基于 Frank-Wolfe 算法的交通流量分配

摘　要

Frank-Wolfe 算法是均衡网络流中最为常见的路段算法，其基本思想是用线性规划逐步迭代逼近非线性规划，在每次迭代中，将目标函数线性化，通过求解相应的线性规划来获得可行下降方向，再沿此可行下降方向在可行域内进行一维搜索求解最优移动步长，进而得到下一个迭代点，重复迭代过程直至找到最优解。

该算法中探索方向的确定相当于求解一个线性规划问题，而在 UE 问题中该线性规划问题恰好等价于一个最短路问题（全有全无网络加载），无须采用一般线性规划问题的单纯形法求解，于是可大大节省计算成本。同时，在迭代过程中，只需保存路段流量，而无须保存路径流量，因而大大节省了计算机内存。

但是，该算法迭代后期的收敛速度较低，原因是当迭代解趋近于最优解时，探索方向将垂直于目标函数的梯度方向。

本文利用 Frank-Wolfe 算法对某路网进行了交通流量分配，其中求解最优步长的一维搜索利用的是黄金分割法，此外，全有全无网络加载中的最短路算法利用的是 Dijkstra 算法；最后，通过 matlab 进行了编程得到了数值结果，并对结果进行了分析。

关键词：交通分配、Frank-Wolfe 算法、全有全无网络加载、Dijkstra 算法

1 引言

给定某种出行方式的分布矩阵（origin-destination matrix），按照某种路径选择原则，将出行需求分配到交通网络中的各条道路上，得到路径流量、路段流量

和交叉口方向流量，这就是固定需求下的交通网络流问题。交通网络要素的流量及相关特性指标，是交通规划、管理和控制的主要依据。固定需求下交通网络流问题的输入包括：①出行需求的分布矩阵；②交通网络的拓扑关系；③路段特性函数。而交通网络流问题的输出包括网络要素的流量和费用。

在交通工程学和交通规划原理中，一般采用“四阶段法”预测交通需求，许多文献把固定需求下的交通网络流问题称为交通流量分配或交通分配。其实，交通网络流问题比前述的内容还要广泛，它包括交通流量分配和需求矩阵估计。本文只讨论其交通流量分配。

交通网络流问题建模的核心是网络用户的选择机理。网络用户就是使用交通网络的出行者，可能是人、机动车、公交车等。Wardrop 第一原则认为，所有出行者独立地做出令自己旅行时间最小的决策，结果形成这样的网络流状态，在相同 OD 对之间，所有使用路径的旅行时间相等并且最小，所有未被使用路径的旅行时间大于或等于使用路径的旅行时间。满足 Wardrop 第一原则的交通网络流状态，通常称为用户均衡（User Equilibrium，简称 UE）。在用户均衡状态下，没有用户能够通过单方面的路径变更行为，来减少自己的旅行时间。

1956 年 Beckmann 等人提出了 Wardrop 第一原则最优化模型（即 Beckmann 变换式），为研究均衡状态下交通网络流特性提供了解析手段，并促进了关于交通网络流问题的数值计算方法的诞生。

与用户均衡条件等价的最优化问题（即 Beckmann 变换式）是：

$$\min z(x)=\sum_a \int_0^{x_a} t_a(x)dx \qquad (a)$$

$$s.t.\begin{cases} \sum_k f_k^w = q^w \quad \forall w \in W & (b) \\ f_k^w \geq 0 \quad \forall k \in K^w, \forall w \in W & (c) \\ x_a = \sum_w \sum_k f_k^w \delta_{a,k}^w \quad \forall a \in A & (d) \end{cases}$$

符号说明：

f_k^w——OD 对 w 之间第 k 条路径的路径流量；

q^w——OD 对 w 之间的分布交通量；

K^w——OD 对 w 之间所有路径的集合，这些路径可能有流量，亦可能无流量；

W——交通网络所有 OD 对的集合；

A——交通网络所有路段的集合；

x_a——路段 a 的路段流量；

$t_a(x_a)$——路段 a 的路段旅行时间函数；

$\delta_{a,k}^w$——连接关系变量。当路径 k 包含路段 a 时，$\delta_{a,k}^w$=1；否则 $\delta_{a,k}^w$=0。

