第三届“ScienceWord 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“ScienceWord 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：1102

参赛队员 (签名) ：

队员 1：

队员 2：

队员 3：

参赛队教练员 (签名)：

参赛队伍组别：大学组

第三届“ScienceWord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：
1102

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）：

2010 年第三届“ScienceWord 杯”数学中国数学建模网络挑战赛

题　　目　道路交通中 Braess 悖论的建模和评价

关 键 词　Braess 悖论 Parto 最优解 Nash 平衡 GPS

摘　　要：

本模型研究了城市交通中的 Braess 悖论问题，并对其中的一种可能的解决方法——GPS 导航的有效性进行了探讨。

本文主要对城市交通的一种典型情况——北京二环路以内的路网的交通情况进行了考察，利用北京市公安局公安交通管理局和 Google Map 提供的实时路况信息作为参考，考察 Braess 悖论对道路拥堵情况的贡献程度，并提出可能的解决方法。

问题 1、首先建立一个“日”字形的简单有向线段模型来模拟路网，运用 Nash 平衡（Nash Equilibrium）、Pareto 最优解（Pareto Optimality）和古典功利主义这三个经济学上的原理来衡量 Braess 悖论的作用和影响力，并且考虑到实际数据和情况的限制，得到了综合评价体系。然后我们从北京市二环以内拥堵较为严重的路段提取出了一个“日”字形的实际道路，并且把前面得到评价体系运用到其中，得出了结论：Braess 悖论的确是造成交通拥堵的原因之一，至少在消除某些路段（即限制某些路段的流量为零）的情况下反而能够缓解交通拥堵。

问题 2、选取一个“日”字型的道路网络模型，通过 C++编程模拟司机在配备 GPS 导航系统和没有 GPS 导航系统两种情况下对行驶路线的决策。通过对系统稳定状态下每辆车通过该系统的平均时间的比较，我们可以看到，在装备 GPS 的系统中，司机通过系统的时间有明显的减少。所以我们可以得出结论：GPS 导航系统可以显著减轻道路中的拥堵情况。

参赛队号　1102

参赛密码 _____
（由组委会填写）

所选题目　B 题：Braess 悖论

Abstract

This model discuss the Braess Paradox of its role in the transportation of urban area, and evaluate the effectiveness of one possible solution—GPS navigation.

This article uses the transportation network inside the Second Ring Road of Beijing city as the research object, which is a typical situation. Utilizing the statistics from the Traffic Administration of Public Security Department (Traffic Police Headquarters) and the Google Map, we investigate the effect of the Braess Paradox on the congestion of the traffic, and put forth the possibilities to solve this problem.

Problem1: First we construct the simple directed segment model in the style of "日" to simulate the transportation network. Under the principle of the Nash Equilibrium, Pareto Optimality and the Classical Utilitarianism, we establish the synthesis evaluation system to demonstrate the influence of Braess Paradox, taking the limited availability of the actual data into consideration. Then we extract the real road in the shape of "日" inside the Second Ring Road with heavy circumstance of congestion, analyzing it with the evaluation system mentioned above. Therefore, we can conclude that the Braess Paradox surely is one of the reasons leading to the traffic jam. At least when some roads are eliminated, which means to restrict the traffic to be zero, the congestion can be relieved.

Problem 2: We construct the model in the style of "日" to simulate the transportation network and simulate the different strategies adopted by drivers with GPS and drivers without GPS. The result shows that driver with GPS use less time than those who do not use GPS. We can reach the conclusion that GPS can significantly reduce the risk of traffic jam.

1、问题重述

Dietrich Braess 在 1968 年的一篇文章中提出了道路交通体系当中的 Braess 悖论。它的含义是：有时在一个交通网络上增加一条路段，或者提高某个路段的局部通行能力，反而使所有出行者的出行时间都增加了，这种为了改善通行能力的投入不但没有减少交通延误，反而降低了整个交通网络的服务水平。请你通过合理的模型来研究和解决城市交通中的 Braess 悖论。

(1) 通过分析实际的城市道路交通情况（自行查询的数据需给出引用来源），建立合理的模型，判断在北京市二环路以内的路网中（包括二环路）出现的交通拥堵，是否来源于 Braess 悖论所描述的情况。

(2) 请你建立模型以分析：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。

2、符号说明及模型假设

2.1 符号说明

t^k——通过第 k 条路径所用的时间

α_k——第 k 条路径自由流（即没有交通阻塞）时的出行时间

β_k——第 k 条路径的延迟参数

f_k——第 k 条路径的车流量

Q——通过所有路径的总车流量

2．2 模型假设

（1）所有的道路均为单向道路
（2）局部道路上所有车辆的起始点和终止点都相同
（3）司机都是自私的，只寻找对自己的最佳线路而不管是否损害了他人的利益
（4）司机出发前就通过对道路情况的分析决定了出行线路
（5）所有车辆的长度都相同
（6）所有车辆的行驶速度仅与路况和汽车的流量有关，与其本身性能无关
（7）所有车辆都严格遵守道路交通安全法规行驶
（8）车辆通过道路所需的时间与当前道路的流量为简单的线性关系

3、模型建立及求解

3.1 问题 1 的模型建立与求解

首先通过一个简单的“日”字形路网模型，主要运用 Nash 平衡、Pareto 最优解和古典功利主义这三个经济学上的原理构建出形成 Braess 悖论的指标系统。然后，利用北京市公安局公安交通管理局和 Google Map 提供的实时路况信息，对交通拥堵的典型情况：北京二环路以内的路网的交通情况进行了考察，并

运用之前提出的指标系统对其典型情况进行分析，评判 Braess 悖论是否是形成拥堵的一个原因。

3.1.1 简单模型

交通网络中各路段的容量，以及司机所拥有的选取不同道路到达目的地的可能性的数目直接影响到交通流在网络上的分布格局。为了分析这个问题，我们必须从每一个司机的心理的角度出发来考虑。1952 年英国道路研究所的 Wardrop 提出了“用户均衡原则”[1]这个概念，即任意一个 O（Origin）--D(Destination)对之间所有被使用的路径上的时间都是相等的，它不大于任何未被使用的路径的时间。Braess 悖论的根源就是由于司机从个人利益出发，选择出行成本最小的路径，致使系统达到均衡状态时的总出行成本增加。

下面我们就建立一个简单的路网模型来说明这个问题。这个路网模型共有 7 条有向边形成了一个“日”字形，代表了 7 条单向行驶的线路。先考虑一种最简单的情况来说明 Braess 悖论。

