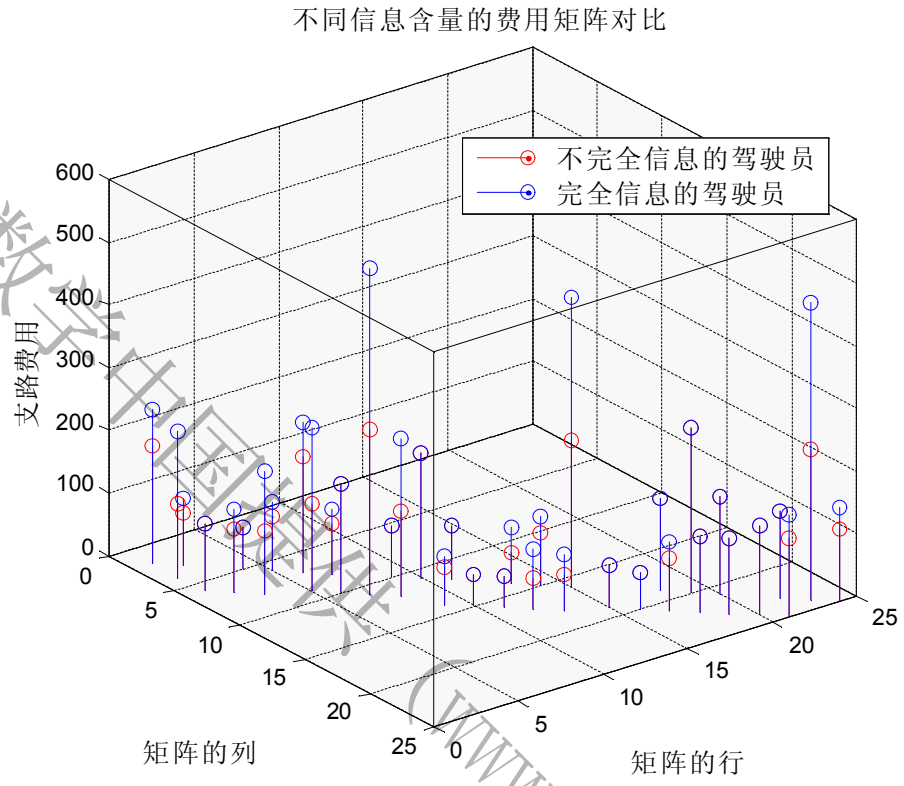
可见，二环的堵塞会导致网络最优费用的上升，因此在堵塞情况较未修复的二环通行量更差时，在 $\theta \le 0.673$ 时，将出现 Breass 悖论的现象。

问题二：

我们用经过支路的时间衡量了驾驶员路径选择的费用，并且假设不具有 GPS 的驾驶员将把理想路况作为决策的依据；拥有 GPS 的驾驶员可以指导当前的路况。对于拥有了 GPS 导航系统的驾驶员他们的信息是完全的，因此，博弈的过程中他们会更具有优势，也更能准确的衡量路径选择的费用。