模型的目标函数是对各路段阻抗函数的积分之和，模型的最优解使得目标函数最小化，对于目标函数很难作出直观的经济学解释，一般认为它只是一种求解均衡问题的数学手段；约束式（b）是路径流量与 OD 分布量之间的守恒关系，式（c）是路径流量的非负约束条件，式（d）是路径流量和路段流量之间的关系。

需要说明的是，模型假设路段旅行时间仅仅是其自身路段流量的函数，而与其他路段的流量无关；其次假设路段旅行时间是路段流量的严格增函数，即拥挤效应。

2 正文

1 问题描述

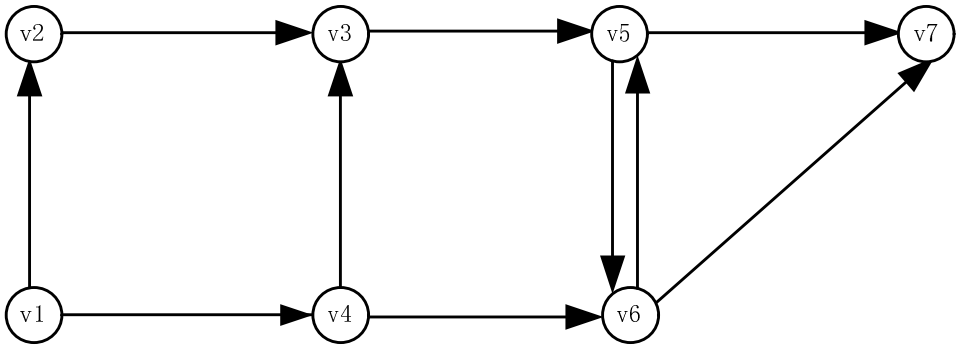


图 1

如图 1 所示，有向图 G(N,A)表示一个交通网络，$N=\{v_i\}$ 为网络中结点的集合，$A=\{(v_i,v_j)\}$ 为路段的集合。

出行的出发地和目的地，分别叫做 O、D 点，为方便研究，这里只取 v_1 为 O 点，其余各点为 D 点。$W=\{w\}$ 表示交通网络所有 OD 对的集合，在这里，W={(1,2),(1,3),(1,4),(1,5),(1,6),(1,7)}。

已知由美国道路局（BPR-Bureau of Public Road）开发的函数，被称为 BPR 函数，形式为：

$$t_a(x_a)=t_a(0)\cdot\left[1+\alpha\left(\frac{x_a}{c_a}\right)^{\beta}\right]$$

式中：$a\in A$ 为路段，x_a 表示路段 a 的交通流量；

c_a ——路段 a 的交通容量，即单位时间里可通过的最大车辆数；

$t_a(0)$ ——道路 a 上的车辆平均自由走行时间；

α、β ——待标定的参数，BPR 建议取：α=0.15，β=4，也可由实际数据用回归分析求得。

对于参数 c_a、$t_a(0)$，可由道路 a 的实测数据分析得到。

则针对图 1 的路网，可先假设各路段特性函数分别为：

$$\begin{cases} t_{12}=5\left[1+0.15\left(x_{12}/28\right)^4\right] \\ t_{14}=4\left[1+0.15\left(x_{14}/26\right)^4\right] \\ t_{23}=5\left[1+0.15\left(x_{23}/35\right)^4\right] \\ t_{43}=6\left[1+0.15\left(x_{43}/25\right)^4\right] \\ t_{35}=7\left[1+0.15\left(x_{35}/18\right)^4\right] \\ t_{46}=5\left[1+0.15\left(x_{46}/32\right)^4\right] \\ t_{56}=8\left[1+0.15\left(x_{56}/27\right)^4\right] \\ t_{57}=5\left[1+0.15\left(x_{57}/23\right)^4\right] \\ t_{65}=7\left[1+0.15\left(x_{65}/22\right)^4\right] \\ t_{67}=5\left[1+0.15\left(x_{67}/26\right)^4\right] \end{cases}$$

并假设各 OD 对之间的分布交通流量分别为：

$q^{(1,2)}$=20， $q^{(1,3)}$=25， $q^{(1,4)}$=18， $q^{(1,5)}$=26， $q^{(1,6)}$=28， $q^{(1,7)}$=30。

