基于群智能算法的改进 Webster 交通信号配时优化模型

摘 要

城市道路交叉口信号配时优化是改善城市交通拥堵、保障道路畅通安全的重要途径。本文采用 Webster 模型、Synchro 仿真系统、群智能算法中的遗传算法及蚁群算法对城市相邻两交叉口信号配时优化问题进行了研究。

在交叉口信号控制中，较为常用的优化算法是 Webster 算法，Webster 模型是纯解析模型，利用该模型我们计算出优化的有效绿灯时间、绿信比、车辆延误时间等关键参数，并对车均停车时间、交叉口通行能力进行建模计算，将优化结果和实际情况进行了对比分析表明其配时方案可以有效调高实际平面交叉口的通行能力和服务水平，但是通过计算验证和文献调研，Webster 模型在近饱和交通流量时准确性很低。因此利用 Synchro 交通仿真系统，基于 Webster 模型提供的控制系统参数，对该相邻交叉口的交通流进行了仿真模拟，给出了一个信号周期内车均延误时间、车均停车时间、交叉口通行能力等重要评价参数，甚至包括总燃油消耗量，克服了 Webster 模型只能描述低流量的缺点。

另外，我们从人工智能的角度来研究交叉口信号配时优化问题。遗传算法是模拟生物自然环境中的遗传和进化过程而形成的一种自适应全局优化概率搜索算法。但是对于较复杂的优化问题，其容易出现早熟收敛的现象，并且初始选定的种群对该算法的影响会很大，有可能影响最优解的好坏以及算法效率的高低，故我们又采用蚁群算法对信号配时进行优化。蚁群算法的基本思想来源于蚂蚁之间的交流过程，它是通过信息素的累积和更新收敛于最优路径，且具有信息正反馈、分布式计算、并行性以及强鲁棒性的优点。为了对有效绿灯时间等控制参数进行优化，我们以车辆延误时间、通行能力、平均停车次数为目标函数构建了配时优化的非线性规划模型，分别用遗传算法和蚁群算法对此非线性规划模型进行求解，得到了使延误时间最小、通行能力最大、平均停车次数最小的信号配时方案，并进行了与 Webster 优化结果的对比分析。最后，我们对算法的时间复杂度、空间复杂度、收敛性进行了讨论。

模型分析认为，目前武汉市某相邻交叉口的配时方案和优化方案有一定距离，有待改进。用 Synchro 仿真和蚁群算法得到的交叉口信号配时能有效提高实际平面交叉口的通行能力和服务水平，减少城市交通网的交通延误，且解的收敛性及鲁棒性较好，复杂度可以接受。通过群智能算法和仿真改进的 Webster 模型可以更好用于指导实际交通控制。

关键词：信号配时优化，Webster 模型，Synchro 仿真，遗传算法，蚁群算法

目 录

一、 问题重述

随着我国城市化速度的加快以及城市规模的不断扩大，交通供需矛盾日益突出，在城市交通网络中产生的交通拥堵现象日趋严重，严重影响了社会经济的发展和人民生活水平的提高。

为了提高城市道路交通管理水平，改善城市交通秩序，保障公路交通的畅通与安全，当今世界各国普遍使用智能交通系统。在该系统中，核心的问题是交通信号智能控制。平面交叉口是道路交通的主要冲突点，不仅机动车数量多，而且行人和非机动车也在同一平面通过。目前在我国的大、中型城市交通管理中，普遍采用的是单点定时交通控制。定时控制这种传统信号灯控制方法会造成某些方向绿时浪费，而在有些防线车辆通行又延误严重。因此，优化交叉信号配时是提高交叉口运行效率最有效的方法之一。

现有武汉市某相邻两交叉口 A、B，交通数据如表 1 所示，交叉口 A 的第一、二、三、四相位时间分别为 56s、23s、35s、26s。测得两个交叉口的相位差为 8s，交叉口 B 的第一、二、三、四相位时间分别为 47s、21s、39s、22s。每个相位时间都包括 3s 黄灯时间、1s 红灯时间。

请你结合表 1 的数据，设计通用模型与算法（从时间复杂度、空间复杂度、收敛性进行对比分析），对交通信号进行配时优化研究，求解出改善后的交通配时方案并进行仿真检验，以期有效指导实际平面交叉口的通行能力和服务水平，减少城市交通网的交通延误，改善城市交通现状。同时你还需指出你的模型的特点与创新行，以便我们认可你的研究成果。

表 1. 交通数据

交通数据		交通流量/（PCU h^{-1}）			车均延误时间/s		
		左转	直行	右转	左转	直行	右转
武汉市 A 交叉口	东进口	366	1394	98	7.55	6.72	5.80
	西进口	295	166	72			
	南进口	525	408	300	8.16	4.89	5.63
	北进口	100	394	576			
武汉市 B 交叉口	东进口	802	1154	576	5.26	11.33	4.96
	西进口	450	304	329			
	南进口	169	420	84	5.03	13.61	5.34
	北进口	132	535	90			

二、 问题分析

1.1 概 论

交叉口是城市交通网络中最重要的组成部分，是交通拥堵的主要发生地[1]。就信号控制的单点平面交叉口而言，信号配时优化往往对于减少车流的平均延误、停车次数、提高交叉口的通行能力及服务水平非常重要。交通信号配时主要包括对交通信号相位及相序、周期时长、相位绿信比等的优化。对于一个特定流量现状的交叉口而言，可以通过对上述参数的优化来实现更好的通行能力与服务水平。本问题所研究的实例采用的是四相信号控制系统，我们在本文的分析中沿用这种四相结构，而着重对绿信比、延误时间进行优化。四相位控制系统如图 1 所示。

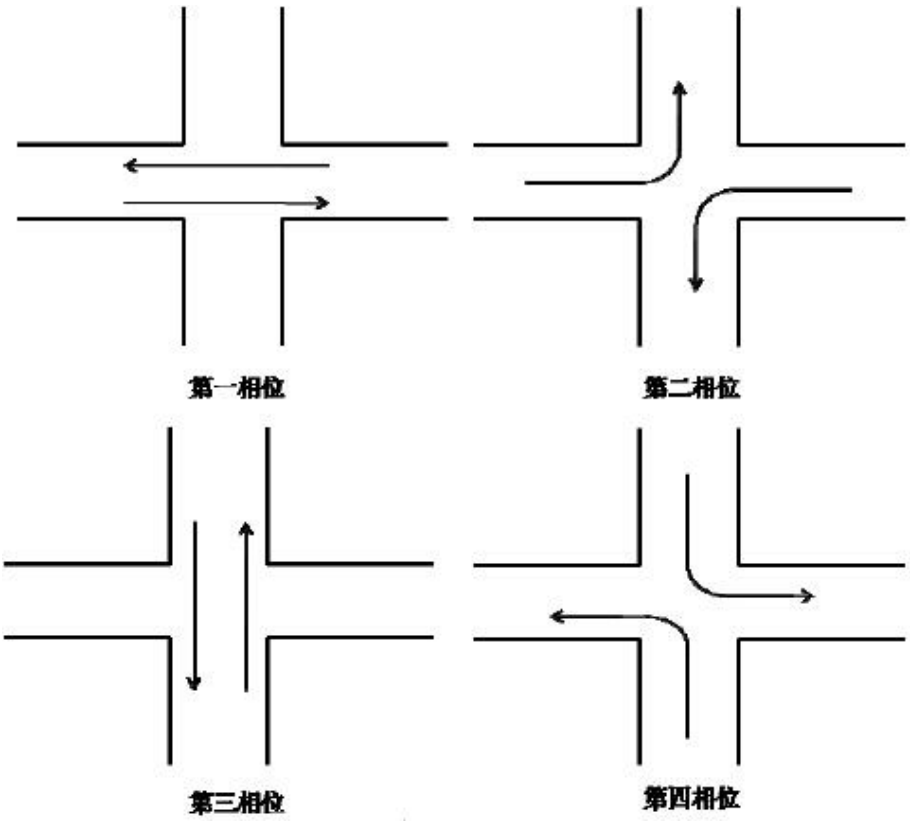

图 1. 四相位交通控制系统

在交叉口信号配时优化研究中，常用的算法是 Webster 算法[2]，Webster 算法的特点是仅将某个交叉点的车辆延误作为其性能指标，其中车辆延误可以根据交通流信息统计得出。但是，使用 Webster 法进行信号配时优化局限于不太大的交叉口饱和度。人工智能技术的发展使得大饱和度交通流时交叉口信号控制优化成为可能，群智能算法开始广泛用于交通信号配时优化当中，诸如遗传算法、蚁群算法等相继被用于信号配时优化研究中，取得了很好的效果。另外，不同国家也开发

的不同的交通信号配时优化系统，比如美国马萨诸塞州大学开发的 OPAC 系统[3,4]、澳大利亚的 SCATS 系统[5]、英国 TRRL 的 TRANSYT 系统[6]、美国的 Synchro 系统及类似的 VISSIM 系统等[7,8]。在本问题的分析中，我们可以综合考虑利用较简单的 Webster 法、较复杂的遗传及蚁群算法和 Synchro 仿真的类比对比研究，来给出一个比较合理的配时优化方案，对于这些方法的比较，也可以帮助我们认识它们自身的优点和局限性。

2.2 具体分析

本问题的背景设定在武汉市某相邻交叉口 A、B。A、B 两交叉口的交通信号配时都采用四相设计，第一、二、三、四相位时间分别为 56s、23s、35s、26s。测得两个交叉口的相位差为 8s，交叉口 B 的第一、二、三、四相位时间分别为 47s、21s、39s、22s。每个相位时间都包括 3s 黄灯时间、1s 红灯时间。基于题给数据，可知其空间结构及交通流量分布如图 2 所示。