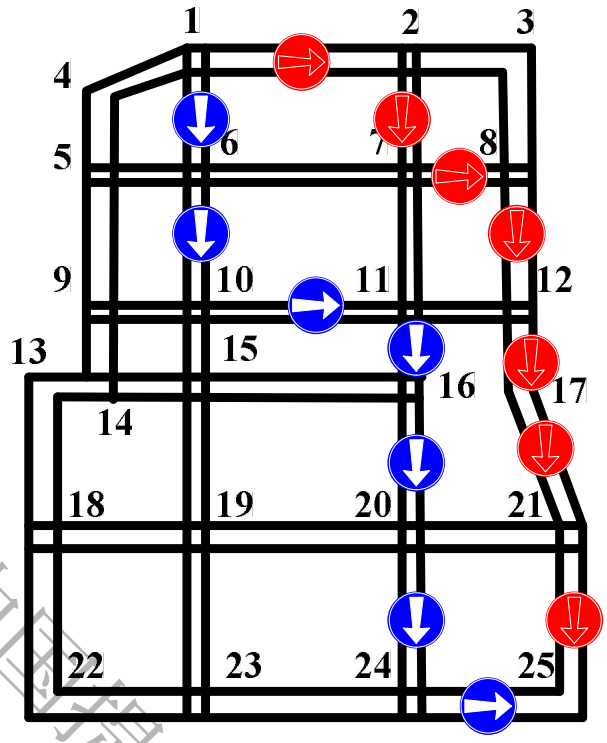
对此，我们选取次高峰时期的二环路况作为实际情况的描述：

本图来自北京市交通局，红色表示限速的 50%，黄色表示限速的 75%，绿色表示限速的 100%。 据此图的先关数据，我们得到拥有完全信息的驾驶员和信息不完全的驾驶员的费用对比：



我们认为，对于每一位驾驶员如果都按照当前路况的最优路径（费用最小）行使，那末，路网将处于最有效率状态。图中可以看出，信息完全的驾驶员较不完全的驾驶员对路况的估计更准确，也更能准的寻优，这样避免了堵车造成的延误；对于整个路网，驾驶员能够根据目前路况选择最优的路径，那么路网将避免严重堵车现象，路网效率将会提高。

信息对称与不对称的驾驶员路径比较如下图：



决策信息完全的驾驶员掌握路况信息并准确寻优，见蓝色路线，其费用为883；信息不完全的驾驶员按照理想的路况寻优，却在实际行驶中得到了较高的费用1102，见红色路线。可见，GPS可以有效分配交通流，提高路网的效率，是驾驶员具有更好的寻优能力，从而达到减少甚至消除Braess悖论的结果。

七、 模型评估

本文针对北京市二环及其内部路网的Braess现象进行分析，通过建立描述驾驶员路径选择决策的模型很好的验证了Braess现象的存在，并利用堵塞率较为准确地判断了现象的发生。模型通过微观机理分析宏观问题，具有较强的理论基础和稳定性。最小流算法通过费用最小的寻优模拟过程很好地解释了Breass现象的发生，即在对信息不完全掌握的情况下，表现出趋同性，致使交通网路效率低下，甚至堵塞。费用矩阵的计算，我们运用了排队论衡量了等待费用，较为贴近现实。模型对驾驶员行为的细致分析具有很好的实用性和理论价值，可以推广到更广的网络领域。对于难以量化的据测因素，本文给出定性的分析，费用函数尚具有开发价值，可以应用到多个个体的复杂网络中。

模型的进一步提升可以从两方面入手：对于驾驶员的决策模型可以加入其主观对信息的评价，可以利用模糊方法求解；对于最优路径的选择，可以运用较为复杂的蚁群模拟算法。同时，假设的放宽也将带给模型更好的准确性。

八、 结束语

本文从宏观交通流角度出发，建立了判定一个路网系统是否存在Braess悖论的数学模型，并根据北京市二环内路网的实测数据给出了算例，检测出其交通系统的确存在Braess悖论中的情况。再进一步探究了在信息完全的情况下，其Braess悖论的消除情况。本文虽研究了判定Braess悖论存在与否的模型，但考虑到初始假设的严格性，仍需进一步的深入研究。尽管实际问题中往往驾驶者还有许多个性化的要求，但本文考虑的仍是局中人出于对最短行驶时间的需求在决策中所表现出的趋同性，考察这种趋同性是否会引发Braess悖论并带来整个交通系统的紊乱。对于GPS所提供的信息的引入，从实际上来说也并不能完美解决局中人行为趋同的问题，随着技术的发展，我们认为GPS的导航中如果能够体现“以人为本”的思想，使用户可以通过目标函数和约束条件将个人意愿施加于路径寻优中，将最终可以导致用户路径选择行为的分散，从而有效的遏制Braess悖论的发生。我们在这里只是提出自己对整个决策趋同性的模型理解，希望能有助于探索到彻底解决Braess悖论的策略。