建立该路网的 Beckmann 变换式模型：

$$\min z(x)=\sum_a\int_0^{x_a}t_a(x)dx \qquad (a)$$

$$s.t.\begin{cases} \sum_k f_k^w=q^w \quad \forall w\in W & (b) \\ f_k^w\geq 0 \quad \forall k\in K^w,\forall w\in W & (c) \\ x_a=\sum_w\sum_k f_k^w\delta_{a,k}^w \quad \forall a\in A & (d) \end{cases}$$

2 Frank-Wolfe 算法

2.1 一般问题的 Frank-Wolfe 算法

Frank 和 Wolfe 于 1956 年提出了一种用于求解含线性约束的非线性规划问题的线性化算法，通常称为 Frank-Wolfe 算法。该方法属于可行方向法的一种。可行方向法的基本思想是从某可行点开始，沿着令目标函数下降的方向进行搜索，求出新的可行迭代点，反复迭代直至找到最优解。算法主要包括两个步骤：确定探索方向和确定移动步长，其中前者的选择方式不同就形成了不同的可行方向法。

考虑如下含线性约束条件的非线性规划问题：

$$\begin{aligned} &\min Z(x) && (a) \\ &s.t.\ Ax=b && (b) \\ &x\geq 0 && (c) \end{aligned} \qquad (2.1.1)$$

式中，A 是 $m\times n$ 的系数矩阵，b 是 m 维常数列向量，x 是 n 维的列向量，Z(x)

是连续可微函数，记可行域为 $D=\left\{x\middle|Ax=b,x\geq 0\right\}$。

Frank-Wolfe 算法的基本思想是用线性规划逐步迭代逼近非线性规划，在每次迭代中，将目标函数 Z(x)线性化，通过求解相应的线性规划获得可行下降方向，再沿此可行下降方向在可行域内进行一维搜索求解当次迭代的最优移动步长，从而得到下一个迭代点，重复迭代直至找到最优解。算法具体过程分析如下：

对 Z(x)在第 n 次迭代点 x^n 上进行一阶泰勒展开，得到 Z(x)的近似线性函数 $Z_L^n\left(x\right)$，即：

$$\begin{aligned} Z_L^n(x) &= Z\left(x^n\right)+\nabla Z\left(x^n\right)^T\cdot\left(x-x^n\right) \\ &= \left[Z(x^n)-\nabla Z(x^n)^T x^n\right]+\nabla Z(x^n)^T x \end{aligned} \tag{2.1.2}$$

将式（2.1.2）代入（2.1.1）中，则在 x^n 的某个领域内，式（2.1.1）可近似等价于为下面这个线性规划问题：

$$\begin{aligned} &\min\left[Z(x^n)-\nabla Z\left(x^n\right)^T x^n\right]+\nabla Z(x^n)^T x \quad (a) \\ &s.t.\ x\in D \qquad\qquad (b) \end{aligned} \tag{2.1.3}$$

又由于 $Z\left(x^n\right)-\nabla Z(x^n)^T x^n$ 为常数，所以式（2.1.3）又等价于

$$\begin{aligned} &\min\nabla Z(x^n)^T\cdot x \qquad (a) \\ &s.t.\ x\in D \qquad\qquad (b) \end{aligned} \tag{2.1.4}$$

式（2.1.4）是一线性规划问题，有很多经典的算法（如单纯形法）可求解该模型，其结果必为以下两种情形之一：

①若 $\nabla Z(x^n)^T\cdot\left(y^n-x^n\right)=0$，则停止迭代，此时的 x^n 就是原问题的 K-T 点；

②若 $\nabla Z(x^n)^T\cdot\left(y^n-x^n\right)\neq 0$，又由于式（2.1.4）可知 $\nabla Z(x^n)^T\cdot y^n-\nabla Z(x^n)^T\cdot x^n\leq 0$，所以必有 $\nabla Z(x^n)^T\cdot\left(y^n-x^n\right)<0$，因此 (y^n-x^n) 为 x^n 处的可行下降方向。