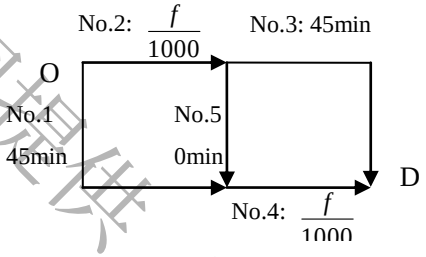

图 1、Braess 悖论说明

各条路径所需要的行驶时间如图 1 所示。在没有路径 5 的情况下，因为两条路是对称的，所以走 1、4 和 2、3 的司机人数都是相同的。假设在道路上的总人数 Q=4000，则每个司机所需要的时间均为：$45+\frac{2000}{100}=65\text{min}$。

现在产生了道路 5，仍然假设 Q=4000。则在判断走 1 还是走 2 时，因为即使每个司机都选择 No.2，所需时间仍然只有 40min，小于 No.1 的 45min。然后仍然是相同的判断，每个司机都会走 No.4，仍然需要 40min。这样，每个司机的时间反而会增加到：40+40=80min。

下面我们考虑一种更广泛的情况，如图 2 所示。

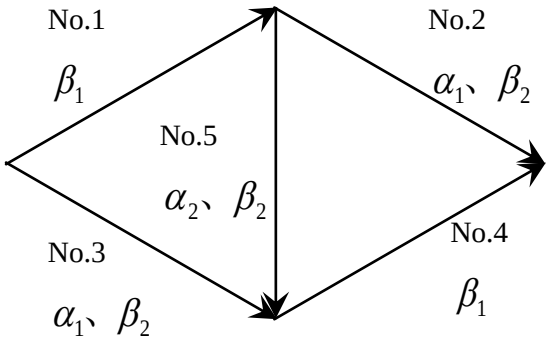

图 2、拓展的简单模型

[1]]Wardrop J G. Some theoretical aspects of road traffic research

定义 No. 1、No. 2 为路径一，No. 3、No. 4 为路径二，No. 1、No. 5、No. 4 为路径三。我们假设，通过一条路径的时间与当前该路径的流量成线性关系，其具体的表达式为：

$$t^k = \alpha_k + \beta_k f_k$$

其中，α_k 为第 k 条路径自由流（即没有交通阻塞）时的出行时间，β_k 为第 k 条路径的延迟参数。

当中间的那条路径不存在时，我们假设走下层路径的车流量为 f_1，上层路径为 $Q - f_1$。则可得：

$$t^1 = \alpha_1 + (\beta_1 + \beta_2) f_1$$

$$t^2 = \alpha_2 + (\beta_1 + \beta_2)(Q - f_1)$$

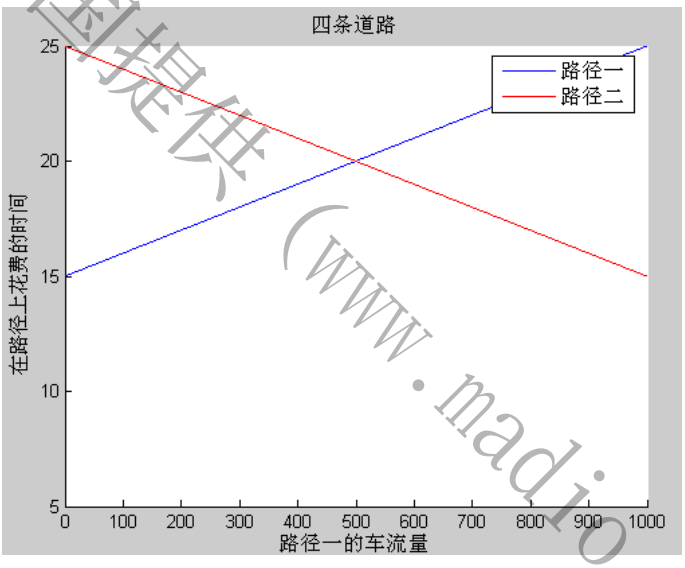

图 3、四条道路时每条路径上的时间情况

图 3 显示了路径一所需要的时间和路径二所需要的时间随路径一车流量的变化。这里我们使用的参数是 Arnott and Small's[2]在 1994 年做的实验的参数，如表一所示。

α_1	α_1	α_1	α_1	Q
15	7.5	0.01	0	1000

为了分析这个图，我们在这里需要首先引出几个经济学原理的概念，并制定出评价 Braess 悖论贡献程度的标准系统。

[2] Arnott, R. and Small, K. (1994) The economics of traffic congestion

3.1.2 评价标准的建立

首先是 Nash 平衡，其概念就是某情况下无一参与者可以独自行动而增加收益[3]。在实际的道路交通的情况中，如图 3 所示，如果路径一的车流量不在两条直线的交点处，总有一条路径的花费时间比另一条花费时间短，这种情况就不是 Nash 平衡。它造成的结果就是未来的驾驶员更加倾向于向花费时间少的路段走，使得车流量朝向图 3 中两条直线的相交点迈进。经过长时间和大量汽车的行为，总是会达到 Nash 平衡。在此情况下，任何一个驾驶员改变方向，向另外一条路行驶，一定会造成他自己的花费时间增大。这里我们假设每个驾驶员都是“自私”的，即一定不会损害自己的利益，因此最终的实际道路交通情况一定会稳定在 Nash 平衡点之上。这就要求任意一条路径的花费时间都相同。

其次是 Pareto 最优情况，其定义是指[3]：在不使其他人境况变糟的情况下，而不可能再使另一部分人的处境变好。如果一种变革能够使没有任何人处境变坏的情况下，至少有一个人处境变得更好，我们就把这个变化称为 Pareto 改进。一般地说，如果一个社会的现状不是处在 Pareto 最优状态，就存在着 Pareto 改进的可能。相应地，如果没有任何 Pareto 改进余地，就意味着现状已经达到了 Pareto 最优的状态。Pareto 最优解是合作博弈论的产物，在这样的情况下一些人群的改善必定导致另外一些人群的损失，这也是实际生活中不可能发生的，因为通常人们只关注自己的利益而忽视对他人的损害，因此一定会竭力避免 Pareto 最优情况的发生。这种最优情况是一种理想的情况，也被很多研究人员用作评判的标准[4]。然而，这种最优情况也会导致很多问题，如贫富差距过大，而且并不违反 Pareto 改进原则。如政府有 10000 元，并把这些钱发给了富人。这是 Pareto 改进，但是没有关注公平性。因此我们不认为把 Pareto 最优情况作为评价标准是最能够反映道路交通管理部门的目标。而且在图 3 中我们发现，坐标中两条直线的任何一点都能够满足 Pareto 最优情况，因此用它作标准是模糊的。这就要求我们找出另外一种评价体系：古典功利最优条件。