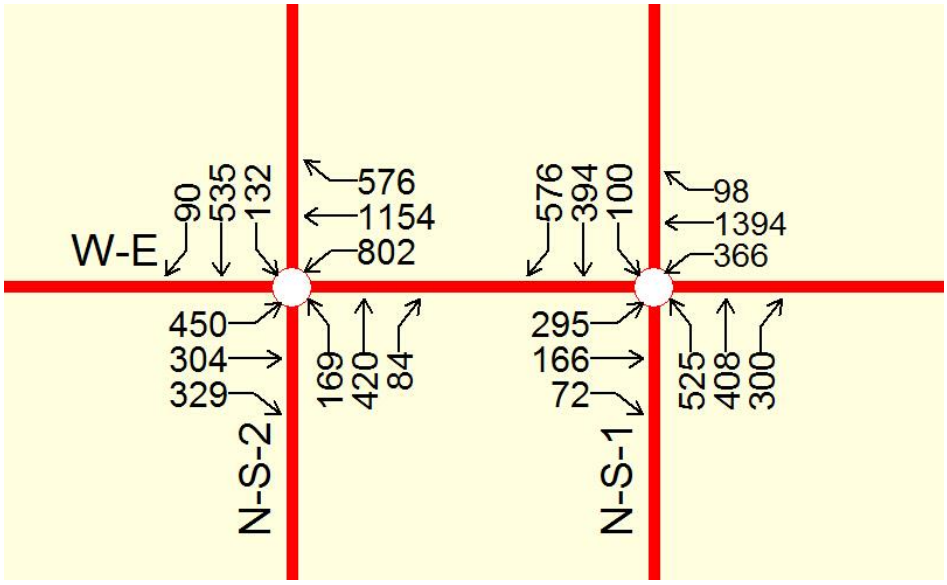

图 2. A、B 两交叉口空间位置及交通流量分布（单位：PCU h^{-1}）

A 为图中的东—西（W-E）与南—北 1（N-S-1）的交叉，B 为东—西（W-E）与南—北 2（N-S-2）的交叉。现有红绿黄分布如图 3 示。

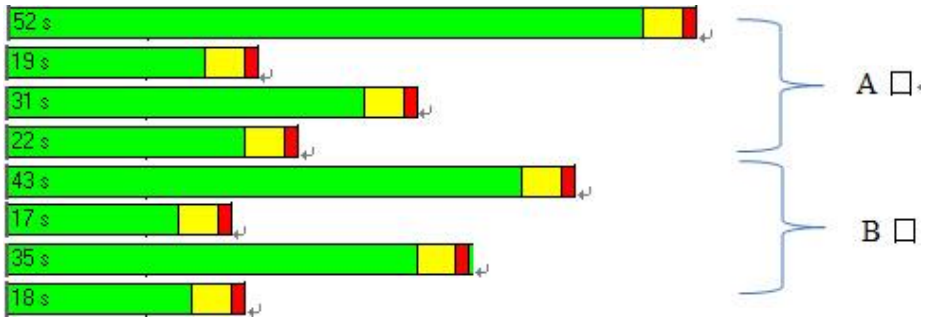

图 3. 现有各相信号分布

基于上述已知，可以利用 Webster 算法得出初步的优化结果。但考虑到 Webster 法本身的局限性，我们同时利用 Synchro 仿真计算、遗传算法优化、蚁群算法，以延误时间为目标函数，来对 Webster 法的结果进行对比分析与修正，提出合理的配时方案。

三、 模型假设

1. 同一转向的车辆通过交叉口的平均速度相同，通过路口所用时间相同。
2. 右转车辆是时时放行的，受四相信号控制系统调节的是直行和左转车辆。
3. 所有司机遵守调度规则。
4. 交叉路口处不发生交通事故。
5. 尽管在 Synchro 仿真系统中可以考虑大小车辆同时通行的情况，我们在本模型的其他计算中均假设通过路口的车辆相同。

四、 符号说明

符号	符号说明
V_e	等效交通流量（$PCU \cdot h^{-1}$ ）
H	公交车、货车数量（$PCU \cdot h^{-1}$ ）
L	左转车数量（$PCU \cdot h^{-1}$ ）
n	进口有效车道数，本文中取 3
T	信号周期长度（s）
P	相位数，本实际问题中考虑四相控制，故$P=4$
Q_i	相位i 流量（$PCU \cdot h^{-1}$ ）

S_i	相位i 的饱和流量（$PCU \cdot h^{-1}$ ）
y_i	相位i 的流量比
Y	交叉口流量比，$Y=\sum_{i=1}^{P} y_i$
C_0	信号最佳周期（s）
g_e	各相位有效绿灯时间（s）
D_i	第i 相位每辆车的平均延误（s）

五、 模型的建立与求解

5.1 Webster 算法与 Synchro 仿真

在信号周期确定之后，各信号相位的绿灯时间是按各相位临界车道的交通流量比进行比例分配的，在非饱和的交通状态下，这一模型充分考虑各个方向上的交通需求，切各相位分配的绿灯时间合理，就可以有效降低信号交叉延误及排队长度等指标。较小流量区间内各车辆之间的相互干扰较小，Webster 模型简单的解析结构所给出的计算结果是比较准确的，流量越大，车辆之间的相互干扰也就越严重，Webster 方法的准确性就会下降，继而我们可以使用 Synchro 仿真来处理这个问题。这里有必要建立 Webster 方法的解析结构。

根据交通量确定信号灯周期长度时，需用等效交叉口流量，其换算公式为：

$$V_e=(V+0.5H+0.6L)/n \qquad (1)$$

考虑到假设 5，我们取$H=0$。周期T 可按下式计算：

$$T=13330P/(1333-V_e) \qquad (2)$$

饱和流量从原则上应该根据$S=522\times W(PCU\cdot h^{-1})$来计算，其中$W$ 为车道宽度，但此数据在本问题背景中缺失，故我们参照文献[9]的建议，取饱和流量序列为：2800；2800；1800；1800；3000；3000；2800；2800$(PCU\cdot h^{-1})$。流量比即为：$y_i=Q_i/S_i$。信号最佳周期C_0 可表达为：

$$C_0=\frac{1.5X+5}{1-Y} \qquad (3)$$

其中 $X=\sum_i (X_s+X_i-A)$，X_s 为车辆起动损失时间，通过实测获取，无数据时可取 3s；I_i 为绿灯间隔时间，及黄灯时间加全红灯清路口时间，一般黄灯为 3s，本

例全红灯为 1s，A为黄灯时间，本例为 3s。一周期内总的有效绿灯时间可以表达为$G_e = C_0 - X$，各相位有效绿灯时间则为$g_{ei} = G_e y_i / Y$，这是容易理解的。第i相位的绿信比因而可以写为

$$g_i = g_{ei} / C_0 = \frac{G_e y_i}{YT} \tag{4}$$

基于公式（4），可以计算延误时间，Webster 模型中延误的表达式为

$$D_i = \frac{C_0(1-g_i)^2}{2(1-y_i)} + \frac{y_i^2}{2Q_i g_i (g_i - y_i)} \tag{5}$$

综上可以计算出相关参数如表 2 所示。

表 2. Webster 优化参数结果

		周期长度 T	饱和流量 Si	流量比 y_i	有效绿灯时间	绿信比 g_i	车辆延误 Di（s）
武汉市 A 交叉口	东进口	83.3	2800.0	0.7	50.2	0.60	19.5
	西进口	48.6	2800.0	0.2	14.4	0.30	14.9
	南进口	65.3	1800.0	0.7	51.8	0.79	4.4
	北进口	55.8	1800.0	0.6	44.9	0.81	2.6
武汉市 B 交叉口	东进口	162.3	3000.0	0.8	79.3	0.49	135.8
	西进口	60.5	3000.0	0.4	33.9	0.56	9.1
	南进口	49.6	2800.0	0.2	22.6	0.46	9.7
	北进口	50.6	2800.0	0.3	25.4	0.50	8.6

考虑第i相位的平均停车次数[10]

$$H_i = 0.9\frac{1-y_i}{1-g_i}, i = 1,2,\cdots,P \tag{6}$$

另一个重要的量则为交叉口通行能力[10] Z_i

$$Z_i = g_{ei} S_i / T \tag{7}$$

我们可以算得这两个重要参数并列于表 3 中

表 3. 交叉口通行能力与车均停车次数

		交叉口通行能力 *Hi*(PCU h^{-1})	车均停车次数 *Zi*
武汉市A交叉口	东进口	1687.2	1.1
	西进口	828.5	0.8
	南进口	1428.3	0.6
	北进口	1450.8	0.4
武汉市B交叉口	东进口	1466.8	2.9
	西进口	1684.0	0.6
	南进口	1275.3	0.6
	北进口	1407.0	0.6

而基于表 2 的计算结果，我们实际已经可以看出，在高流量下 Webster 方法的优化的局限性，如图 4 所示。

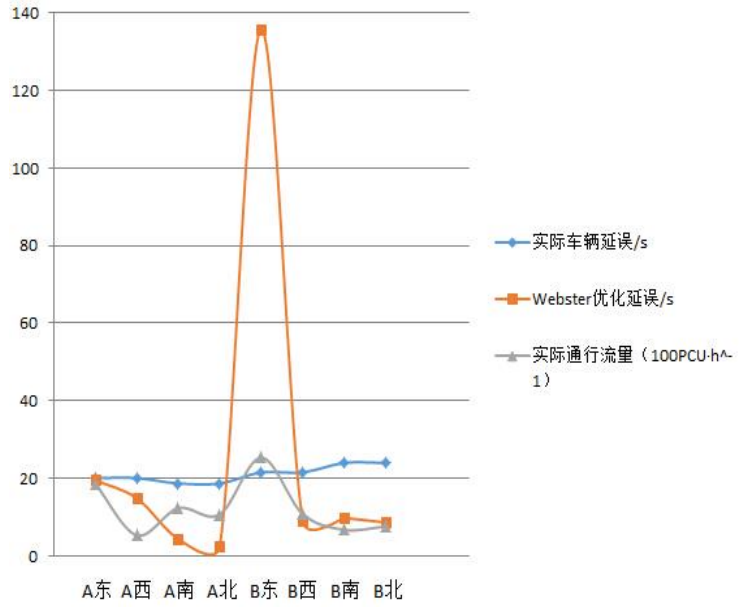

图 4. 在不同流量下 Webster 优化延误与实际延误的关系

从图 4 中我们可以看到，除了 B 东这种情况，Webster 优化延误都要比实际延误小，这说明 Webster 方法给出了比较正确的优化，但因为在 B 东有一个流量的高峰，这时 Webster 方法给出了一个近乎荒谬的优化结果，这个结果比实际的还要

大近 6 倍，这说明了该方法的局限性。但这个关系也说明了 Webster 方法在较小流量的大区间内是适用的。

在此结果基础上，我们可以进一步地进行 Synchro 仿真，以期解决 Webster 方法在大流量下优化不准的问题。

按照四相控制设计，考虑交叉口 A 周期 129 秒，交叉口 B 周期 140 秒，相位差 8 秒，利用 Synchro 软件构造出仿真交通流如图 5 所示。

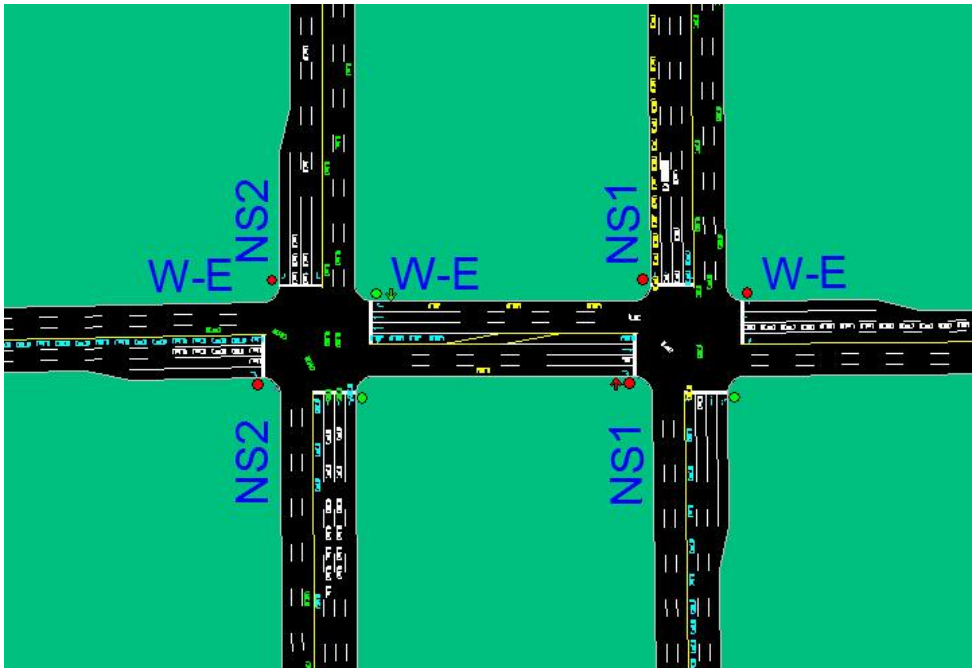

图 5. Synchro 仿真交通流

我们对本题给定的交通情景在一个周期（根据题给数据，在仿真设定周期为 120s, 即 2s）内进行了仿真，其结果列出如下。

对 A 口（W-E 与 NS2 交点）及 B 口（W-E 与 NS1）分别统计了我们关心的参数。

表 4. A 口在一个信号周期内仿真参数结果

运动形式	总延误 (hr)	车均延误 (s)	总停车次数	车均停车次数
	0.2	0.314410	22	0.09607
	0	0	3	0.0131
	0	0	3	0.0131
	0.1	0.157205	7	0.030568
	0	0	2	0.008734
	0	0	0	0
	0.1	0.157205	6	0.026201
	0.1	0.157205	9	0.039301
	0	0	3	0.0131
	0	0	3	0.0131
	0.2	0.314410	14	0.061135
	0	0	2	0.008734