从本次迭代点 x^n 开始，沿可行下降方向 (y^n-x^n) 进行一维搜索：

$$\min_{0\leq\lambda\leq 1}\mathrm{Z}\left[x^n+\lambda\left(y^n-x^n\right)\right] \tag{2.1.5}$$

得到最优移动步长 λ_n，更新迭代点：

$$x^{n+1}=x^n+\lambda_n\left(y^n-x^n\right) \tag{2.1.6}$$

由于 $(y^n-x^n)\neq 0$，且为下降方向，所以 $Z\left(x^{n+1}\right)<Z\left(x^n\right)$。得到 x^{n+1} 后，反复迭代直至满足某收敛准则。

式（2.1.6）也可改写成 $x^{n+1}=(1-\lambda_n)x^n+\lambda_n y^n$，因此新可行解 x^{n+1} 实际上是 x^n 与 y^n 的凸组合，故 Frank-Wolfe 算法又被称为凸组合法。

2.2 UE 问题的 Frank-Wolfe 算法

Beckmann 等人在 1956 年提出的 UE 问题的变换式（即 Beckmann 变换式）一直没有有效的求解算法，直到 1975 年，Leblanc 等将 Frank-Wolfe 算法用于求

解 Beckmann 变换式，形成了目前广泛使用的均衡交通分配求解算法。

Frank-Wolfe 算法中探索方向的确定相当于求解一个线性规划问题，这在理论上很容易实现，如采用单纯形算法求解，但对于大型交通网络，算法就因计算量庞大而不实用。但通过下面的分析可以看出，由于交通分配问题的特殊性，上述线性规化问题实际上等价于一个最短路问题（全有全无网络加载），从而大大节省了计算时间。下面具体介绍 UE 问题的 Frank-Wolfe 算法。

在第 n 次迭代中，目标函数 Z(x)对路段流量的梯度是 $\partial Z(x^n)/\partial x_a = t_a^n$，可行下降方向可由求解下面的线性规划得到：

$$\begin{aligned} &\min Z^n(y) = \left[\nabla Z(x^n)\right]^T \cdot y = \sum_a \left[\partial Z(x^n)/\partial x_a\right] y_a = \sum_a t_a^n y_a \quad (a) \\ &s.t.\ \sum_k g_k^w = q^w, \quad \forall w \qquad (b) \\ &g_k^w \geq 0, \quad \forall k, w \qquad (c) \\ &y_a = \sum_w \sum_k g_k^w \delta_{ak}^w, \quad \forall a \qquad (\mathrm{d}) \end{aligned} \tag{2.2.1}$$

式中，y_a 是辅助路段流量，g_k^w 是辅助路径流量。t_a^n 是已知数，y_a 是所求的未知数。上式实际上是当路段旅行时间固定为 t_a^n 时令网络总旅行时间最小的经典运输问题。显然，只要将每个 OD 对之间的出行需求全部分配到各 OD 对之间的最短路径上，即对所有 OD 对执行一次全有全无网络加载，即可求得 y_a，进而得到可行下降方向 $y^n - x^n$，而不必像一般的线性规划问题那样用单纯形法经过繁杂的求解步骤，这正是 UE 问题的 Frank-Wolfe 算法的特殊之处。

Frank-Wolfe 算法的另外一个主要步骤是确定最优移动步长，可由求解下面的一维极小化问题得到：

$$\min_{0\leq\lambda\leq 1} Z\left[x^n + \lambda\left(y^n - x^n\right)\right] = \sum_a \int_0^{x_a^n + \lambda(y_a^n - x_a^n)} t_a(w)dw \tag{2.2.2}$$

该极小化问题可用许多一维搜索方法求得。

UE 问题的 Frank-Wolfe 算法步骤可归纳如下：

Step 0：初始化。令路段旅行时间为 $t_a^0 = t_a(0), \forall a$，在此基础上执行一次全有全无网络加载，得到路段流量 $\{x_a^1\}$，令 n=1；

Step 1：更新路段旅行时间。计算 $t_a^n = t_a\left(x_a^n\right), \forall a$；

Step 2：确定下降方向。在 $\{t_a^n\}$ 的基础上，执行一次全有全无网络加载，得到一组附加路段流量 $\{y_a^n\}$；

Step 3：确定迭代步长。求解一维搜索问题式（2.2.2），得到最优移动步长 λ_n；

Step 4：移动。令 $x_a^{n+1} = x_a^n + \lambda_n\left(y_a^n - x_a^n\right), \forall a$；