古典功利最优情况是指总体情况达到最优。拿本题为例，就是指所有当前在路网中的车辆的总的通行时间达到最优，也就是交通管理部门所希望达到的情况。这样必然会导致某些人的雏形时间变长。但是当我们考虑到：通过路网的司机都会在他的一生中通过无数次相同的路网，每一次他都会按照交管部门的指示或者个人“自私”的原则走不同的路线来满足体系的最优情况。因此，从长期的统计学角度来说，这 3 条路线他都是有一定概率通过的。如果我们简化这个条件为 3 条路线均等概通过，则他一生在这条路网上消耗的时间为：

$$T = \frac{1}{3} \times \sum_{i=1}^{3} t^i$$

由此可见，寻找 T 的最小值不仅可以使交通管理部门的目的即实现整个社会通行时间最短，对于个人来说，从长期大量的统计学规律来说他也会最终受益。

通过以上讨论分析，我们可以设定出讨论 Braess 悖论的评价体系：即 Nash 平衡点与古典功利最优点不重合，导致增加了一条路，或者拓展了道路容量会导致整个通行时间的增加。

[3] 百度百科 http://baike.baidu.com 2010 年 4 月 25 日
[4] 董菁, 张佐 非合作交通网络中的 Braess 悖论及其避免

3.1.3 简单模型的 Braess 悖论分析

有了评价体系以后，我们先考虑图 2 中路径一、二的最优解情况，因为它是一种特殊的两条路径对称的情况，对我们解决接下来的问题很有帮助。

通过 3.1.1 节的分析，我们直接给出通行总时间的公式：

$$T=[\alpha_1+(\beta_1+\beta_2)f_1]\times f_1+[\alpha_2+(\beta_1+\beta_2)(Q-f_1)]\times(Q-f_1)$$

由此绘制出总时间随路径一流量变化的函数图像，如图 4 所示：

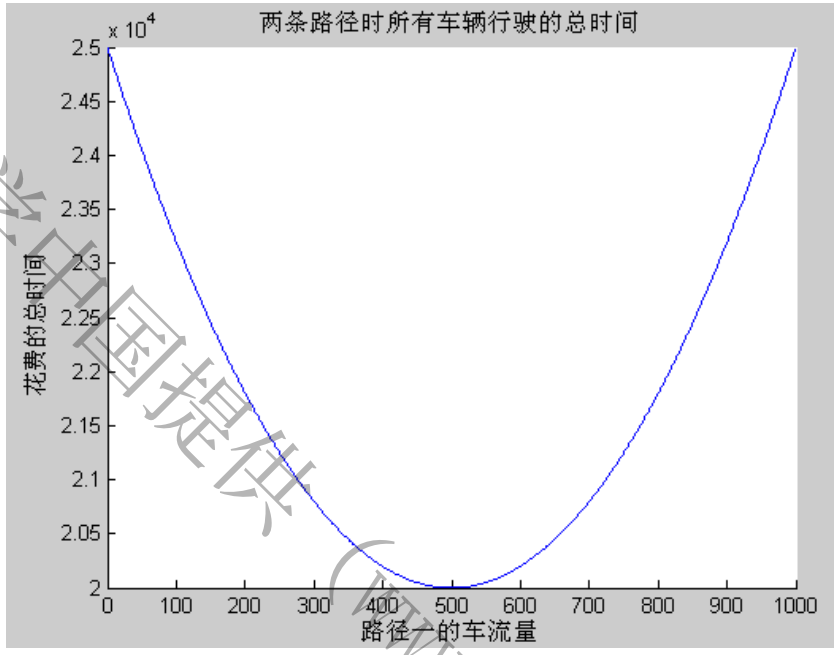

图 4、两条对称路径时行驶的总时间

从这张图我们可以看出，两条路径对称时，功利最优解等于 Nash 平衡解，即此时道路的情况为最佳，甚至不需要政府部门的监管，驾驶员会自动根据市场的需求来调整至最佳状态，其最佳的总时间为：

$$T=\frac{(\beta_1+\beta_2)Q^2}{2}+\alpha_1 Q$$

下面我们就考虑图 2 中加入 No.5 以后的道路交通情况，分析它是否会因为 Braess 悖论而导致情况变坏。

我们假设走路径一、二的车流量为 f_1 和 f_2。那么根据图 2 所示的参数：

$$\begin{cases} t^1=\alpha_1+(\beta_1+\beta_2)\times f_1+\beta_1\times(Q-f_1-f_2) \\ t^2=\alpha_1+(\beta_1+\beta_2)\times f_2+\beta_1\times(Q-f_1-f_2) \\ t^3=\alpha_2+\beta_1\times(f_1+f_2)+(2\beta_1+\beta_2)\times(Q-f_1-f_2) \end{cases}$$

通过我们对两条对称路径情况的讨论，我们可知对称路径流量相等，为 Nash 平衡解与功利最优解相同的理想最优情况。而增加了 No.5 后，路径一与路径二仍然是对称的，且路径三对它们的影响程度也是相同的。因此，为了简化求解并

且不损失正确性，我们定义 $f_1 = f_2$。于是上式变为：

$$\begin{cases} t^1 = t^2 = \alpha_1 + (\beta_1 + \beta_2) \times f + \beta_1 \times (Q - 2f) \\ t^3 = \alpha_2 + 2\beta_1 f + (2\beta_1 + \beta_2) \times (Q - 2f) \end{cases}$$