表 5. B 口在一个信号周期内仿真参数结果

运动形式	总延误时间(hr)	车均延误时间(s)	总停车次数	车均停车次数
	0	0	2	0.008734
	0	0	0	0
	0	0	0	0
	0.2	0.314410	12	0.052402
	0.6	0.943231	40	0.174672
	0	0	3	0.0131
	0.2	0.314410	17	0.074236
	0.1	0.157205	10	0.043668
	0	0	7	0.030568
	0	0	3	0.0131
	0.1	0.157205	6	0.026201
	0	0	9	0.039301

表 6. 两交叉口在一个信号周期内的总体情况

进车数	420
出车数	235
总延误时间（hr）	6.9
总停车次数	424
总燃油消耗(L)	49.77519

用上述的 12 种基本运动形式可以组合出以东西南北的四种形式，即 E=EBL+EBT+EBR、W=WBL+WBT+WBR，以此类推，组合后的结果可以和 Webster 优化的结果进行对比，结果如下表所示。

表 7. 一个信号周期内 Webster 方法与 Synchro 仿真优化结果对比

	Synchro 仿真优化的车均停车次数	Webster 方法优化的车均停车次数	Synchro 仿真优化的车均延误（s）	Webster 方法优化的车均延误(s)	实际车均延误（s）
A 东进口	0.122271	1.063187	0.31441	0.45476	0.467101
A 西进口	0.039301	0.782686	0.157205	2.068047	2.787245
A 南进口	0.082969	0.590056	0.31441	0.19776	0.835696
A 北进口	0.078603	0.430483	0.31441	0.156145	1.127236
B 东进口	0.008734	2.948424	0	1.190216	0.188827
B 西进口	0.240175	0.61784	1.257642	0.500538	1.184949
B 南进口	0.078603	0.645145	0.157205	1.044037	2.585837
B 北进口	0.148472	0.61366	0.471616	0.806637	2.254842

现在，我们就可以清晰地看出 Webster 方法在描述大流量情形的局限性了，如图 6 所示。

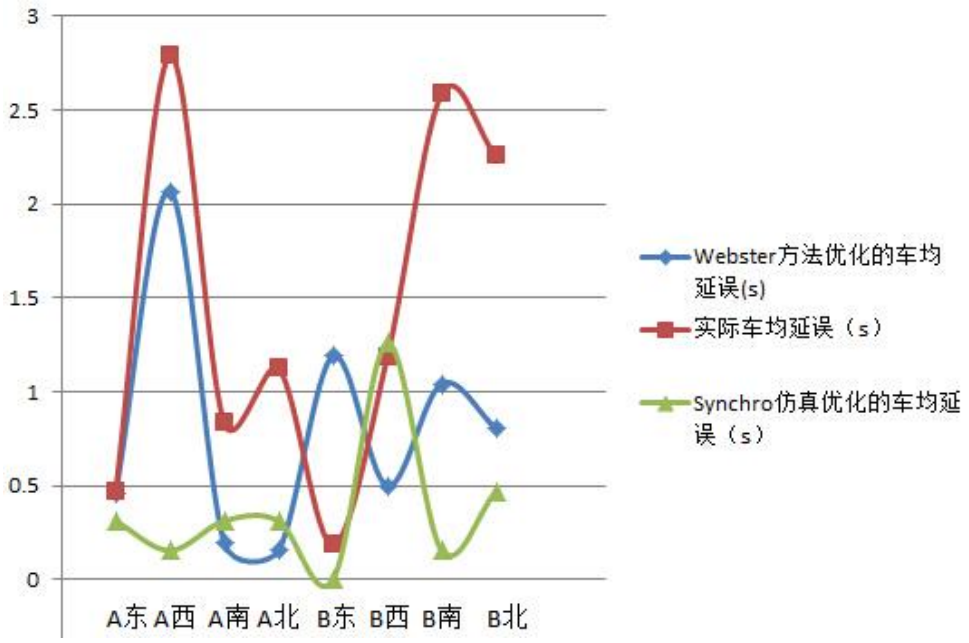

图 6. Webster 方法优化延误与 Synchro 仿真延误及实际延误的对比

B 东的流量是最大的，正如前述，Webster 方法的延误优化值反而比实际的要大。而 Synchro 仿真可以克服大流量描述不准的问题，其车均延误优化值一直处于实际值之下。流量越大，车辆之间的干扰也就越大，这是造成 Webster 方法在近饱和流量时优化失准的根本原因。从原则上讲，任何一种交通控制系统在实际实施的时候都会因为人为因素，如驾驶员不遵守规则、驾驶习惯差、车辆非均质性、随机性等而使其效果大打折扣。实际情况包含了所有的这些不利影响，故优化的结果应该具有小于实际值的一致性。

5.2 遗传算法配时优化

在前面我们探讨了 Webster 方法与 Synchro 仿真在解决这个问题中的应用，得到了一些有用的结果。但我们注意到一个事实，即前述优化模型在绿信比的优化上是基于 Webster 优化的，也就是说，Synchro 仿真虽然解决了 Webster 法在近饱和流量时失准的问题，但没有解决绿信比的优化问题，因为在 Synchro 仿真中必须要首先给定一个绿信比方案才能得出其他各参数的仿真值。为了更好地基于减少延误时间而优化绿信比，我们采用智能群算法之一的遗传算法来探讨这个问题。有必要对遗传算法的基本思想做出说明。

在遗传算法中，将 n 维决策向量 $X=[\mathrm{x}_1,\mathrm{x}_2,\cdots,\mathrm{x}_n]^T$ 用 n 个记号 X_i（i=0,1,2…）所组成的符号串 X 来表示

$$X=X_1X_2\cdots X_n \Rightarrow \dot{X}=[\mathrm{x}_1,\mathrm{x}_2,\cdots,\mathrm{x}_n]^T \qquad (8)$$

把每一个 X_i 看成一个遗传基因，它的所有可能取值称为等位基因，这样，X 就可看作是由 n 个遗传基因所组成的一个染色体。一般情况下，染色体的长度 n 是固定的，但对一些问题 n 也可以是变化的。根据不同的情况，等位基因可以是一组整数，也可以是某一范围之内的实数值，或者是纯粹的一个记号。最简单的等位基因是由 0 和 1 这两个整数组成的，相应的染色体就可以表示为一个二进制符号串。这种编码所形成的排列形式 X 是个体的基因型，与它对应的 X 值是个体的表现型。染色体 X 也称为个体 X，对于每一个个体 X，要按照一定的规则确定出其适应度。个体的适应度与其对应的个体的表现型 X 的目标函数值相关联，X 越接近于目标函数的最优点，其适应度越大。

遗传算法中，决策变量 X 组成了问题的解空间。对问题最优解的搜索是通过对染色体 X 的搜索来进行的，从而由所有的染色体 X 就组成了问题的搜索空间。生物的进化是以集团为主体的。与此相对应，遗传算法的运算对象是由 M 个个体所组成的集合，称为群体。与生物一代一代的自然进化过程相类似，遗传算法的运算过程也是一个反复迭代过程，第 t 代群体记作 $P(t)$，经过一代遗传和进化后，得到第 t+1 代群体，他们也是由多个个体组成的集合，记作 $P(t+1)$。这个群体不断地经过遗传和进化操作，并且每次都按照优胜劣汰的规则将适应度较高的个体更多地遗传到下一代，这样最终在群体中将会得到一个优良的个体 X，它所对应的表现型 X 将达到或接近于问题的最优解 X^*。

- 遗传算法的基本操作：

(1) 编码：当采用遗传算法求解优化问题时，从表现型到基因型的映射称为编码，通常我们采用二进制编码。

(2) 初始群体的生成：遗传算法将初始种群作为第一代，逐代进化，直到生成问题的最优解。

(3) 适应度值评估：遗传算法主要通过适应度函数来判别每个个体或解的好坏，即适应度值越大，向下一代遗传的概率就越大，反之亦然。

(4) 选择：根据各个个体的适应度,按照一定的规则或方法，从第 t 代群体中选择出一些优良的个体遗传到下一代群体中。

(5) 交叉：将群体内的各个个体随机搭配成对,对每对个体,以某个概率交换它们之间的部分染色体。

(6) 变异：对群体中的每一个体,以某一概率改变某一个或某一些基因座上的基因值为其他的等位基因。

遗传算法的算法流程图如图 7 所示。

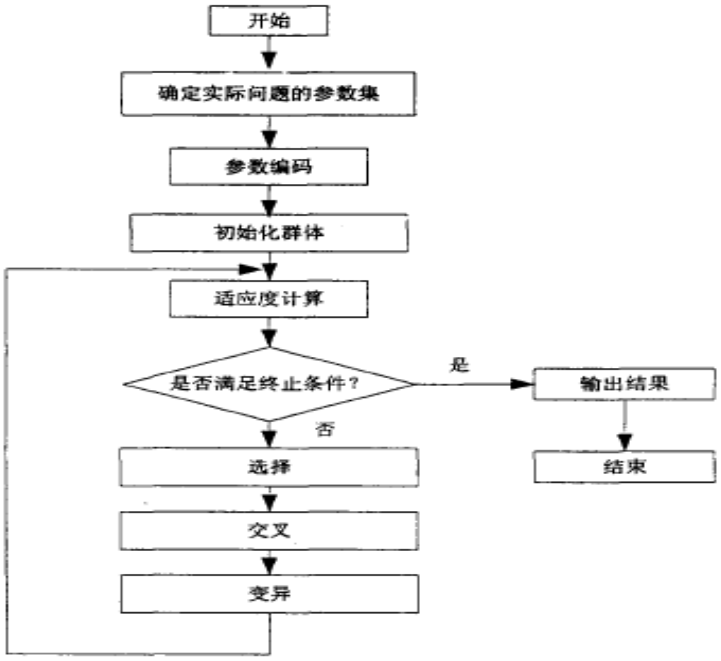

图 7. 遗传算法的算法流程图

在具体分析交叉口车辆平均延误优化的遗传算法时，需要找到正确的目标函数和约束条件。交叉口车辆平均延误主要由两部分组成[9]：一致性延误 d_u 和随机延误 d_r。其表达式分别为

$$d_{ui}=\sum_j \frac{C_0(1-g_{ei}/C_0)^2}{2(1-y_{ij})} \quad ; \quad d_{ri}=\sum_j \frac{x_{ij}^2}{2Q_{ij}(1-x_{ij})} \quad ; \quad D_i=d_{ui}+d_{ri},$$

i 代表相位，则各相位总延误时间为 $D=\sum_i D_i$。文献[3,11]改进了公式（5）并给出了第 i 相位车辆延误表达式

$$D_i=\frac{C_0Q_i\left(x-y_i\right)^2}{2x^2\left(1-y_i\right)}+\frac{x^2}{2\left(1-x\right)}+8, i=1,2,\cdots,P \qquad (9)$$