Step 5：收敛性检验。若满足收敛准则，算法终止，此时的 $\{x_a^{n+1}\}$ 即是均衡路段流量；否则令 n=n+1，转入到 Step1。

算法收敛准则可根据目标函数值的下降情况确定，也可以根据前后两次迭代的路段流量的变化情况而定，如当

$$\sqrt{\sum_a \left(x_a^{n+1} - x_a^n\right)^2} \Big/ \sum_a x_a^n \le \varepsilon$$

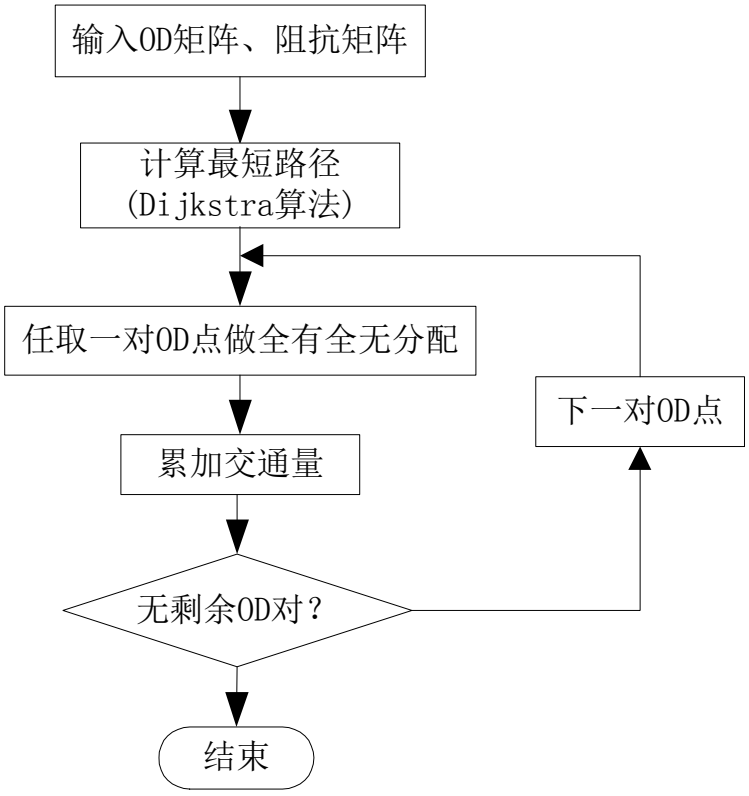
成立时，迭代终止，其中ε是预先设定的精度参数。

通过分析算法步骤可以发现，在该算法迭代过程中，只需要保存路段流量，而无须保存路径流量，这样大大节省了计算机内存，此算法可用在大型实际交通网络中，目前许多交通规划商业软件如TransCAD等都采用该算法进行交通分配。该算法的缺陷是迭代过程后期的收敛速度较低，主要原因是当迭代解趋近于最优解时，探索方向将垂直于目标函数的梯度方向。

当然，影响算法收敛速度的主要原因还有：初始解、网络结构和拥挤程度。初始解离均衡解越近，需要的迭代次数就越少；网络结构越复杂，或者说从起点到终点的可行路径越多，需要的迭代次数也就越多；在相对不怎么拥挤的网络中，每次更新路段流量并不显著改变路段旅行时间，且最短路径基本保持不变，所以只需要少数几次迭代就可达到 UE 状态，但随着网络拥挤程度的上升，达到 UE 状态所需要的迭代次数将显著增加。在实际大型交通网络中，一般只需要 4-6 次迭代过程就可以达到工程应用所需要的精度要求。

2.3 全有全无网络分配法和 Dijkstra 最短路径算法

基于 Dijkstra 算法的全有全无分配法算法流程图如下：



Dijkstra 算法是最为常用的最短路径探索算法，它由 Dijkstra 在 1959 年提出，可探索从起始点到网络中任一节点的最短路径。

考虑交通网络 G=(N,A)，其中 $N=\{v_i\}$ 为网络中结点的集合，$A=\{(v_i,v_j)\}$ 为路段的集合，t_{ij} 为路段 (v_i,v_j) 的费用（或阻抗）。