我们绘出图像：

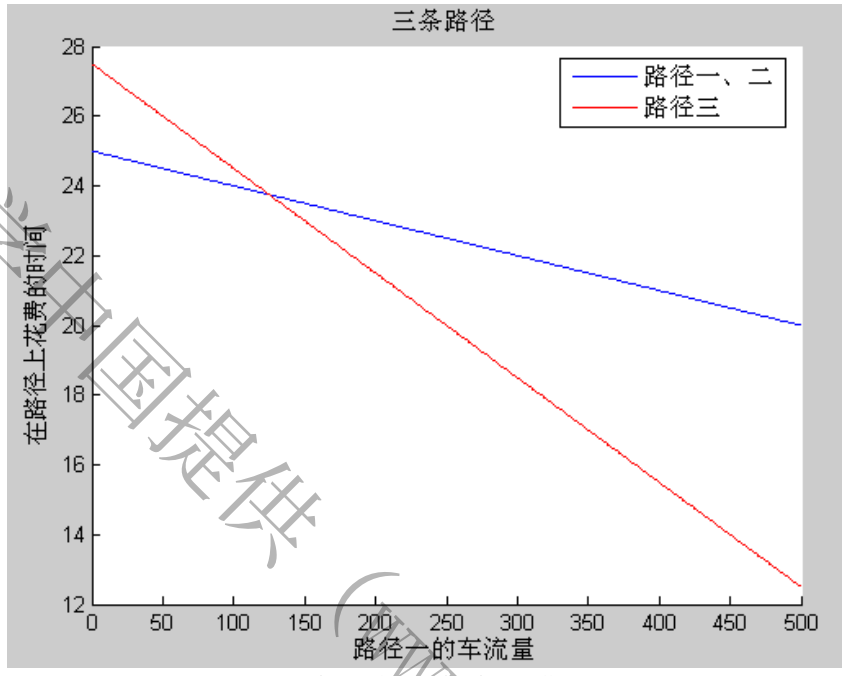

图 5、三条路径时每条花费的时间

下面我们绘出总时间，并计算出古典功利最优解：

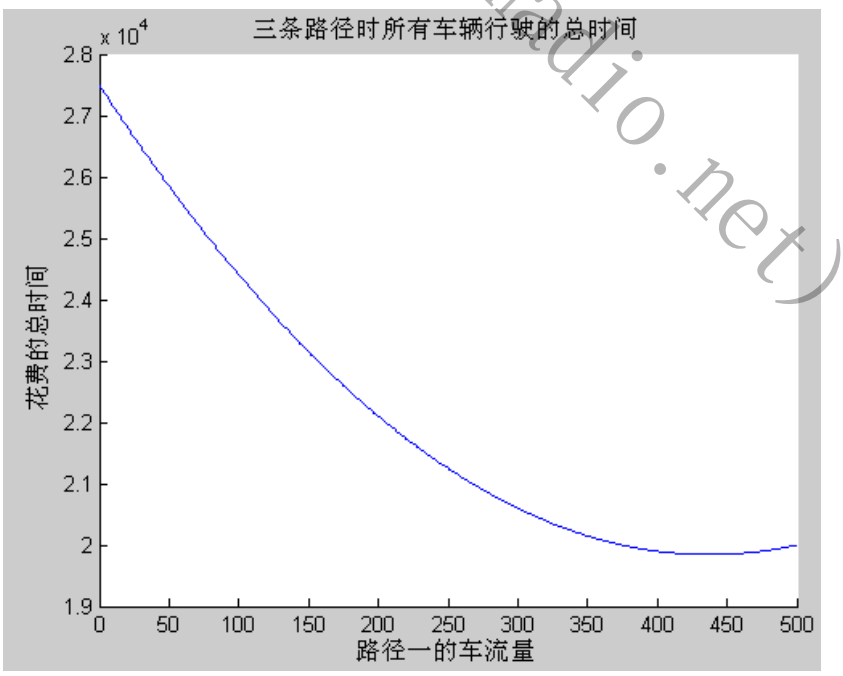

图 6、三条路径所有车花费的总时间

以上的图均根据表一所示的数据进行绘制。

从中我们可以看出 Nash 平衡解发生在 f=126 处，而古典功利最优解发生在 f=431 处。由于古典功利最优解与用户均衡解不相等，此时，必定会发生 Braess 悖论。验证如下：

路网仅有两条路径时，不管走哪条路径，司机所用的时间为：

T=20min（两条路的流量均为 500）。

路网扩展为三条路径时，不管走哪条路径，司机所用的时间为：

T=23.75min（路径一、二的流量均为 126）。

下面我们计算参数情况下的 Nash 平衡解和古典功利最优解。由 Nash 平衡解的定义，三条路径所用的时间均相等，即 $t^1 = t^2 = t^3$。故得：

$$\begin{cases} f_1 = f_2 = \dfrac{\alpha_2 - \alpha_1 + Q(\beta_1 + \beta_2)}{\beta_1 + 3\beta_2} \\ f_3 = Q - f_1 = \dfrac{\alpha_1 - \alpha_2 + 2Q\beta_2}{\beta_1 + 3\beta_2} \end{cases}$$

$$t^1 = t^2 = t^3 = \alpha_1 + Q\beta_1 + (\beta_2 - \beta_1)[\frac{\alpha_2 - \alpha_1 + Q(\beta_1 + \beta_2)}{\beta_1 + 3\beta_2}]$$

则总的时间为：

$$T = Q\{\alpha_1 + Q\beta_1 + (\beta_2 - \beta_1)[\frac{\alpha_2 - \alpha_1 + Q(\beta_1 + \beta_2)}{\beta_1 + 3\beta_2}]\}$$

古典功利最优解要求总的时间最短，因此：

$$T = 2f \times [\alpha_1 + (\beta_1 + \beta_2) \times f + \beta_1 \times (Q - 2f)] + (Q - 2f)$$
$$\times [\alpha_1 + \alpha_2 + 2\beta_1 f + (2\beta_1 + \beta_2) \times (Q - 2f)]$$

这是一个二次方程，得到最小值时：

$$f = \frac{\frac{1}{2} \times (\alpha_2 - \alpha_1) + Q(\beta_1 + \beta_2)}{\beta_1 + 3\beta_2}$$

对比 Nash 平衡解中的 f_1，我们可以发现当 $\alpha_1 \neq \alpha_2$ 时，Nash 平衡解与古典功利最优解不相等，因此就会产生 Braess 悖论。

3.1.4 简单模型的 Braess 悖论分析

接下来我们针对北京二环以内的某一具体路段进行数据分析和研究。

图 7 是 2010 年 4 月 24 日 20:08 在北京市公安局公安交通管理局网站上的路况信息截图。

图 7 北京二环道路状况图[5]

图中红色表示路段车辆严重拥堵，时速在 15km 以下，黄色表示路段车辆行驶缓慢，时速大于 15km 小于 40km，绿色表示路段车辆行驶畅通，时速 40 公里以上。

根据图 7 中的路网结构以及前文所述的“日”字路网模型，我们可以把北京市崇文门大街附近的道路情况抽象成数学模型如图 2 所示。

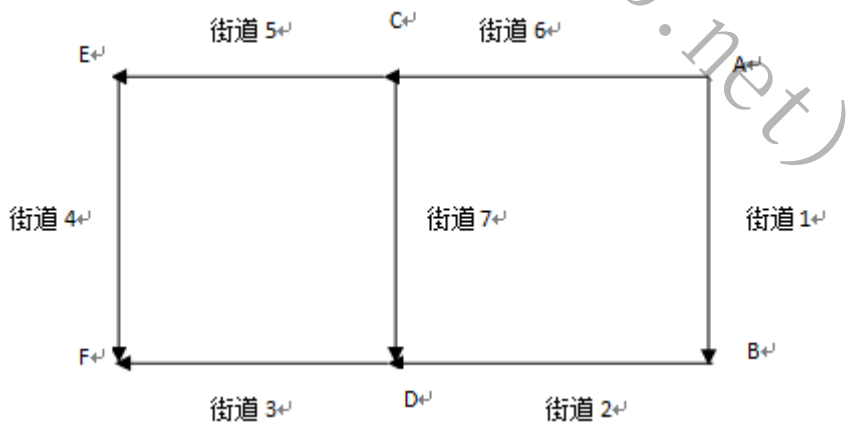

图 8 实际道路的数学抽象模型

其中图 8 中各个街道的参数见表 2 所示：

[5] 北京市公安局公安交通管理局网站 http://www.bjjtgl.gov.cn/ 2010 年 4 月 24 日

	街道 1	街道 2	街道 3	街道 4	街道 5	街道 6	街道 7
长度 L(m)	781	751	533	805	548	758	793
自由速度 V_f (m/s)	34.8	37.2	38.1	28.7	29．1	37.5	33.2