参考文献

【1】新浪汽车 . 新浪网 . www.sina.com . 2009-3-6 ；
【2】北京市统计局 .北京统计信息网 .www.bjstats.gov.cn .2010-4-22；
【3】王殿海. 交通流理论. 北京：人民交通出版社，2002. P80～83；
【4】赵升,王淑敏,熊天文.城市道路网容量供需模型及平衡分析: P4～5. 2003；
【5】王雪，邵春福.基于宏观交通流的城市路网容量研究： P2～3 .2003；
【6】陈春妹，任福田.理想条件下路网临界车头间距的研究： P1～2 .2002；
【7】北京交通局 . 北京交通网. www.bjjtw.gov.cn . 2010-4-22；

附录（本文中提及的源代码）

最小费用最大流算法（C++）

```
#include "iostream.h"
#include "stdio.h"
#define max 32767
#define n  25

int maxstream=1400;
int c[n][n]; //最大容量矩阵
int flow[n][n]; //实际流量矩阵
int money[n][n]; //费用矩阵
//增广链向量
int p[n];
int D[n];
int pt[n];
int stream;
```

```
int maxflow;


//------------------------------ 计 算 Vs--Vt 最 短 路 径 模 块
-------------------------------------------//

void Dijkstra()
{
int s[n];
int i,j,k,vl,pre;
int min;
int inf=32767;
vl=0;
for(i=0;i<n;i++)
{
  D[i]=money[vl][i];
  if((D[i]!=max) && (D[i]!=0)) p[i]=1;
  else p[i]=0;
}

for(i=0;i<n;i++)  s[i]=0;
s[vl]=1; D[vl]=0;
for(i=0;i<n;i++)
{
  min=inf;
  for(j=0;j<n;j++)
   if((!s[j]) && (D[j]<min))
   {
    min=D[j];
    k=j;
   }
  s[k]=1;
  if(min==max)  break;
  for(j=0;j<n;j++)
     if((!s[j])  &&  (D[j]>D[k]+money[k][j]))
     {
      D[j]=D[k]+money[k][j];
      p[j]=k+1;
     }
}


cout<<"Vs 到 Vt 的最短路径为(长和径):\n";
for(i=0;i<n;i++)
```

```
{
  if(i=n-1){
  cout<<D[i]<<"   "<<i+1;
  pre=p[i];
  while (pre!=0)
  {
   cout<<"<-- "<<pre;
   pre=p[pre-1];
  }
  cout<<"\n";}
}
}
//-------------------------------END
Dijkstra()-----------------------------------------//


//---------------用最大流算法的方法调整实际流量矩阵 flow[][]，以扩
充其流量-----------//
void modify()
{
int i,min;
int pre;
    if(D[n-1]==max)
{
  cout<<"不存在增广链";
        return;
}

pre=p[n-1];
i=n-1;
min=c[pre-1][i]-flow[pre-1][i];
while(pre!=0)
{
     i=pre-1;
  pre=p[pre-1];
  if(min>c[pre-1][i]-flow[pre-1][i])
   min=c[pre-1][i]-flow[pre-1][i];
  if(pre==1)
   pre=0;
}
    if((min+maxflow)>stream)
  min=stream-maxflow;
pre=p[n-1];
i=n-1;
```

```
flow[pre-1][i]+=min;
while(pre!=0)
{
  i=pre-1;
  pre=p[pre-1];
  flow[pre-1][i]+=min;
  if(pre==1)
   pre=0;
}
}
//--------------------------------END
modify()---------------------------------//


//------------------------------   调   整   费   用   矩   阵
money[][]-----------------------------//
void modifymoney()
{
int i,j;
int moneypre[n][n];
    for(i=0;i<n;i++)
  for(j=0;j<n;j++)
  moneypre[i][j]=money[i][j];
for(i=0;i<n;i++)
  for(j=0;j<n;j++)
  {
   if(i<j){
   if(c[i][j]!=max && c[i][j]>flow[i][j])
    money[i][j]=moneypre[i][j];