其中 x 为交叉口饱和度。这里加了 8s 主要考虑 8s 的相位差。根据文献[3]给出的建议，平均停车次数 H_i 有表达式 $H_i=\frac{y_i\left(x-y_i\right)}{2x^2\left(y_i-x^2\right)}, i=1,2,\cdots,P$。

优化的目标是最小的车辆延误，最小的平均平车次数，最大的交叉口通行能力。因此得到规划模型

$$\min\begin{cases} D_i = \dfrac{C_0 Q_i (x-y_i)^2}{2x^2(1-y_i)} + \dfrac{x^2}{2(1-x)} + 8, i = 1,2,\cdots,P \\ H_i = \dfrac{y_i(x-y_i)}{2x^2(y_i-x^2)}, i = 1,2,\cdots,P \end{cases}$$

$$\max\left\{S_i \times \frac{g_{ei}}{C_0}\right.$$

$$s.t.\begin{cases} 10 \le x_i \le 85, 1 \le i \le P; \\ g_{\min} \le g_{ei} \le g_{\max} \\ \sum_{i=1}^{4} g_{ei} = C_0 - X \\ 0.75 \le \dfrac{y_i C_0}{g_{ei}} \le 0.95, (1 \le i \le P) \end{cases} \quad (10)$$

利用遗传算法求取最优解。其求解代码及算法具体步骤说明参见附录。

采用 MATLAB 语言编制程序，对目标函数进行仿真计算，仿真过程中的一些具体参数设置为：个体数目 1000，最大遗传代数 500，变量的二进制位数 20，代沟 0.9，交叉概率 0.9。编写遗传算法程序（见附录二），在 MATLAB 下运行得到该问题的最优解：C1=[54.17 12.66 55.93 39.54]、C2=[63.46 29.86 19.88 22.36]，x=0.667。遗传算法迭代过程如下图所示。

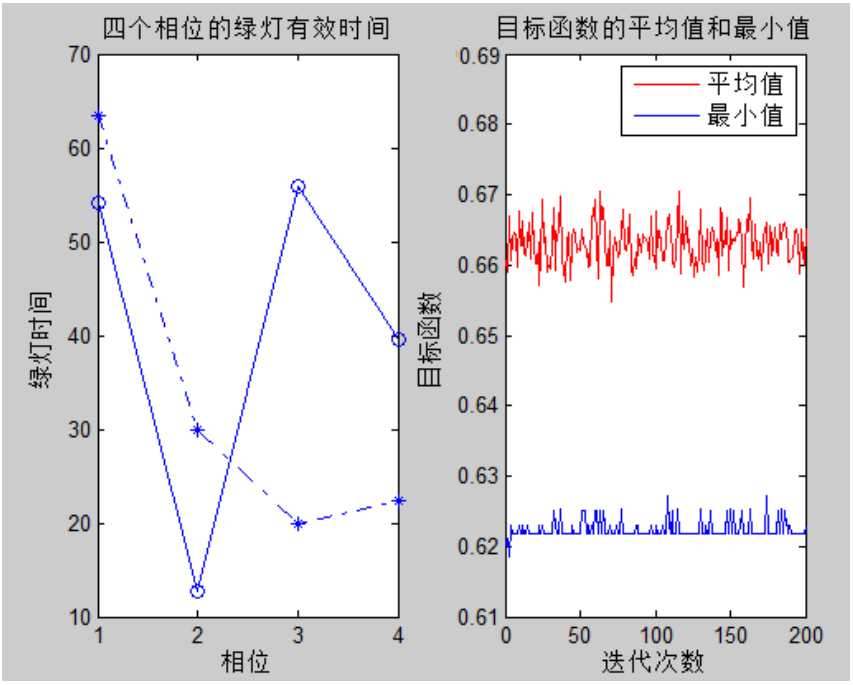

图 8. 遗传算法迭代过程

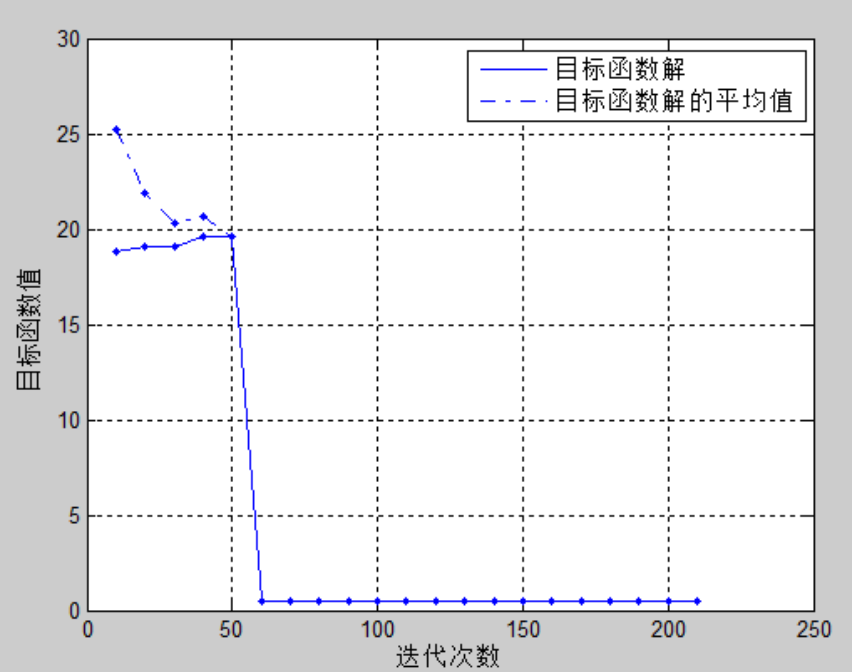

图 9. 目标平均值同迭代次数的关系

5.3 基于蚁群算法的修正 Webster 模型

在遗传算法求解最优绿信比（或有效绿灯时间）的过程中，存在快速收敛和遗弃全局最优解陷入局部的问题[3]，为了克服这个问题，我们采用另一种群智能算法—蚁群算法来处理这个问题。有必要对蚁群算法做基本的说明。

经过长时间观察蚂蚁的行为，仿生学家发现：蚂蚁虽然没有视觉，但它们行走时会在走过的路上留下一种叫做信息素的物质用来寻找路线。当它们到达一个陌生的地方时，就任意挑选一条路线继续行进，与此同时留下和路线长短对应的信息素，也就是说，蚂蚁行进的路线越长，其留下的信息素越少。当其它蚂蚁也经过这个地方时，它们会选择信息素较多的路线前行，从而构成了一个正反馈机制。最短路线上的信息素会越来越多，但其它路线上的信息素会随着时间的推移而慢慢挥发，到最后整个蚁群都会找到最短路线。另外，当蚂蚁的行进途中有障碍物出现时，它们也能迅速地寻找到新的最短路线。由此可见，在蚂蚁寻找最短路线的过程中，尽管每只蚂蚁的寻找能力有限，但由于信息素的存在，整个蚁群的行为具有了很高的自组织性，从而找到了最短路线。

蚁群算法的寻优过程分为适应和协作两个基本阶段。当处于适应阶段时，蚂蚁会根据附近的环境做出判断并留下一定量的信息素，即该路线被蚂蚁选择得越多，信息素浓度越大，则该路线被选择的概率就越大，然而随着时间的推移，信息素浓度会不断减小；当处于协作阶段时，各路线上的信息素浓度会不断地更新和调整，从而可以得到最短的路线。也就是说，蚁群算法充分利用蚂蚁自身的特

点，通过信息素浓度的变化，对优化问题不断地迭代求解，从而得到该问题的最优解。下图为蚁群算法的算法逻辑结构图。

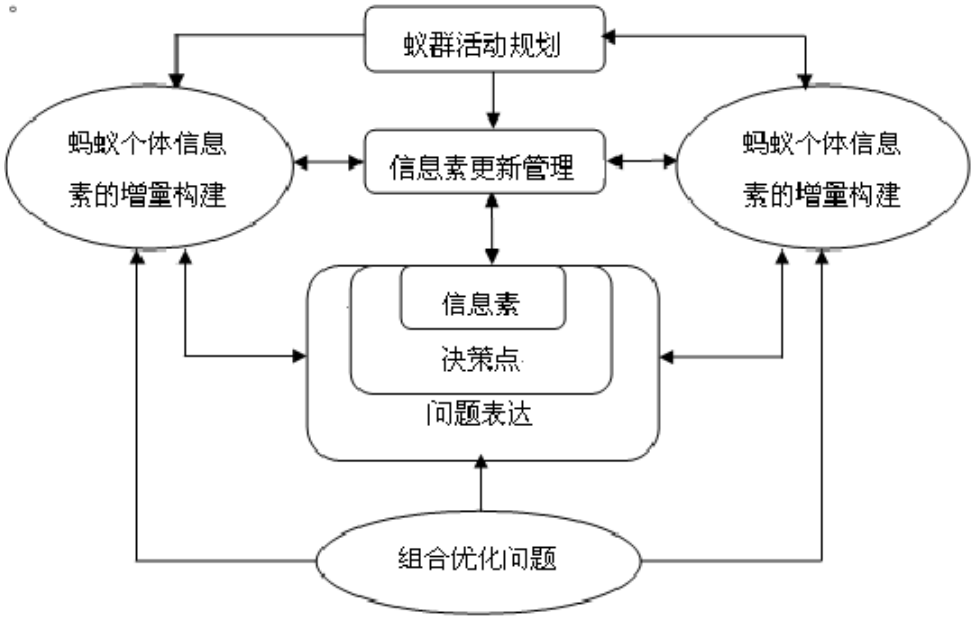

图 10. 蚁群算法逻辑结构图

在蚁群算法的正反馈机制下，根据一定的信息素更新规则对蚂蚁的信息素进行更新，然后对该组合优化问题进行优化求解，从而得到待求解问题的最优解。

基本蚁群算法的结构可用旅行商问题（TSP）来说明。设蚁群中有 m 只蚂蚁，将它们随机地放到 n 个城市中，城市 i 和 j 之间相距（i, j= 1,2,...,n），t 时刻城市 i 和 j 连线上的残留信息量为 $\tau_{ij}(t)$。起初每条路线上的信息量均相同，即 $\tau_{ij}(0)=Const$。对于蚂蚁 k（k =1,2,..., m），根据路线上的信息量来判断其行进方向。在蚂蚁行进过程中，根据每条路线上的信息量及其启发信息来计算它的状态转移概率。t 时刻蚂蚁 k 由城市 i 向城市 j 转移的概率 $p_{ij}^{k}(t)$ 定义为

$$p_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}}{\sum\limits_{s\in allowed_k}\tau_{is}^{\alpha}(t)\eta_{is}^{\beta}}, j\in allowed_k\\ 0, \text{否则}\end{cases} \tag{11}$$