表二、街道的情况

表二中，街道的长度是根据网站中的比例尺计算得出的。自由速度V_f为街道中无时速限制以及安全隐患时车辆自由流的速度。该数据是根据北京市公安局公安交通管理局网站道路情况说明以及 Google 查询的每个道路保养状况而估计出的。该组数据根据我们所在城市类似道路的无阻碍速度、城市道路的限速，以及咨询了我们在北京的同学得出。此时自由流情况下出行时间可由公式$\alpha = \frac{L}{V_f}$直接得出。车速 V 是根据实时路况信息中的显示而得出，前文已经说明，即：红色表示路段车辆严重拥堵，时速在 15km 以下，黄色表示路段车辆行驶缓慢，时速大于 15km 小于 40km，绿色表示路段车辆行驶畅通，时速 40 公里以上。通过街道的时间可以表示为$t = \frac{L}{V}$。根据北京市公安局公安交通管理局对于街道平均车速的记录以及安全驾驶时的跟车距离限制（如表 3 所示），我们可以衡量出街道的流量，进而得出延时参数β。

车速（km/h）	40	50	12	30	25	35	25
车距(m)	25.1	32.8	8	14.8	12	20.8	12

表 3、交管局规定的安全车距

得到道路的具体参数之后，我们可以用 MATLAB 进行编程仿真。编程代码参见 4.2 节附录。

首先应该假设整个路网中的总流量 Q。考虑到计算时间的复杂性以及避免失去真实性这对矛盾，再分析了实际的数据之后我们选取总流量为 4000。车辆的起始地点为图 8 中的 A，终止点为图 8 中的 F。从图中可以看出一共有三条路径：

Path1:A----->B----->D----->F
Path2:A----->C----->E----->F
Path3:A----->C----->D----->F

1.根据 Wardrop 的用户均衡原理，所有路径的出行时间达到相等时的状态即达到了现实中的均衡。根据 MATLAB 的仿真结果可以得出：达到均衡时所有车辆出行时间总和为 722762。

2.从总体最优角度或者古典功利主义的角度出发，当所有车辆的出行时间达到所有情况的最小值时即为该段路网的最优解。根据 MATLAB 的仿真结果得出：

达到系统最优时所有车辆出行时间总和为 722505。

3.由此可知该路网在实际使用情况中并未达到最高的效率。根据我们所查，图表 2 中的 7 条街道是同时修建的，我们不能完全依照 Braess 悖论的情况来解释。但是我们可以根据最优解中的计算数据得出其中哪些道路的利用率最低，甚

至是带来负面了的影响。

根据最优解模拟的结果，三条路径的通行流量分别为：

	Path1	Path2	Path3
流量	1608	2380	12

表 4 古典功利最优解时流量分布

从表 4 中我们可以看出达到古典功利最优情况时街道 7 中的车流量只有 12，几乎可以忽略不计。因此我们有足够的理由相信是因为街道 7 途径的存在导致了整体路网利用率的下降。

4. 下面我们将街道 7 从路网中删除，然后再仿真计算此时路网中达到用户均衡解。得出实际均衡时所有车辆出行时间总和为 722749，要比删除 7 街道之前路网的实际使用效率更高。

因此，通过上面的仿真结果，我们我可以看出北京市二环路内的该路网中确实存在因为 Braess 悖论而产生的交通运输效率下降甚至造成拥堵的情况。

3.1.5 模型的优缺点：

3.1.5.1 模型优点

（1）模型采用 MATLAB 编程，很好的模拟了二环路内某一时刻路网中各街道的使用情况以及车流量的分配情况。
（2）经过合理推断假设，可以解释造成拥堵的问题街道，并且提出了可能的改善措施。
（3）该模型提出了路网中的 Pareto 最优解，可以应用于交通分配策略，实现道路资源利用率的最大化。

3.1.5.2 模型的缺点

（1）对于路径花费时间的建模不够准确，如果道路的情况极为复杂，采用简单的线性关系将会产生较大的误差，限制了本模型的应用范围。
（2）未能考虑经过该路网车辆造成的影响。

3.1.6 模型的推广

可以将本模型中的拓扑结构复杂化，使其能够普遍适用于真是情况中的各种道路模型。另外，如果能够获得充分多的数据，可以分别分析不同天气情况下道路资源的使用情况及车流量分配，对于交通管理和城市规划建设都有重要的意义。

3.2 问题 2 的模型建立与求解

选取一个“日”字型的道路网络模型，通过 C++编程模拟司机在配备 GPS 导航系统和没有 GPS 导航系统两种情况下对行驶路线的决策。通过对系统稳定状态下每辆车通过该系统的平均时间的比较，我们可以看到，在装备 GPS 的系统中，司机通过系统的时间有明显的减少。