   if((c[i][j]!=max  &&  c[i][j]==flow[i][j])  ||(c[i][j]==max  &&
flow[i][j]==max))
    money[i][j]=max;}
   if(i>j){
   if(  flow[j][i]>0 )
    money[i][j]=-moneypre[j][i];
   if(flow[j][i]==0)
    money[i][j]=max;}
  }
     for(i=0;i<n;i++)
  for(j=0;j<n;j++)
  {
   if(i==j)
    money[i][j]=0;
            if(money[i][j]==-max)
```

```
    money[i][j]=max;
  }
}
//-----------------------------------END
modifymoney()-----------------------------------//

//---------------------------采用逐次逼近法得到一条增广链
----------------------------//
void approach()
{
int pf[n],ptf[n];
int min=max;
int i,j,flag;
for (i=0;i<n;i++)
{
   ptf[i]=0;
}
for(j=0;j<n;j++)
  pt[j]=money[0][j];
    do{
  flag=1;
  for(j=0;j<n;j++)
     pf[j]=pt[j];
  for(i=0;i<n;i++)
  {
   min=pt[i];
   for(j=0;j<n;j++)
   {

      if(min>(pt[j]+money[j][i]))
       min=pt[j]+money[j][i];
   }
            ptf[i]=min;
  }

  for(i=0;i<n;i++)
  {
    pt[i]=ptf[i];
     if(pf[i]!=pt[i])
    flag=0;
  }
}while(flag==0);

j=n-1;
```

```
    for(i=0;i<j;i++)
if(pt[i]+money[i][j]==pt[j])
  {
     p[j]=i+1;
        if(p[j]==1) break;
     j=i;
     i=-1;


  }
    for(i=0;i<n;i++)
  D[i]=pt[i];
}
//-------------------------------END
approach()---------------------------//


void main()
{
int i,j,k;
FILE* fp;

for (i=0;i<n;i++)
{
    for (j=0;j<n;j++)
    {
        c[i][j]=max;
        flow[i][j]=max;
        money[i][j]=max;
    }
}
for (i=0;i<n;i++)
{
    c[i][i]=0;
    flow[i][i]=0;
    money[i][i]=0;
}
fp=fopen("sj_1.txt","r");
while (!feof(fp))
{
    fscanf(fp,"%d%d%d",&i,&j,&k);
    money[i-1][j-1]=k;money[j-1][i-1]=k;flow[i-1][j-1]=0;flow[j-1][i-
1]=0;
}
```

```
fclose(fp);
fp=fopen("sj_2.txt","r");
while (!feof(fp))
{
    fscanf(fp,"%d%d%d",&i,&j,&k);
    c[i-1][j-1]=k;c[j-1][i-1]=k;flow[i-1][j-1]=0;flow[j-1][i-1]=0;
}
fclose(fp);

while (maxstream>0)
{
if (maxstream<50)
{
    stream=maxstream;
    maxstream=0;
}
else
{
    stream=50;
    maxstream=maxstream-50;
}
for (i=0;i<n;i++)
{
    p[i]=0;pt[i]=0;
}

Dijkstra();

  while( 1 )
  {
    modify();
    maxflow=0;
    for(j=0;j<n;j++)
    {
     if(flow[0][j]!=max)
   maxflow+=flow[0][j];
    }
    if(maxflow==stream)  break;
    modifymoney();
    approach();



  }
```

```
for (i=0;i<n;i++)
{
    for (j=0;j<n;j++)
    {
        if ((c[i][j]!=max)&&(flow[i][j]!=max))
        {
            c[i][j]=c[i][j]-flow[i][j];
            if (c[i][j]==0)
            {
                money[i][j]=max;
                c[i][j]=max;
            }
        }
    }
}

}

}
```