其中，α 为信息素启发式因子，表示轨迹的相对重要性；β 为期望启发式因子，表示能见度的相对重要性；$allowed_k$ 为蚂蚁 k 下一步可以选择的城市集合，它随着蚂蚁的进化过程而动态改变；η_{ij} 为启发函数，表示城市 i 转移到城市 j 的期望

程度，表达式为$\eta_{ij}=1/d_{ij}$。当群体中的所有蚂蚁都走过 n 个城市后，t + n时刻路线(i,j)上的信息量要根据如下规则做出更新及调整。

$$\tau_{ij}(t+n)=\rho\bullet\tau_{ij}(t)+\Delta\tau_{ij}(t)$$

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \qquad (12)$$

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q}{L_k}, \text{若蚂蚁k在本次循环中经过（i, j）}\\ 0, \text{否则}\end{cases}$$

其中，ρ为信息素残留系数，且$\rho\in(0,1)$；$\Delta\tau_{ij}(t)$为本次循环中路径(i,j)上的信息增量，$\Delta\tau_{ij}^{k}(t)$为蚂蚁 k 在本次循环中留在路径(i,j)上的信息量；Q表示信息素强度，且$Q>0$；L_k为蚂蚁 k 在本次循环中所走路径的总长度。

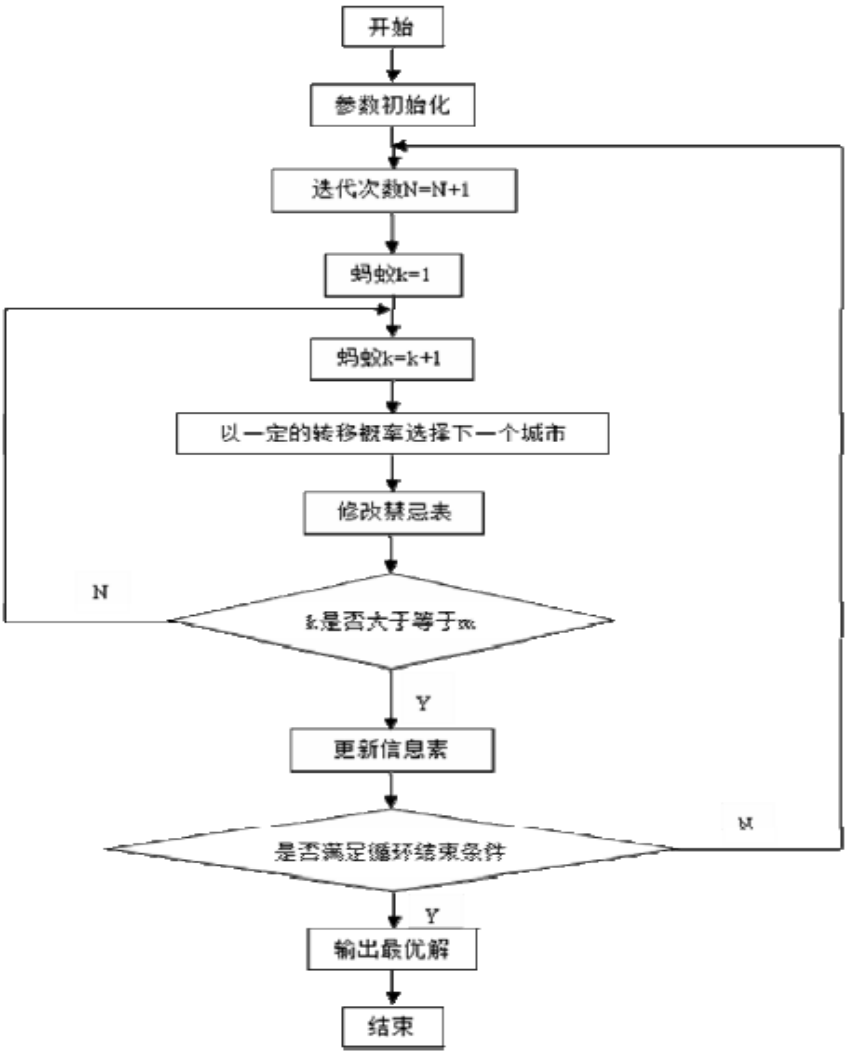

图 11. 蚁群算法基本流程图

具体实现步骤为：

(1) 算法中的所有参数初始化，即时间 $t=0$，迭代次数 $N=0$，设置最大迭代次数 ，将 m 只蚂蚁都置于 n 个城市上，蚂蚁 k=1，令初始时刻$\tau_{ij}(0)=C$，$\Delta\tau_{ij}^{k}(0)=0$ 赋予 ρ 和 Q 初值；

(2) 迭代次数 $N \leftarrow N+1$；

(3) 蚂蚁数目 $k \leftarrow k+1$；

(4) 每只蚂蚁都按照状态转移概率 去选择城市 j 并行进；

(5) 等蚂蚁选择后将蚂蚁移动到新的城市，并把该城市移到该蚂蚁的禁忌表中；

(6) 若没有走完 n 个城市，则转(3)，否则转(7)；

(7) 根据公式（11）和（12）对每条路线上的信息量进行更新；

(8) 若迭代次数 $N \geq N_{\max}$，则迭代过程结束，否则转(2)。

使用蚁群算法同样是对公式（10）所表达的非线性规划模型进行优化。具体的实现代码及说明参见附录。

用蚁群算法来求解各相位的单点信号配时优化问题。算法参数设置为：蚂蚁数量 20，迭代次数 500，蚂蚁爬行速度 0.3，信息素蒸发系数 $\rho=0.85$，信息素增加强度 Q = 0.8。在蚁群算法原理的基础上，编写蚁群算法程序（见附录三、四、五），在 MATLAB 下运行可得该问题的最优解：C1=[55.18 11.51 46.61 35.95]、C2=[67.43 27.14 18.07 20.33]，x=0.673。蚁群算法迭代过程如下列图示。

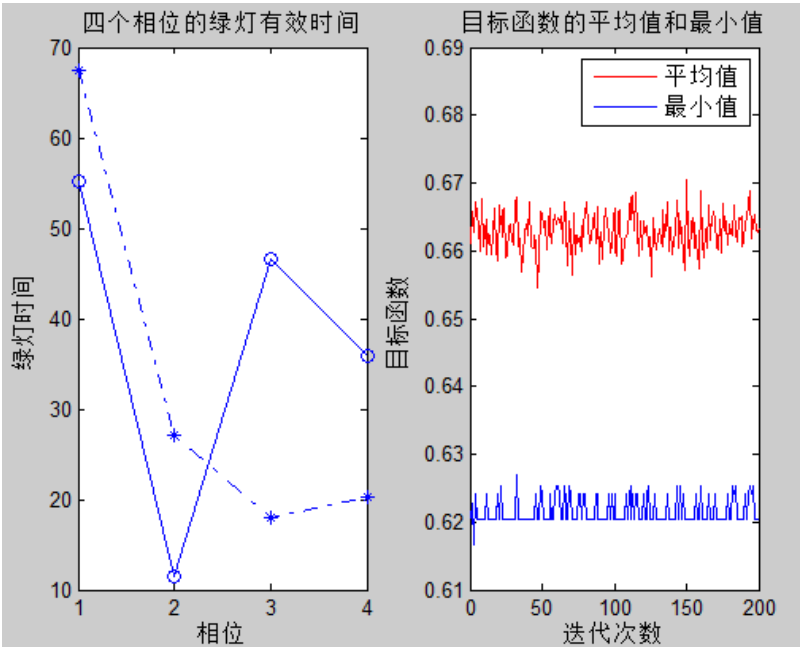

图 12. 蚁群算法迭代过程

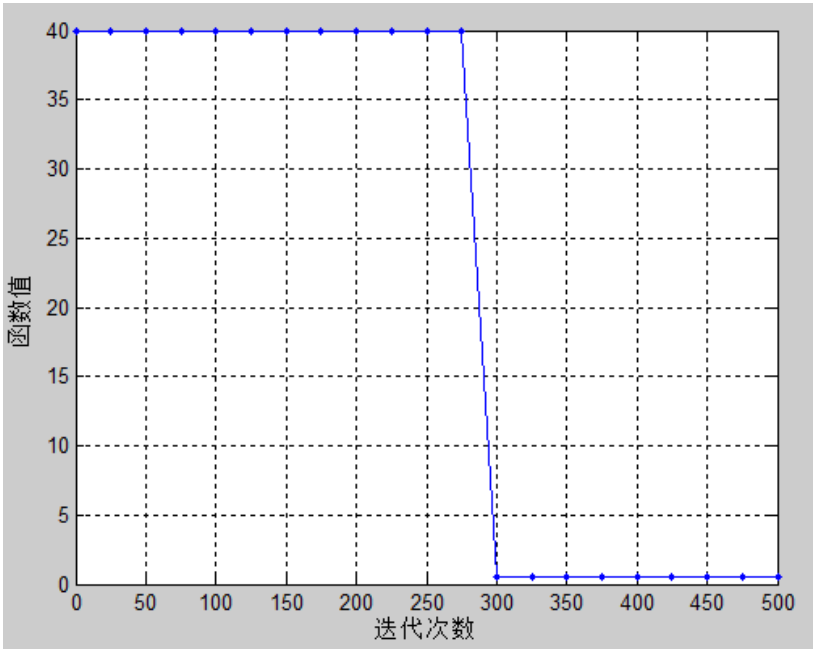

图 13. 目标函数值与迭代次数的关系

这里我们将各算法计算的结果列出以对比。

表 8. 各算法计算的一周期内车均停车时间

车均停车时间				
交叉口	Webster 方法	Synchro 仿真	遗传算法方法	蚁群算法方法
A 东	1.0632	0.1223	0.7495	0.4359
A 西	0.7827	0.0393	0.5349	0.2871
A 南	0.5901	0.0830	0.4210	0.2520
A 北	0.4305	0.0786	0.3132	0.1959
B 东	2.9484	0.0087	1.9685	0.9886
B 西	0.6178	0.2402	0.4920	0.3661
B 南	0.6451	0.0786	0.4563	0.2674
B 北	0.6137	0.1485	0.4586	0.3035

遗传算法和蚁群算法优化后的车均停车次数相比 Webster 方法都有所下降，不过相比 Synchro 仿真的结果还较大，但停车次数只是延误的一方面影响，要说明交叉口服务水平还是要看延误时间和通行能力的大小。

表 9. 各算法计算的一周期内车均延误

车均延误（s）					
交叉口	实际车均延误	Webster 方法	Synchro 仿真	遗传算法方法	蚁群算法方法
A 东	0.4671	0.4548	0.3144	0.4547	0.4307
A 西	2.7872	2.0680	0.1572	1.7098	1.2957
A 南	0.8357	0.1978	0.3144	0.3202	0.3282
A 北	1.1272	0.1561	0.3144	0.3216	0.3270
B 东	0.1888	1.1902	0.0000	0.8124	0.6011
B 西	1.1849	0.5005	1.2576	0.8714	1.0252
B 南	2.5858	1.0440	0.1572	1.0070	0.7759
B 北	2.2548	0.8066	0.4716	0.9204	0.8112

各种模型方法都在实际车均延误时间的基础上有不同程度的下降。综合来看蚁群算法优化效果比遗传算法略好，而遗传算法和蚁群算法优化效果好于 Webster 法，Synchro 仿真的车均延误时间最小，这主要是由于 Synchro 仿真对交通流的控制最理想化，并且模拟时间太短，系统并没有达到稳态。