3.2.1 附加假设

（1） 在没有 GPS 系统的情况下，司机会在进入系统前对道路系统中的情况进行调查，根据行驶时间期望最小的原则选取最佳路线，并且在行驶的过程中一直保持这个路线。

（2） 在有 GPS 系统的情况下，司机在每一个分岔口都会进行路线的调查，根据行驶时间期望最小的原则选取最佳的路线。

3.2.1 模型建立：

如前所述，一个“日”字型的道路网络模型可以表示成以下的拓扑形式：

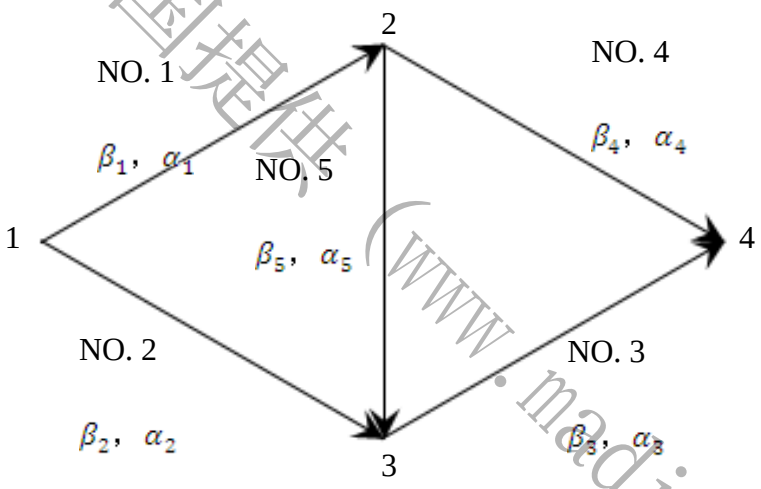

图 9 道路模型拓扑结构

图中，1,2,3,4 分别表示道路系统的 4 个节点，其中 1 为起点，4 为终点。

3.2.1.1 没有配备 GPS 导航系统

在没有配备 GPS 系统的情况下，司机在进入道路系统前（1 号节点）会进行调查，确定其在系统的中的行驶路线。此时可供司机选择的路线有 3 条：1-2-4、1-2-3-4 和 1-3-4。其行驶时间期望分别为：

$$\begin{cases} t^{1-2-4} = t^1 + t^4 = \beta_1 f_1 + \alpha_1 + \beta_4 f_4 + \alpha_4 \\ t^{1-2-3-4} = t^1 + t^5 + t^3 = \beta_1 f_1 + \alpha_1 + \beta_5 f_5 + \alpha_5 + \beta_3 f_3 + \alpha_3 \\ t^{1-3-4} = t^2 + t^3 = \beta_2 f_2 + \alpha_2 + \beta_3 f_3 + \alpha_3 \end{cases}$$

当$t^{1-2-3-4} < t^{1-2-4}$且$t^{1-2-4} < t^{1-3-4}$时，司机将会选取路线 1-2-4；当$t^{1-2-3-4} < t^{1-2-4}$且$t^{1-2-3-4} < t^{1-3-4}$时，司机将会选取路线 1-2-3-4；当

$t^{1-3-4} < t^{1-2-4}$ 且$t^{1-3-4} < t^{1-2-3-4}$时，司机将会选取路线 1-3-4。

3.2.1.2 配备 GPS 导航系统

在配备 GPS 系统的情况下，司机在进入道路系统前（1 号节点）和道路分岔处（2 号节点）会进行调查，确定其下一步的行驶方向。在 1 号节点可供司机选择的路线有 3 条：1-2-4、1-2-3-4 和 1-3-4。其行驶时间期望分别为：

$$\begin{cases} t^{1-2-4} = t^1 + t^4 = \beta_1 f_1 + \alpha_1 + \beta_4 f_4 + \alpha_4 \\ t^{1-2-3-4} = t^1 + t^5 + t^3 = \beta_1 f_1 + \alpha_1 + \beta_5 f_5 + \alpha_5 + \beta_3 f_3 + \alpha_3 \\ t^{1-3-4} = t^2 + t^3 = \beta_2 f_2 + \alpha_2 + \beta_3 f_3 + \alpha_3 \end{cases}$$

当$t^{1-2-4} < t^{1-2-3-4}$ 且$t^{1-2-4} < t^{1-3-4}$时，司机将会选择路线 1-2-4，进入 1 号公路；当$t^{1-2-3-4} < t^{1-2-4}$且$t^{1-2-3-4} < t^{1-3-4}$时，司机将会选择线路 1-2-3-4 同样会进入 1 号公路；当$t^{1-3-4} < t^{1-2-4}$ 且$t^{1-3-4} < t^{1-2-3-4}$时，司机将会选择路线 1-3-4 进入 2 号公路。

在 2 号节点可供司机选择的路线有 2 条：2-4 和 2-3-4。其行驶时间期望分别为：

$$\begin{cases} t^{2-4} = t^4 = \beta_4 f_4 + \alpha_4 \\ t^{2-3-4} = t^5 + t^3 = \beta_5 f_5 + \alpha_5 + \beta_3 f_3 + \alpha_3 \end{cases}$$

当$t^{2-4} < t^{2-3-4}$时，在 1 号节点选择路线 1-2-3-4 的司机将会改变其原来的行驶方案，选择路线 2-4，而原先选择路线 1-2-4 的司机将不会改变其原来的行驶方案；当$t^{2-4} > t^{2-3-4}$时，在 1 号节点选择路线 1-2-4 的司机将会改变其原来的行驶方案，选择路线 2-3-4，而原先选择 1-2-3-4 的司机将不会改变其行驶方案。

3.2.2 模型检验

用 C++进行仿真，取入口处的车流量为 30 辆/秒。道路网络的其他参数取值如下：

道路名称	β	α
道路 1	0.0245	25.689
道路 2	0.0292	34.4559
道路 3	0.0433	21.3134

道路 4	0.0315	42.6566
道路 5	0.0078	13.981

表 5 道路网络参数表[4]

当进单位时间内进入系统的车辆数与离开系统的车辆数大致相等时，认为当前系统处于相对稳定的状态。对此时系统中车辆数据进行统计，计算出每辆车在系统中的平均时间进行比较。平均时间t_{av}的计算公式如下：