Dijkstra 算法的基本思想是从起始结点 v_s 开始，为每一个结点记录下一个数（称之为标号），标号分 T 标号和 P 标号两种。T 标号表示从 v_s 到该结点的最短路权的上界，称为临时标号；P 标号表示从 v_s 到该结点的最短路权，称为固定标号。算法的每一步就是去修改 T 标号，并且把某一个点的 T 标号修改为 P 标号。经过有限步后，所有的结点都可获得 P 标号，这样就得到了从起始结点 v_s 到任意结点的最短路径费用。

下面定义算法使用的一些符号和变量。用 P、T 分别表示某结点的 P 标号和 T 标号，S_i 表示第 i 步时，具有 P 标号的结点的集合。用 $c(v_s,v)$ 表示从起始节点 v_s 到结点 v 的最短路径费用。为了在求出最短路径费用的同时，也获得最短路径信息（即最短路径所包含的路段），给每个结点以一个 λ 值，算法终止时，若 $\lambda(v)=m$，表示在从 v_s 到 v 的最短路径上，v 的前一个结点是 v_m；若 $\lambda(v)=M$，M 为初始化时设置的一个非常大的常数，则表示网络 G 中不包含从 v_s 到 v 的路径，即 v_s 和 v 是不连通的；若 $\lambda(v)=0$，表示 $v=v_s$。

Dijkstra 算法的具体步骤如下：

Step 0：初始化。令 $i=0,S_0=\{v_s\},P(v_s)=0,\lambda(v_s)=0$，对 $\forall v\neq v_s$，令 $T(v)=\infty$，$\lambda(v)=M$，令当前检查点的标号 $k=s$；

Step 1：终止检验。若 $S_i=N$，算法终止，此时，$c(v_s,v)=P(v),\forall v\in S_i=N$；否则转入 Step 2；

Step 2：修改 T 标号。对每个使 $(v_k,v_j)\in A$ 且 $v_j\notin S_i$ 的结点 v_j，若 $T(v_j)>P(v_k)+t_{kj}$，则修改结点 v_j 的 T 标号，令 $T(v_j)=P(v_k)+t_{kj},\lambda(v_j)=k$；否则转入 Step 3；

Step 3：确定 P 标号。令 $T(v_{j_i})=\min\limits_{v_j\notin S_i}\{T(v_j)\}$。若 $T(v_{j_i})<+\infty$，则把 v_{j_i} 的 T 标号置为 P 标号，即 $P(v_{j_i})=T(v_{j_i})$，同时令 $S_{i+1}=S_i\cup\{v_{j_i}\},k=j_i$。然后令 $i=i+1$，转入 Step 1；否则，算法终止，此时对每一个结点 $v\in S_i$，有 $c(v_s,v)=P(v)$，而对每一个 $v\notin S_i,c(v_s,v)=T(v)$。