表 10. 各算法计算的一周期内交叉口通行能力

交叉口通行能力（pcu/h）			
交叉口	Webster 方法	遗传算法方法	蚁群算法方法
A 东	1687.1971	3964.9132	3880.5533
A 西	828.5017	1946.9790	1905.5539
A 南	1428.2647	3356.4221	3285.0089
A 北	1450.8304	3409.4514	3336.9099
B 东	1466.8195	3447.0258	3373.6848
B 西	1684.0014	3957.4033	3873.2032
B 南	1275.3079	2996.9737	2933.2083
B 北	1406.9924	3306.4322	3236.0826

由表 10 可以知道，遗传算法和蚁群算法优化后的通行能力比 Webster 法大大提高。通行能力是衡量服务水平的非常重要的指标，通信能力高，说明综合的服务水平就高。而其他的因素如车均延误时间、车均停车次数等都是通过影响通行能力来影响服务水平的。

表 11. 各算法计算的一周期内有效绿灯时间

有效绿灯时间（s）			
算法	Webster 方法	遗传算法方法	蚁群算法方法
A 区相位 1	50.1651	54.1783	55.1816
A 区相位 2	14.3907	12.6638	11.5126
A 区相位 3	51.7850	55.9278	46.6065
A 区相位 4	44.9392	39.5465	35.9513
B 区相位 1	79.3374	63.4699	67.4368
B 区相位 2	33.9346	29.8624	27.1477
B 区相位 3	22.5940	19.8827	18.0752
B 区相位 4	25.4140	22.3643	20.3312

有效绿灯时间的优化可以说是配时优化的核心，从表 11 看出，实际上三种优化方法对有效绿灯时间的优化（本文中 Synchro 法的配时方案由 Webster 法给出）结果没有特别大的差异，但不同的有效时间序列却造成了表 10 中所示通行能力的巨大差异，表明交叉口的通行能力对有效绿灯时间，或者说绿信比是非常敏感的。

5.4 算法的收敛性、复杂度分析

我们主要对蚁群算法的收敛性和复杂度进行分析。

- **收敛性分析**

假设初始时刻信息素残留系数 ρ 的取值为 ρ_1，集合 $A=\{\rho_1,\rho_2,\ldots,\rho_k\}$ 满足 $\rho_1 \geq \rho_2 \geq \ldots \geq \rho_k \geq \rho_{\min}$。令优化问题的最优解为 x^*，且 $g(x^*)=\max\{Q\cdot\Delta\tau_{ij}\}$。文献[3]提出了引理 1 与定理 2，可以描述该算法的收敛性，这里不加证明地列出，感兴趣的可以参阅该文献。

引理 1：对于任意的 τ_{ij}，都存在 $\tau_{\min}$ 和 $\tau_{\max}$ 使得 $\tau_{\min} \leq \tau_{ij} \leq \tau_{\max}$，且

$$\tau_{\max} = g(x^*)/(1-\rho_1)$$

另外，假设 n 次迭代中至少有一次找到优化问题最优解的概率为 $P(n)$，能找到优化问题最优解的概率为 P，记 P 的最小值为 $P_{\min}$。则有

定理 2：对于任意小的正常数ε，如果n足够大，那么$P(n)\geq 1-\varepsilon$，且 $\lim_{n\to\infty}P(n)=1$。蚁群算法的收敛性是很好的，并且鲁棒性很强。遗传算法有时存在快速收敛和遗弃全局最优解陷入局部的问题，但蚁群算法可以较好地克服这个问题。

- **时间复杂度**

蚁群算法的复杂度[12]包括两个方面，一是所有蚂蚁迭代的次数，若有 M 只蚂蚁各迭代了 C 次，则总次数为 CM 次；二是每只蚂蚁通过构造图生成一个解的计算次数，在层状图中蚂蚁要确定通向下一层节点的每个连接的选择概率p_{ij}，当层状图的第t层节点个数为w_t时，构造一个解的计算量为$O(\sum_{t=0}^{N-1}w_t)$，因此总的计算量为$O(\sum_{t=0}^{N-1}CMw_t)$。

从时间复杂度的角度看，设n为前述的城市数目，即问题的规模，m为基本蚁群算法的蚂蚁数目，循环变量为N，最大循环次数为$N_{\max}$，则针对前面对基本蚁群算法程序结构流程的描述，可逐步分析出其时间复杂度，整理为下表：

表 12. 蚁群算法时间复杂度

步骤	操作	时间复杂度
①	初始化参数	$O(n^2+m)$
②	设置蚂蚁禁忌表	$O(m)$
③	每只蚂蚁单独构造解	$O(n^2m)$
④	解的评价和轨迹更新量的计算	$O(n^2m)$
⑤	信息素轨迹浓度的更新	$O(n^2)$
⑥	判断是否达到算法终止条件（$N\geq N_{\max}$），没有，则转到第②步	$O(nm)$
⑦	输出计算结果	$O(1)$

当n足够大时，低次幂的影响可以忽略不计，则由上表知m只蚂蚁要遍历n个元素（城市），经过N次循环，整个计算过程的时间复杂度为：

$$T(n)=O(N\cdot n^2\cdot m) \qquad (13)$$

- **空间复杂度**

基本蚁群算法在实现中需要用到的数据对其空间复杂度影响很大，这里空间指的是蚁群算法执行过程中相应的 NDTM 的读写所访问过的不同带格的总数目。基本蚁群算法中的数据主要实现两方面的功能：一是问题的描述，二是实现蚁群算法功能的辅助数据。

由于基本蚁群算法的求解通常采用有向图来描述，这里假设所求解问题的规模为 n，则需要一个 n 阶二维距离矩阵描述问题本身的特征。为了表示有向图上的信息量，需要用另外一个 n 阶二维距离来表示图上的信息素轨迹浓度。同时，在基本蚁群算法的求解过程中，为了保证矩阵不重复，需要为每只蚂蚁设定一个 n 阶一维数组的禁忌表。为了保存蚂蚁寻到的解，需要为每只蚂蚁设定一个数组。为了便于更新轨迹，需要设置每条边上的信息素更新量，其为一个二维数组。为了评价解的优劣，需要定义中间变量等等。这些都是在基本蚁群算法中所用到的一些基本存储变量. 渐进空间复杂度的定义与渐进时间复杂度的定义类似，通过对基本蚁群算法各步骤的综合分析，可得其整个计算过程的空间复杂度为：

$$S(n)=O(n^2)+O(n\bullet m) \qquad (14)$$

由式（14）可见，基本蚁群算法在数据存储上的空间复杂度是非常简单的，从而算法易于程序编制。

六、 模型评价

6.1 模型优点

- 本模型基于传统的 Webster 交通信号配时优化模型对本实例进行绿信比、车均延误、车均停车数等参数的优化，同时引入 Synchro 仿真，解决 Webster 模型在近饱和交通流量时准确性低的问题。
- Synchro 仿真虽然可以解决 Webster 模型在高交通流量时准确性低的问题，但是其不能解决绿信比这种系统参数的优化问题。基于此，我们引入了群智能算法家族中的遗传算法和蚁群算法来进行绿信比和周期这种系统配时参数优化。这两种算法都属于启发式算法，收敛性好，在寻找全局最优解时优势很大。
- 同时引入 4 种优化算法，并指出其内部的联系和优缺点互补性，所得的结果相互验证和改进。利用仿真手段和智能算法改进传统的解析性质的 Webster 法，具有一定的实践和理论指导意义。同时也具有一定的创新性。

6.2 模型缺点

- 虽然在 Synchro 仿真部分考虑了不同大小车辆的非均质交通流，但在主要的遗传和蚁群算法优化模型中，认为所有的车都大小相同。
- 缺乏对司机驾驶习惯的考虑，认为所有司机都完全遵守信号控制，没有考虑如闯红灯、随机加减速、随意变道这些因素。但在某些情况下这些因素可能会很重要。

6.3 模型改进

在一定程度上，Synchro 仿真在高流量下对 Webster 模型的补充可以称为两种模型的结合。但从全面来看，本文中所采用的 Webster 模型、遗传算法、蚁群算法三者的作用几乎是并行的，都在优化同样的目标函数，只是各自的优劣程度不同。在改进工作中可以考虑综合这三种算法，在基本结构上融合它们，形成混合型算法，提高计算效率和准确度。另外，本文研究的基础是单点信号配时优化问题，但实际生活中城市道路交通是一个很复杂的区域控制问题，还需要深入研究区域信号配时优化问题，以期对本模型做出区域控制的改进。

七、 参考文献

[1] 李成利. 单点信号交叉口配时优化理论方法与Synchro软件的结合运用[J]. 公路与汽运, 2014, (6):48-51.

[2] 何佳佳. 基于蚁群算法的交通信号配时优化[D]. 陕西西安:陕西科技大学, 2012.

[3] 张娟子. 基于人均延误最小的信号交叉口公交优先配时优化方法研究[D]. 陕西西安:长安大学, 2013.

[4] 阴炳成，杨晓光. 交叉口单点公共汽车交通优先控制方法研究[J]. 公路交通科技, 2005, 22 (12): 123-126.

[5] 张卫华，陆化普，石琴等. 公交优先的信号交叉口配时优化方法[J]. 交通运输工程学报, 2004, 4(3): 49-53.

[6] 关伟，申金升，葛芳. 公交优先的信号控制策略研究[J]. 系统工程学报, 2001, 16(3): 176-180.

[7] 邹志云, 陈绍宽等. 基于Synchro 系统的典型信号交叉口配时优化研究[J]. 北京交通大学学报, 2004, 28(6).

[8] 毛保华, 杨肇夏, 陈海波. 道路交通仿真技术与系统研究[J]. 北方交通大学学报,2002 ,26 (5) ,37 - 46.

[9] 王秋平，谭学龙. 城市道路两相邻交叉口信号控制组合优化研究[J]. 中国科技论文在线：http://www.paper.edu.cn.

[10] 王栋. 蚁群算法在交通信号实时控制中的研究及应用[D]. 北京:中国地质大学（北京）, 2009.

[11] 杨锦冬,杨东援.城市信号控制交叉口信号周期时长优化模型[J]. 同济大学学报, 2001, 29 (7): 789-794.

[12] 段海滨. 蚁群算法原理及应用[M]. 北京:科学出版社, 2005.