$$t_{av} = \frac{\sum_{i=1}^{5} \beta_i f_i + \alpha_i}{Q}$$

用 C++进行仿真得到输出如下：

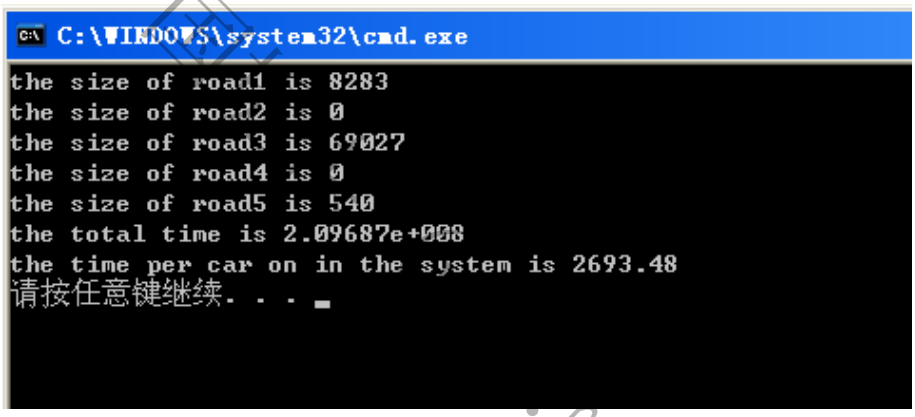

图 10 没有配备 GPS 系统的仿真输出

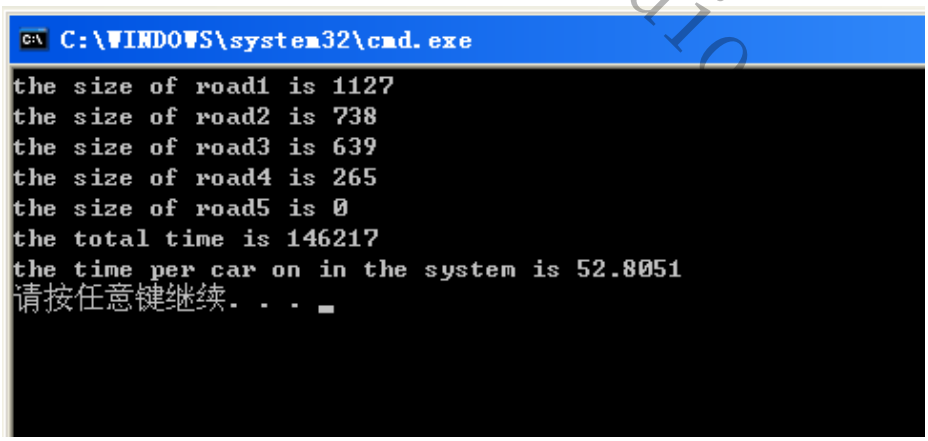

图 11 配备有 GPS 系统的仿真输出

通过仿真比较，可以看到使用了 GPS 系统以后每辆车在系统中所花费的时间明显减少，从而证明了 GPS 系统可以有效地减少道路的拥堵状况。

[4] 董菁，张佐 非合作交通网络中的 Braess 悖论及其避免

3.2.3 模型的优缺点：

3.2.3.1 模型优点

（1）本模型较为真实地体现了实际中道路的使用情况，真实地模拟了实际驾驶中司机所做的决策。

（2）本模型可以使用 C++仿真实现，将参数进行量化计算，使结论更加直观。

（3）本模型在交通部门对道路使用情况的研究和城市规划部门规划城市道路的规划有较大的利用价值。

3.2.3.2 模型的缺点

（1）模型所取的道路模型相对简单，没有体现更加复杂的道路网络情况。

（2）没有体现不同车辆对道路交通影响。

3.2.4 模型的推广

对本模型中的道路系统的拓扑结构加以复杂化，并且考虑不同道路中不同车辆的不同参数特性，可以使本模型更加精确地体现一般道路的交通情况，从而更加准确地体现 GPS 导航系统的使用对道路拥堵情况的改善。通过根据不同地段的具体情况对模型中具体参数进行修正，可以将本模型运用于交通研究，城市规划等领域中

4、附录

4.1 参考文献

[1]Wardrop J G. Some theoretical aspects of road traffic research.[A]. In: Proceedings of the Institution of Civil Engineers Ⅱ(1) [C]. 1952. 325-378.
[2] Arnott, R. and Small, K. (1994) The economics of traffic congestion. *American Scientist 82, 446-455.*
[3]百度百科
[4] 董菁, 张佐 非合作交通网络中的 Braess 悖论及其避免 JOURNAL OF HIGHWAYAND TRANSPORTATION RESEARCH AND DEVELOPMENT, 100200268 (2004) 0500092004
[5] 北京市公安局公安交通管理局网站 http://www.bjjtgl.gov.cn/

4.2 问题 1 程序代码

1、用户均衡解代码：

```
function [f1,f2,f3,t,T]=bal
Q=4000;

a=[22.4425,20.1881,13.9895,28.0292,18.8965,20.4864,23.8855];
b=[0.015012144,0.011113896,0.048636833,0.016914119,0.014403708,0.017079542,0.02167355];

syms x1 x2 x3 t

f1=a(1)+b(1)*x1+a(2)+b(2)*x1+a(3)+b(3)*(x1+x3)-t;
f2=a(6)+b(6)*(x2+x3)+a(5)+b(5)*x2+a(4)+b(4)*x2-t;
f3=a(6)+b(6)*(x2+x3)+a(7)+b(7)*x3+a(3)+b(3)*(x1+x3)-t;
f4=x1+x2+x3-Q;

S=solve(f1,f2,f3,f4);
f1=S.x1;f2=S.x2;f3=S.x3;t=S.t;
T=Q*t;
```

2、帕累托最优解代码：

```
function [x1,x2,x3,t1,t2,t3,Tmin]=optimal

a=[22.4425,20.1881,13.9895,28.0292,18.8965,20.4864,23.8855];
b=[0.015012144,0.011113896,0.048636833,0.016914119,0.014403708,0.017079542,0.02167355];

q=4000;
```

```
x1=0;x2=0;x3=q;
Tmin=(a(6)+b(6)*q+a(7)+b(7)*q+a(3)+b(3)*q)*q;

for i=0:q
    for j=0:q-i
        t1=a(1)+b(1)*i+a(2)+b(2)*i+a(3)+b(3)*(q-j);
        t2=a(6)+b(6)*(q-i)+a(5)+b(5)*j+a(4)+b(4)*j;
        t3=a(6)+b(6)*(q-i)+a(7)+b(7)*(q-i-j)+a(3)+b(3)*(q-j);
        temp=t1*i+t2*j+t3*(q-i-j);
        if (temp<=Tmin)
            x1=i;x2=j;x3=q-i-j;
            Tmin=temp;
        end
    end
end

t1=a(1)+b(1)*x1+a(2)+b(2)*x1+a(3)+b(3)*(x1+x3);
t2=a(6)+b(6)*(x2+x3)+a(5)+b(5)*x2+a(4)+b(4)*x2;
t3=a(6)+b(6)*(x2+x3)+a(7)+b(7)*x3+a(3)+b(3)*(x1+x3);
```