算法终止后，可通过 λ 值“反向追踪”到从起始结点 v_s 到任意结点 v 的最短路径。

3 数值结果及分析

取收敛的精度参数 $\varepsilon=10^{-2}$，用 matlab 编程，得到 Frank-Wolfe 算法迭代计算过程如下：

注：(v_i,v_j) 表示 v_i 到 v_j 的路段

表 3.1

迭代次数	算法步骤	(v_1,v_2)	(v_1,v_4)	(v_2,v_3)	(v_3,v_5)	(v_4,v_3)
0	初始化	$t^0_{(1,2)}$=5	$t^0_{(1,4)}$=4	$t^0_{(2,3)}$=5	$t^0_{(3,5)}$=7	$t^0_{(4,3)}$=6
		$x^1_{(1,2)}$=20	$x^1_{(1,4)}$=127	$x^1_{(2,3)}$=0	$x^1_{(3,5)}$=0	$x^1_{(4,3)}$=25
1	更新	$t^1_{(1,2)}$=5.2	$t^1_{(1,4)}$ =345.56	$t^1_{(2,3)}$=5	$t^1_{(3,5)}$=7	$t^1_{(4,3)}$=6.9
	探索方向	$y^1_{(1,2)}$=129	$y^1_{(1,4)}$=18	$y^1_{(2,3)}$=109	$y^1_{(3,5)}$=84	$y^1_{(4,3)}$=0
	移动	$x^2_{(1,2)}$ =68.73	$x^2_{(1,4)}$ =78.27	$x^2_{(2,3)}$ =48.73	$x^2_{(3,5)}$ =37.55	$x^2_{(4,3)}$ =13.82
2	更新	$t^2_{(1,2)}$ =32.22	$t^2_{(1,4)}$=53.29	$t^2_{(2,3)}$=7.82	$t^2_{(3,5)}$ =26.89	$t^2_{(4,3)}$=6.08
	探索方向	$y^2_{(1,2)}$=71	$y^2_{(1,4)}$=76	$y^2_{(2,3)}$=51	$y^2_{(3,5)}$=26	$y^2_{(4,3)}$=0
	移动	$x^3_{(1,2)}$=71	$x^3_{(1,4)}$=76	$x^3_{(2,3)}$=51	$x^3_{(3,5)}$=26	$x^3_{(4,3)}$=0
3	更新	$t^3_{(1,2)}$ =36.01	$t^3_{(1,4)}$=47.8	$t^3_{(2,3)}$=8.38	$t^3_{(3,5)}$ =11.57	$t^3_{(4,3)}$=6
	探索方向	$y^3_{(1,2)}$=101	$y^3_{(1,4)}$=46	$y^3_{(2,3)}$=81	$y^3_{(3,5)}$=56	$y^3_{(4,3)}$=0
	移动	$x^4_{(1,2)}$=72.1	$x^4_{(1,4)}$ =74.91	$x^4_{(2,3)}$ =52.09	$x^4_{(3,5)}$ =27.09	$x^4_{(4,3)}$=0
4	更新	$t^4_{(1,2)}$ =37.95	$t^4_{(1,4)}$=45.36	$t^4_{(2,3)}$=8.68	$t^4_{(3,5)}$ =12.38	$t^4_{(4,3)}$=6

(v_4,v_6)	(v_5,v_6)	(v_5,v_7)	(v_6,v_5)	(v_6,v_7)	目标函数值 Z(x)	步长	收敛指标
$t^0_{(4,6)}$=5	$t^0_{(5,6)}$=8	$t^0_{(5,7)}$=5	$t^0_{(6,5)}$=7	$t^0_{(6,7)}$=5			
$x^1_{(4,6)}$=84	$x^1_{(5,6)}$=0	$x^1_{(5,7)}$=0	$x^1_{(6,5)}$=26	$x^1_{(6,7)}$=30	10807.9		
$t^1_{(4,6)}$ =40.61	$t^1_{(5,6)}$=8	$t^1_{(5,7)}$=5	$t^1_{(6,5)}$ =9.05	$t^1_{(6,7)}$ =6.33			
$y^1_{(4,6)}$=0	$y^1_{(5,6)}$ =28	$y^1_{(5,7)}$ =30	$y^1_{(6,5)}$=0	$y^1_{(6,7)}$=0		0.447	
$x^2_{(4,6)}$ =46.45	$x^2_{(5,6)}$ =12.52	$x^2_{(5,7)}$ =13.41	$x^2_{(6,5)}$ =14.38	$x^2_{(6,7)}$ =16.59	3184.22		0.3318
$t^2_{(4,6)}$ =8.33	$t^2_{(5,6)}$ =8.06	$t^2_{(5,7)}$ =5.09	$t^2_{(6,5)}$ =7.19	$t^2_{(6,7)}$ =5.12			
$y^2_{(4,6)}$ =58	$y^2_{(5,6)}$=0	$y^2_{(5,7)}$=0	$y^2_{(6,5)}$=0	$y^2_{(6,7)}$ =30		1	