附录

附录一

目标函数

```
function f=f(x)
f=154681.84*(x(:,1)-0.66)./(2*x(:,1).^2*(1-0.66))+x(:,1).^2./(2*(1-x(:,1)));
```

附录二

遗传算法代码：

```
%定义遗传算法参数
NIND=1000;          %个体数目(Number of individuals)
MAXGEN=500;         %最大遗传代数(Maximum number of generations)
NVAR=2;             %变量个数
PRECI=20;           %变量的二进制位数(Precision of variables)
GGAP=0.9;           %代沟(Generation gap)
trace=ones(MAXGEN+1,2);%寻优结果的初始值
FieldD=[rep([PRECI],[1,NVAR]);[40,0.6;120,0.8]; rep([1;0;1;1],[1,NVAR])];% 区域描述器(Build field descriptor)
Chrom=crtbp(NIND,NVAR*PRECI); %初始种群
v=bs2rv(Chrom,FieldD); %计算初始种群的十进制转换
gen=1;  %代计数器
kk=100;
while kk >= 4
    [NIND,N]=size(Chrom);
    M=fix(NIND/2);
    ObjvV=f(v(1:M,:));
    FitnV=ranking(ObjvV);%分配适应度值(Assign fitness values)
    SelCh=select('sus',Chrom(1:M,:),FitnV,GGAP);%选择
    SelCh=recombin('xovsp',SelCh,0.9);%重组
    Chrom=mut(SelCh);%变异
    kk=size(Chrom);
    v=bs2rv(Chrom,FieldD);%子代个体的十进制转换
    trace(gen,1)=min(f(v));
```

```
    trace(gen,2)=sum(f(v))/length(f(v));
    gen=gen+1;
end
figure();clf;
plot(trace(:,1));
hold on;
plot(trace(:,2),'-.');
plot(trace(:,1),'.');
plot(trace(:,2),'.');
grid;
legend('目标函数解','目标函数解的平均值')
xlabel('迭代次数');ylabel('目标函数值');
```

附录三

交叉口 A 蚁群算法代码

```
% 主程序。
% 作用： 对蚁群算法的相关系数或参数进行设置后，调用蚁群算法的源
% 程序, 并进行绘图。
% 主要符号说明
% Ni_max 最大迭代次数
% m 蚂蚁个数
% Alpha 表征信息素重要程度的参数
% Beta 表征启发式因子重要程度的参数
% Rho 信息素蒸发系数
% Q 信息素增加强度系数
% Road_best 各代最佳路线
% Fx_best 各代最佳路线的长度
%MainFunction_Pioneer
%function[Road_best,Fx_best,Fx_ave,BestGreenTime,BestDestFun]=CrossOneACA(
X,Ni_max,m,Alpha,Beta,Rho,Q)
%X 有效绿灯时间可取值的范围
Ni_max = 200;
Alpha = 1.7;
Beta = 5;
%Garma = 5;
```

```
Rho = 0.9;
Q = 100;
m = 25;
X = 15:50; %绿灯时间的范围
[Road_best,Fx_best,Fx_ave,Times,FunResult]=CrossOneACA(X,Ni_max,m,Alpha,B
eta,Rho,Q);
CrossOneACA_Signal(Times);
Times
FunResult




% 被调蚁群函数
% 作用：  用蚁群算法进行计算的函数，对相关过程进行调控和迭代，
% 模拟蚂蚁的实际爬行情况，进而求解得出一个相对最优解。
function[Road_best,Fx_best,Fx_ave,BestGreenTime,BestDestFun]=CrossOneACA(X
,Ni_max,m,Alpha,Beta,Rho,Q)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
% 主要符号说明
% X 绿灯信号的可选序列
% Ni_max  最大迭代次数
% m 蚂蚁个数
% Alpha  表征信息素重要程度的参数
% Beta 表征启发式因子重要程度的参数
% Rho 信息素蒸发系数
% Q 信息素增加强度系数
% Road_best 各代最佳路线
% Fx_best 各代最佳路线的长度
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
% Garma = 5;
MaxTlight = 200;
%%===Step1：变量初始化
n=size(X,2);%n  表示问题的规模（点的个数）
```

```
D=zeros(n,n);%D 表示赋权邻接矩阵
for i=1:n
    for j=1:n
        if i~=j
            D(i,j)=abs(X(i)-X(j));
        else
            D(i,j)=eps;
            %i=j 时不计算，为 0，后面启发因子要取倒数，eps（浮点相对精度）表示
        end
        D(j,i)=D(i,j);
    end
end
%Eta=1./D; %Eta 为启发因子矩阵，这里设为距离的倒数
AntInfo=ones(n,n); %AntInfo 为信息素矩阵
Eta = ones(n,n);
AntRun=zeros(m,4);
%存储并记录路径的生成 m 只蚂蚁，每只走 n 中的 4 个点为一条路
Ni=1; %迭代计数器，记录迭代次数
Road_best=zeros(Ni_max,4); %各次迭代的最佳路线
Fx_best=inf.*ones(Ni_max,1);% 各次迭代的最小目标函数值
%Length_best=inf.*ones(Ni_max,1);% 各代最佳路线的长度
Fx_ave=zeros(Ni_max,1); %各次迭代的目标函数平均值（m 只蚂蚁的平均）
while Ni<=Ni_max %停止条件之一：达到最大迭代次数，停止
    %%===Step2：将 m 只蚂蚁放到 n 个点上
    Randpos=[]; %0*0 的矩阵 一随机存取
    for i=1:(ceil(m/n))
        Randpos=[Randpos,randperm(n)];
        %randperm-随机生成 1－n 的序列
    end %Randpos 为一组 ceil(m/n) * n 长的序列
    AntRun(:,1)=(Randpos(1,1:m))'; %取随机序列的前 m 个值
    %将 m 只蚂蚁随机分布在 n 个点上
    %%===Step3：m 只蚂蚁按概率函数选择下一个点，完成各自的周游
    for j=2:4 %所在点不计算
        for i=1:m %对于每只蚂蚁
```

```
            P=zeros(1,n); %待访问点的选择概率分布
            %下面计算待选点的概率分布
            for k=1:n
                %信息素越多，距离越小，选择的概率就越大
                P(k)=( AntInfo( AntRun(i,j-1),k )^Alpha )*(Eta( AntRun(i,j-
1),k )^Beta );
                %选择下一个点的可能性
            end
            P=P/(sum(P));
            Pcum=cumsum(P); %cumsum，元素累加和 -数组
            Select=find(Pcum>=rand); %rand-0 到 1 的随机数
            %若计算的概率大于原来的就选择这条路线
            if size(Select,1)==0
                Select = [1];
            end
            to_visit=Select(1);
            AntRun(i,j)=to_visit; %确定第 i 只蚂蚁的第 j 个点
        end
    end
    %===Step4：记录本次迭代最佳路线
    F=zeros(m,1);%m*1 的列向量，m 只蚂蚁，每只所走序列的目标函数值
    for i=1:m
        Gx=AntRun(i,:); %对于蚂蚁 i 得到的序列
        if sum(Gx) + (X(1)-1)*4 > MaxTlight
            F(i) = inf;
        else
            F(i) = CrossOneACA_Signal(Gx+X(1)-1 );
        end
    end
    Fx_best(Ni)=min(F); %第 Ni 次迭代得到的优化值取最小
    posAnt = find( F==Fx_best(Ni) );%找出所有最佳路线的蚂蚁序号
    Road_best(Ni,:) = AntRun(posAnt(1),:);
    %此轮迭代后的一只蚂蚁的最佳路线
    Fx_ave(Ni)=mean(F); %此轮迭代后的平均距离
    %===Step5：更新信息素
```

```
    Delta_AntInfo=zeros(n,n); %开始时信息素为 n*n 的 0 矩阵
    for i=1:m
        for j=1:3

Delta_AntInfo(AntRun(i,j),AntRun(i,j+1))=Delta_AntInfo(AntRun(i,j),AntRun(i,j+1)
)+Q/F(i);
                %此次循环在路径（i，j）上的信息素增量
        end %Delta_AntInfo(AntRun(i,4),AntRun(i,1))=
        Delta_AntInfo(AntRun(i,4),AntRun(i,1))+Q/F(i);
        %此次循环在整个路径上的信息素增量
    end
    AntInfo=(1-Rho).*AntInfo+Delta_AntInfo;
    %考虑信息素挥发后，再增强更新后的信息素
    %===Step6：蚂蚁搜索表清零
    AntRun=zeros(m,4); %%每次迭代后清 0
    if mod(Ni,10)==0
        Ni %打印迭代次数
    end
    Ni=Ni+1; %迭代继续
end %(while)
%%===Step7：输出结果
Pos=find(Fx_best==min(Fx_best)); %找到最佳路径（非 0 为真）
BestGreenTime = Road_best(Pos(1),:)+14 %最大迭代次数后最佳路径序列
BestDestFun = Fx_best(Pos(1)) %最大迭代次数后最优目标函数值
subplot(1,2,1) %绘制第一个子图形
%ACATSPdrawRoute(C,Shortest_Route) %画路线图的子函数
t=1:4;
plot(t,BestGreenTime,'o-')
xlabel('相位');
ylabel('绿灯时间');
title('四个相位的绿灯有效时间'); %标题
subplot(1,2,2) %绘制第二个子图形
plot(Fx_ave,'r');
hold on %保持图形
plot(Fx_best);
```

```
xlabel('迭代次数');
ylabel('目标函数');
legend('平均值','最小值');
title('目标函数的平均值和最小值'); %标题



% 目标函数
% 将各个参数代入模型中的公式后，对目标函数值进行计算，
% 进而针对每条%路径求出一个目标函数值。
function ans = CrossOneACA_Signal(x)
n =size(x,2);
L = 3 * 4; % 黄灯时间
c = sum(x) + L; %周期=有效绿灯时间+黄灯时间
s = [2800 2800 1800 1800]; %四个饱和流量
q = [1858 533 1233 1070];%相位流量
%y = [0.66 0.19 0.68 0.59]; %流量比，交叉口流率比 Y 为四个相位之和
y = q./s;
g = x/c;%绿信比
%Y = max(y(1:2)) + max(y(3:4));
Y = sum(y);
%- 第 i 相位车辆平均延误时间
D = ((c - x).^2)./(2*c*(1-y)) + (c-x)./(2*c*q)+(q*c*c)./(2*s.*x.*(s.*x-q*c)) ;
k1 = 2*(1-Y).*(s.^(1/7)) ;
%- 第 i 相位平均停车次数
H = 0.9 * (1-g)./(1-y) ;
k2 =(1-Y)/0.9*(s.^(1/7));
%- 第 i 相位交叉口通行能力
Q = s.*x/c;
k3 = 2*Y*c/3600;
ans = ( sum(D.*k1) + sum(H.*k2) ) / ( sum(k3*Q) );
```