4.2 问题 2 程序代码

1、没有 GPS 导航系统的情况：

```
#include<iostream>
#include"road.h"
#include<cmath>

using namespace std;

int main()
{
    int number=4000;
    int innumber=30;
    int complete=0;
    int etime=0;
    double sum=0;
    double ave=0;
    road* way0;
    road* way1;
    road* way2;
    road* way3;
    road* way4;
    road* end;
    way0=new road;
```

```
way1=new road;
way2=new road;
way3=new road;
way4=new road;
end=new road;
way0->number=1;
way1->number=2;
way2->number=3;
way3->number=4;
way4->number=5;
for(int i=1;i<=number;i++)
{
    car* temp;
    temp=new car;
    double time1=0;
    double time2=0;
    double temptime1;
    double temptime2;
    temptime1=way0->size*0.0245+25.689+way3->size*0.0315+42.6566;

temptime2=way0->size*0.0245+25.689+way4->size*0.0078+13.9831+way2->size*0.
0433+21.3134;
    if(temptime1<=temptime2)
    {
        time1=temptime1;
        temp->routine=1;
    }
    else
    {
        time1=temptime2;
        temp->routine=2;
    }
    time2=way1->size*0.0292+34.4559;
    temp->no=i;
    if(time1<=time2)
    {
        way0->roadenter(temp);
    }
    else
    {
        way1->roadenter(temp);
        temp->routine=3;
    }
```

```
    }
    while(complete!=1)
    {
        double time1=0;
        double time2=0;
        road* way3c=way3;
        road* way4c=way4;
        road* way0c=way0;
        road* way1c=way1;
        road* way2c=way2;
        int oldsize=end->size;

        for(int i=0; i<innumber;i++)
        {
            car* temp;
            temp=new car;
            double temptime1;
            double temptime2;
            temptime1=way0->size*0.0245+25.689+way3->size*0.0315+42.6566;

    temptime2=way0->size*0.0245+25.689+way4->size*0.0078+13.9831+way2->size*0.
0433+21.3134;
            if(temptime1<=temptime2)
            {
                time1=temptime1;
                temp->routine=1;
            }
            else
            {
                time1=temptime2;
                temp->routine=2;
            }
            time2=way1->size*0.0292+34.4559+way2->size*0.0433+21.3134;
            temp->no=i;
            if(time1<=time2)
            {
                way0->roadenter(temp);
            }
            else
            {
                way1->roadenter(temp);
                temp->routine=3;
            }
        }
```

```
        car* out0;
        out0=way0c->roadtime();
        if(out0!=NULL)
        {
            car* temp;
            car* postemp;
            temp=out0;
            while(temp!=NULL)
            {
                postemp=temp->next;
                temp->next=NULL;
                if(temp->routine==1)
                {
                    way3->roadenter(temp);
                }
                else
                {
                    way4->roadenter(temp);
                }
                temp=postemp;
            }

        }
        way2->roadenter(way1c->roadtime());
        way2->roadenter(way4c->roadtime());
        end->roadenter(way3c->roadtime());
        end->roadenter(way2c->roadtime());

        if(abs(end->size-oldsize)<innumber+6 && abs(end->size-oldsize)>innumber-6)
        {
            complete=1;
        }
        else
        {
            etime=0;
        }
        if(complete==1 && etime<10)
        {
            etime++;
            complete=0;
        }
    }
    way0->print();
    way1->print();
```

```
        way2->print();
        way3->print();
        way4->print();
sum=way0->size*(way0->size*0.0245+25.689)+(way1->size*0.0292+34.4559)*way1->siz
e+(way2->size*0.0433+21.3134)*way2->size+(way3->size*0.0315+42.6566)*way3->size
+(way4->size*0.0078+13.9831)*way4->size;
        ave=sum/(way0->size+way1->size+way2->size+way3->size+way4->size);
        cout<<"the total time is "<<sum<<"\n";
        cout<<"the time per car on in the system is "<<ave<<"\n";
        return 0;
}
```

2、有 GPS 导航系统的情况：

```
#include<iostream>
#include"road.h"
#include<cmath>
using namespace std;
int main()
{
        int number=4000;
        int innumber=30;
        int complete=0;
        int etime=0;
        double sum=0;
        double ave=0;
        road* way0;
        road* way1;
        road* way2;
        road* way3;
        road* way4;
        road* end;
        way0=new road;
        way1=new road;
        way2=new road;
        way3=new road;
        way4=new road;
        end=new road;
        way0->number=1;
        way1->number=2;
        way2->number=3;
        way3->number=4;
        way4->number=5;
        for(int i=1;i<=number;i++)
        {
                car* temp;
```

```
        double time1=0;
        double time2=0;
        double temptime1;
        double temptime2;
        temptime1=way0->size*0.0245+25.689+way3->size*0.0315+42.6566;
    temptime2=way0->size*0.0245+25.689+way4->size*0.0078+13.9831+way2->size*0.
0433+21.3134;
        if(temptime1<=temptime2)
        {
            time1=temptime1;
        }
        else
        {
            time1=temptime2;
        }
        time2=way1->size*0.0292+34.4559;
        temp=new car;
        temp->no=i;
        if(time1<=time2)
        {
            way0->roadenter(temp);
        }
        else
        {
            way1->roadenter(temp);
        }
    }
    while(complete!=1)
    {
        double time1=0;
        double time2=0;
        road* way3c=way3;
        road* way4c=way4;
        road* way0c=way0;
        road* way1c=way1;
        road* way2c=way2;
        int oldsize=end->size;
        for(int i=0; i<innumber;i++)
        {
            car* temp;
            double temptime1;
            double temptime2;
            temptime1=way0->size*0.0245+25.689+way3->size*0.0315+42.6566;
    temptime2=way0->size*0.0245+25.689+way4->size*0.0078+13.9831+way2->size*0.
```

```
0433+21.3134;
            if(temptime1<=temptime2)
            {
                time1=temptime1;
            }
            else
            {
                time1=temptime2;
            }
            time2=way1->size*0.0292+34.4559+way2->size*0.0433+21.3134;
            temp=new car;
            temp->no=i;
            if(time1<=time2)
            {
                way0->roadenter(temp);
            }
            else
            {
                way1->roadenter(temp);
            }
        }
        car* out0;
        out0=way0c->roadtime();
        if(out0!=NULL)
        {
            time1= way3->size*0.0315+42.6566;
            time2= way4->size*0.0078+13.9831+way2->size*0.0433+21.3134;
            if(time1<=time2)
            {
                way3->roadenter(out0);
            }
            else
            {
                way4->roadenter(out0);
            }
        }
        way2->roadenter(way1c->roadtime());
        way2->roadenter(way4c->roadtime());
        end->roadenter(way3c->roadtime());
        end->roadenter(way2c->roadtime());
        if(abs(end->size-oldsize)<innumber+6 && abs(end->size-oldsize)>innumber-6)
        {
            complete=1;
        }
```

```
            else
            {
                etime=0;
            }
            if(complete==1 && etime<10)
            {
                etime++;
                complete=0;
            }
        }
        way0->print();
        way1->print();
        way2->print();
        way3->print();
        way4->print();

sum=way0->size*(way0->size*0.0245+25.689)+(way1->size*0.0292+34.4559)*way1->siz
e+(way2->size*0.0433+21.3134)*way2->size+(way3->size*0.0315+42.6566)*way3->size
+(way4->size*0.0078+13.9831)*way4->size;
        ave=sum/(way0->size+way1->size+way2->size+way3->size+way4->size);
        cout<<"the total time is "<<sum<<"\n";
        cout<<"the time per car on in the system is "<<ave<<"\n";
        return 0;
}
```