$x_{(4,6)}^{3}$=58	$x_{(5,6)}^{3}$=0	$x_{(5,7)}^{3}$=0	$x_{(6,5)}^{3}$=0	$x_{(6,7)}^{3}$=30	2802.25		0.0987
$t_{(4,6)}^{3}$ =13.09	$t_{(5,6)}^{3}$=8	$t_{(5,7)}^{3}$=5	$t_{(6,5)}^{3}$=7	$t_{(6,7)}^{3}$ =6.33			
$y_{(4,6)}^{3}$ =28	$y_{(5,6)}^{3}$=0	$y_{(5,7)}^{3}$ =30	$y_{(6,5)}^{3}$=0	$y_{(6,7)}^{3}$=0		0.0362	
$x_{(4,6)}^{4}$ =56.91	$x_{(5,6)}^{4}$=0	$x_{(5,7)}^{4}$ =1.09	$x_{(6,5)}^{4}$=0	$x_{(6,7)}^{4}$ =28.91	2798.84		0.0092
$t_{(4,6)}^{4}$ =12.51	$t_{(5,6)}^{4}$=8	$t_{(5,7)}^{4}$=5	$t_{(6,5)}^{4}$=7	$t_{(6,7)}^{4}$ =6.15			

从表 3.1 可以看出，经过 3 次迭代计算，得到了满足收敛准则的均衡流量模式，随着迭代过程的推进，目标函数值减小的幅度越来越小，在接近均衡状态时，相邻迭代次数之间的目标函数值基本保持不变。

由表 3.1 得均衡路段流量如下：

$x_{(1,2)}^{4}$ =72.1	$x_{(1,4)}^{4}$ =74.91	$x_{(2,3)}^{4}$ =52.09	$x_{(3,5)}^{4}$ =27.09	$x_{(4,3)}^{4}$ =0	$x_{(4,6)}^{4}$ =56.91	$x_{(5,6)}^{4}$ =0	$x_{(5,7)}^{4}$ =1.09	$x_{(6,5)}^{4}$ =0	$x_{(6,7)}^{4}$ =28.91

可以看出，$(v_4,v_3),(v_5,v_6),(v_6,v_5)$三条路段未被分配流量，即没有经过这三条路段的路径。将得到的均衡流量分配到各路段上，计算各路径旅行时间，并与未使用路径的旅行时间作比较，如下表所示：

表 3.2

OD 对	（1,2）	各路径旅行时间	（1,3）	各路径旅行时间	（1,4）	各路径旅行时间
使用路径	1-2	37.95	1-2-3	46.63	1-4	45.36
使用路径						
未使用路径			1-4-3	51.36		
未使用路径						

（1,5）	各路径旅行时间	（1,6）	各路径旅行时间	（1,7）	各路径旅行时间
1-2-3-5	59.01	1-4-6	57.87	1-2-3-5-7	64.01
				1-4-6-7	64.02
1-4-3-5	63.74	1-2-3-5-6	67.01	1-2-3-5-6-7	73.16
1-4-6-5	64.87				

从表 3.2 可以看出，在相同 OD 对之间，所有使用路径的旅行时间相等并且最小，所有未被使用路径的旅行时间大于或等于使用路径的旅行时间。这符合 Wardrop 第一原则。

三、结论

本文运用 UE 问题的 Frank-Wolfe 算法解决了固定需求下的交通网络流分配问题。利用 matlab 进行了编程，得到了数值结果并进行了分析。

以 Frank-Wolfe 方法为代表的路段算法具有许多优点：①遵循 Wardrop 均衡原则，所有使用路径具有相等的旅行时间，并且不高于未使用路径的旅行时间；②因为不必要记忆“使用路径”，所以无需太大的储存空间，对计算机存储容量要求不高；③算法构造简单，容易被交通工程师理解和接受。路段算法已经完全取代 OD 分割算法，成为交通网络流问题的基本算法，在交通规划等应用问题中发挥着极其重要的作用。

然而 Frank-Wolfe 方法在接近解的邻域收敛稳定性很差，表现为目标函数值发生振荡现象，完全收敛需要无数次迭代，甚至可能不收敛。因此，当面对更加复杂的网络问题时，需进一步学习更加高效的用户均衡网络流算法来解决问题。

四、参考文献

[1]程琳.城市交通网络流理论[M].南京：东南大学出版社，2010
[2]刘灿齐.现代交通规划学[M].北京：人民交通出版社，2001