附录四

交叉口 B 蚁群算法代码

```
% 主程序。
% 作用： 对蚁群算法的相关系数或参数进行设置后，调用蚁群算法的源
% 程序, 并进行绘图。
% 主要符号说明
% Ni_max 最大迭代次数
% m 蚂蚁个数
% Alpha 表征信息素重要程度的参数
% Beta 表征启发式因子重要程度的参数
% Rho 信息素蒸发系数
% Q 信息素增加强度系数
% Road_best 各代最佳路线
% Fx_best 各代最佳路线的长度
%MainFunction_Pioneer
%function[Road_best,Fx_best,Fx_ave,BestGreenTime,BestDestFun]=CrossOneACA(
X,Ni_max,m,Alpha,Beta,Rho,Q)
%X 有效绿灯时间可取值的范围
Ni_max = 200;
Alpha = 1.7;
Beta = 5;
%Garma = 5;
Rho = 0.9;
Q = 100;
m = 25;
X = 15:50; %绿灯时间的范围
[Road_best,Fx_best,Fx_ave,Times,FunResult]=CrossOneACA(X,Ni_max,m,Alpha,B
eta,Rho,Q);
CrossOneACA_Signal(Times);
Times
FunResult



% 被调蚁群函数
% 作用： 用蚁群算法进行计算的函数，对相关过程进行调控和迭代，
```

```
% 模拟蚂蚁的实际爬行情况，进而求解得出一个相对最优解。
function[Road_best,Fx_best,Fx_ave,BestGreenTime,BestDestFun]=CrossOneACA(X
,Ni_max,m,Alpha,Beta,Rho,Q)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 主要符号说明
% X 绿灯信号的可选序列
% Ni_max 最大迭代次数
% m 蚂蚁个数
% Alpha 表征信息素重要程度的参数
% Beta 表征启发式因子重要程度的参数
% Rho 信息素蒸发系数
% Q 信息素增加强度系数
% Road_best 各代最佳路线
% Fx_best 各代最佳路线的长度
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Garma = 5;
MaxTlight = 200;
%%===Step1：变量初始化
n=size(X,2);%n 表示问题的规模（点的个数）
D=zeros(n,n);%D 表示赋权邻接矩阵
for i=1:n
    for j=1:n
        if i~=j
            D(i,j)=abs(X(i)-X(j));
        else
            D(i,j)=eps;
            %i=j 时不计算，为 0，后面启发因子要取倒数，eps（浮点相对精
度）表示
        end
        D(j,i)=D(i,j);
    end
end
%Eta=1./D; %Eta 为启发因子矩阵，这里设为距离的倒数
```

```
AntInfo=ones(n,n); %AntInfo 为信息素矩阵
Eta = ones(n,n);
AntRun=zeros(m,4);
%存储并记录路径的生成 m 只蚂蚁，每只走 n 中的 4 个点为一条路
Ni=1; %迭代计数器，记录迭代次数
Road_best=zeros(Ni_max,4); %各次迭代的最佳路线
Fx_best=inf.*ones(Ni_max,1);% 各次迭代的最小目标函数值
%Length_best=inf.*ones(Ni_max,1);% 各代最佳路线的长度
Fx_ave=zeros(Ni_max,1); %各次迭代的目标函数平均值（m 只蚂蚁的平均）
while Ni<=Ni_max %停止条件之一：达到最大迭代次数，停止
    %%===Step2：将 m 只蚂蚁放到 n 个点上
    Randpos=[]; %0*0 的矩阵 一随机存取
    for i=1:(ceil(m/n))
        Randpos=[Randpos,randperm(n)];
        %randperm-随机生成 1－n 的序列
    end %Randpos 为一组 ceil(m/n) * n 长的序列
    AntRun(:,1)=(Randpos(1,1:m))'; %取随机序列的前 m 个值
    %将 m 只蚂蚁随机分布在 n 个点上
    %%===Step3：m 只蚂蚁按概率函数选择下一个点，完成各自的周游
    for j=2:4 %所在点不计算
        for i=1:m %对于每只蚂蚁
            P=zeros(1,n); %待访问点的选择概率分布
            %下面计算待选点的概率分布
            for k=1:n
                %信息素越多，距离越小，选择的概率就越大
                P(k)=( AntInfo( AntRun(i,j-1),k )^Alpha )*(Eta( AntRun(i,j-
1),k )^Beta );
                %选择下一个点的可能性
            end
            P=P/(sum(P));
            Pcum=cumsum(P); %cumsum，元素累加和 -数组
            Select=find(Pcum>=rand); %rand-0 到 1 的随机数
            %若计算的概率大于原来的就选择这条路线
            if size(Select,1)==0
                Select = [1];
```

```
            end
            to_visit=Select(1);
            AntRun(i,j)=to_visit; %确定第 i 只蚂蚁的第 j 个点
        end
    end
    %===Step4：记录本次迭代最佳路线
    F=zeros(m,1);%m*1 的列向量，m 只蚂蚁，每只所走序列的目标函数值
    for i=1:m
        Gx=AntRun(i,:); %对于蚂蚁 i 得到的序列
        if sum(Gx) + (X(1)-1)*4 > MaxTlight
            F(i) = inf;
        else
            F(i) = CrossOneACA_Signal(Gx+X(1)-1 );
        end
    end
    Fx_best(Ni)=min(F); %第 Ni 次迭代得到的优化值取最小
    posAnt = find( F==Fx_best(Ni) );%找出所有最佳路线的蚂蚁序号
    Road_best(Ni,:) = AntRun(posAnt(1),:);
    %此轮迭代后的一只蚂蚁的最佳路线
    Fx_ave(Ni)=mean(F); %此轮迭代后的平均距离
    %===Step5：更新信息素
    Delta_AntInfo=zeros(n,n); %开始时信息素为 n*n 的 0 矩阵
    for i=1:m
        for j=1:3

Delta_AntInfo(AntRun(i,j),AntRun(i,j+1))=Delta_AntInfo(AntRun(i,j),AntRun(i,j+1)
)+Q/F(i);
            %此次循环在路径（i，j）上的信息素增量
        end %Delta_AntInfo(AntRun(i,4),AntRun(i,1))=
        Delta_AntInfo(AntRun(i,4),AntRun(i,1))+Q/F(i);
        %此次循环在整个路径上的信息素增量
    end
    AntInfo=(1-Rho).*AntInfo+Delta_AntInfo;
    %考虑信息素挥发后，再增强更新后的信息素
    %===Step6：蚂蚁搜索表清零
```

```
    AntRun=zeros(m,4); %%每次迭代后清 0
    if mod(Ni,10)==0
        Ni %打印迭代次数
    end
    Ni=Ni+1; %迭代继续
end %(while)
%%===Step7：输出结果
Pos=find(Fx_best==min(Fx_best)); %找到最佳路径（非 0 为真）
BestGreenTime = Road_best(Pos(1),:)+14 %最大迭代次数后最佳路径序列
BestDestFun = Fx_best(Pos(1)) %最大迭代次数后最优目标函数值
subplot(1,2,1) %绘制第一个子图形
%ACATSPdrawRoute(C,Shortest_Route) %画路线图的子函数
t=1:4;
plot(t,BestGreenTime,'o-')
xlabel('相位');
ylabel('绿灯时间');
title('四个相位的绿灯有效时间'); %标题
subplot(1,2,2) %绘制第二个子图形
plot(Fx_ave,'r');
hold on %保持图形
plot(Fx_best);
xlabel('迭代次数');
ylabel('目标函数');
legend('平均值','最小值');
title('目标函数的平均值和最小值'); %标题



% 目标函数
% 将各个参数代入模型中的公式后，对目标函数值进行计算，
% 进而针对每条%路径求出一个目标函数值。
function ans = CrossOneACA_Signal(x)
n =size(x,2);
L = 3 * 4; % 黄灯时间
c = sum(x) + L; %周期=有效绿灯时间+黄灯时间
```

```
s = [3000 3000 2800 2800]; %四个饱和流量
q = [2532 1083 673 757];%相位流量
%y = [0.84 0.36 0.24 0.27]; %流量比，交叉口流率比 Y 为四个相位之和
y = q./s;
g = x/c;%绿信比
%Y = max(y(1:2)) + max(y(3:4));
Y = sum(y);
%- 第 i 相位车辆平均延误时间
D = ((c - x).^2)./(2*c*(1-y)) + (c-x)./(2*c*q)+(q*c*c)./(2*s.*x.*(s.*x-q*c)) ;
k1 = 2*(1-Y).*(s.^(1/7)) ;
%- 第 i 相位平均停车次数
H = 0.9 * (1-g)./(1-y) ;
k2 =(1-Y)/0.9*(s.^(1/7));
%- 第 i 相位交叉口通行能力
Q = s.*x/c;
k3 = 2*Y*c/3600;
ans = ( sum(D.*k1) + sum(H.*k2) ) / ( sum(k3*Q) );
```

附录五

作为整体的蚁群算法代码

```
function [BESTX,BESTY,ALLX,ALLY]=ACOUCP(K,N,Rho,Q,Lambda,LB,UB)
K=500; % 迭代次数
N=20; % 蚁群规模
Rho=0.85; % 信息素蒸发系数
Q=0.80; % 信息素增加强度
Lambda=0.3; % 蚂蚁爬行速度
LB=[40,0.6]' ;% 决策变量的下界
UB =[120,0.8]'; % 决策变量的上界
%第一步：初始化
M=length(LB); %决策变量的个数
% 蚁群位置初始化
X=zeros(M,N);
for i=1:M
```

```
    x=unifrnd(LB(i),UB(i),1,N);
    X(i,:)=x;
end
% 输出变量初始化
ALLX=cell(K,1); % 细胞结构，记录每一代的个体
ALLY=zeros(K,N); % K×N 矩阵，记录每一代评价函数值
BESTX=cell(K,1); % 细胞结构，记录每一代的最优个体
BESTY=ones(K,1); % K×1 矩阵，记录每一代的最优个体的评价函数值
k=1; % 迭代计数器初始化
Tau=zeros(1,N); % 信息素初始化
Y=zeros(1,N); % 适应值初始化
%第二步：迭代过程
while k<=K
    YY=zeros(1,N);
    for n=1:N
        x=X(:,n);
        YY(n)=FIT(x);
    end
    maxYY=min(YY);
    temppos=find(YY==maxYY);
    POS=temppos(1);
    % 蚂蚁随机探路
    for n=1:N
        if n~=POS
            x=X(:,n);
            Fx=FIT(x)
            mx=GaussMutation(x,LB,UB)
            Fmx=FIT(mx)
            if Fmx<Fx
                X(:,n)=mx;
                Y(n)=Fmx;
            elseif rand>1-(1/(sqrt(k)))
                X(:,n)=mx;
                Y(n)=Fmx;
            else
```

```
                X(:,n)=x;
                Y(n)=Fx;
            end
        end
    end
    for n=1:N
        if n~=POS
            x=X(:,n);
            Fx=FIT(x);
            mx=GaussMutation(x,LB,UB);
            Fmx=FIT(mx);
            if Fmx<Fx
                Y(n)=Fmx;
            else if rand>1-(1/(sqrt(k)))
                    X(:,n)=mx;
                    Y(n)=Fmx;
                else
                    X(:,n)=x;
                    Y(n)=Fx;
                end
            end
        end
    end
    % 朝信息素最大的地方移动
    for n=1:N
        if n~=POS
            x=X(:,n);
            r=(K+k)/(K+K);
            p=randperm(N);
            t=ceil(r*N)
            pos=p(1:t)
            TempTau=Tau(pos)
            maxTempTau=max(TempTau)
            pos2=find(TempTau==maxTempTau)
            pos3=pos(pos2(1))
```

```
            x2=X(:,pos3(1))
            x3=(1-Lambda)*x+Lambda*x2
            Fx=FIT(x);
            Fx3=FIT(mx);
            if Fx3<Fx
                X(:,n)=x3;
                Y(n)=Fx3;
            else if rand>1-(1/(sqrt(k)))
                    X(:,n)=x3;
                    Y(n)=Fx3;
                else
                    X(:,n)=x;
                    Y(n)=Fx;
                end
            end
        end
    end
    % 更新信息素并记录
    Tau=Tau*(1-Rho)
    maxY=max(Y)
    minY=min(Y)
    DeltaTau=(maxY-Y)/(maxY-minY);
    Tau=Tau+Q*DeltaTau
    ALLX{k}=X;
    ALLY(k,:)=Y;
    minY=min(Y);
    pos4=find(Y==minY);
    BESTX{k}=X(:,pos4(1));
    BESTY(k)=minY;
    disp(k);
    k=k+1;
end

% 交叉变异
    function mx=GaussMutation(x,LB,UB)
```

```
            aph=max(rand(),0.618);
            mx=LB+aph*(UB-LB);
            % 绘图
            plot(BESTY,'-
ko','MarkerEdgeColor','k','MarkerFaceColor','k','MarkerSize',2)
            ylabel('目标函数值')
            xlabel('迭代次数')
            grid on
        end
end
```