1. 核心模块设计

在整网优化的过程中需要对特定的一条数据流进行找寻满足其时延和丢包率约束的花费最小的路径。在本节中，我们将介绍核心算法的主要思想。

该算法主要是在经典的求最短路的Dijkstra算法的基础上进行改进，最终求得的最短路要满足约束条件。在实施Dijkstra算法求解最短路的过程中，目标函数是原先总花费和多约束的加权求和；在每次选取某条边加到路径上之前，还要判断此时的路径是否满足时延约束和丢包率约束。如果是，加边进路径；否则调整权重因子，重新实施Dijkstra算法。

我们以加权的形式将约束和目标统一为

**C=cost+w1\*delay+w2\*passratio 　(3.1)**

其中w1,w2为权重。其中cost是边上的传输花费单价属性，delay为边上的时延属性，passration为边上传数据包的到达率（到达率=1-丢包率）的属性。算法在求解满足时延和丢包率的两个约束下极小化花费时，通过先确定乘子的上下界，然后逐步二分法调整每个约束对应的Lagrange 乘子，使得两个乘子迭代达到最优解对应的Lagrange 乘子。该算法理论上是收敛的，即，算法最终可以找到满足时延和丢包率约束的最短路。

**算法1.(基础算法)**

Min f(x) s.t. g(x)<=0

Step 0： 输入w=w1=0, M=1000, p=10(可以改动该数据)，N=0，eps=10^(-3).

Step 1(找到下界)： 计算 min f(x) 得到最优解x0. 若g(x0)<eps, 则停止计算且输出x\*=x0; 否则, N=N+1，w=w+p, 转Step2.

Step 2(找到上界)： 计算 min f(x)+wg(x) 得到最优解x0. 若- eps<g(x0)<eps, 则停止计算且输出x\*=x0. 若g(x0)>=eps, 则当N>M时停止计算且输出信息“约束条件g(x)<=0无法满足”,当N<=M时取N=N+1, w1=w, w=w+p, 重复Step 2 上述步骤. 若g(x0)<= - eps，取w2=w且x\*=x0, 转Step3.

Step 3（二分法）：若w2-w1<eps 或N>M时停止计算且输出x\*. 否则取w=(w1+w2)/2, 计算 min f(x)+wg(x) 得到最优解x0. 若- eps<g(x0)<eps, 则停止计算且输出x\*=x0. 若g(x0)>=eps, 取w1=w, 重复Step3上述步骤. 若g(x0)<= - eps，取w2=w且x\*=x0, 重复Step3上述步骤.

**算法2. （核心算法）**

Min f0(x) s. t. f1(x)<=0, f2(x)<=0.

Step 0： 输入w2=w21=0, M=1000, p=10，N=0，eps=10^(-3).

Step1(找到下界) ：取f=f0+w2f2, g=f1, 调用算法1得到x0. 若f2(x0)<eps, 则停止计算且输出x\*=x0; 否则, N=N+1，w2=w2+p, 转Step2.

Step 2(找到上界)： 取f=f0+w2f2, g=f1, 调用算法1得到x0. 若- eps<f2(x0)<eps, 则停止计算且输出x\*=x0. 若f2(x0)>=eps, 则当N>M时停止计算且输出信息“约束条件f2(x)<=0无法满足”; 当N<=M时取N=N+1, w21=w2, w2=w2+p, 重复Step 2 上述步骤. 若f2(x0)<= - eps，取w22=w2且x\*=x0, 转Step3.

Step 3（二分法）：若w22-w21<eps或N>M时停止计算且输出x\*. 否则取w2=(w21+w22)/2, 取f=f0+w2f2, g=f1, 调用算法1得到x0. 若- eps<f2(x0)<eps, 则停止计算且输出x\*=x0. 若f2(x0)>=eps, 取w1=w, 重复Step3上述步骤. 若f2(x0)<= - eps，取w2=w且x\*=x0, 重复Step3上述步骤.

核心算法——带双约束的Dijkstra算法

需要说明的是，算法框架中涉及到的f0，f1以及f2分别对应我们整网优化中的花费，时延以及丢包率函数。算法1中的step2 ，min f(x) 指的是找寻花费最小的路径（经典的最短路算法）。