基于遗传算法的城市两相邻交叉口协调控制优化方案

摘要

本文针对合理设计城市相邻两交叉口信号配时优化方案问题，我们将问题转化为算法与模型两个小问题，运用协调控制系统、理论分析、对比分析、综合分析法构建相邻两交叉口协调控制模型，以遗传算法为基础，以车辆趋于停车线处的平均延误时间、有效绿灯时间以及实际平面交叉口的通行能力为指标，运用 EXCEL、MATLAB 等软件编程，并用 VISSIM 软件进行仿真检验。最后，根据题中的基础数据，对城市两相邻交叉口信号配时问题提出协调控制方案，有效指导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，改善城市交通现状。

针对问题一，从时间复杂度、空间复杂度、收敛性对算法进行对比分析。我们对单点定时交通信号灯控制及其算法进行简要说明，在此基础上，提出智能控制中的遗传算法，在时间复杂度、空间复杂度、收敛性上对这两种算法进行对比分析。分析说明：遗传算法是一种运行效率较高，且有较高的计算精度，高效的全局寻优算法。

针对问题二，设计模型对交通信号进行配时优化研究，求解方案并进行仿真检验。首先，以协调控制系统中的主线进口道的流率模式为基础，得出以总延误时间为目标的目标函数，其次，以遗传算法为基础，构建相邻两交叉口协调控制模型，给出城市相邻两交叉口协调控制的交通信号配时优化方案，最后，运用 VISSIM 软件进行仿真检验，检验我们的研究成果。

本文后续对模型进行了误差分析，还对总延误时间与周期时长做了灵敏度分析。最后，指出了模型的特点与创新性。

关健词：交通延误；遗传算法；协调控制系统；灵敏度分析；MATLAB

目　录

§1 问题的重述

一、背景知识

1.问题概况

经过30多年的改革开放，我国城市化速度的加快以及城市规模的也在不断扩大，城市人口和车辆不断增加，城市交通需求不断增加，国内许多城市出入口地段交通状况恶化，进出城时间延长、交通堵塞、道路通行能力差、交通安全没保障和环境污染问题都愈来愈严重，成为制约城市交通系统发展的又一个瓶颈，城市的交通面临着越来越艰巨的任务。

2.问题的原因

城市交通拥堵可以从三个层面来说：①从宏观层面来说，城市交通拥堵形成的根本原因在于交通供给与交通需求之间的不平衡关系；②从中观层面来说，城市交通拥堵源于交通管理水平与城市交通快速发展间的不匹配关系；③从微观层面来说，城市交通拥堵的原因包括两个方面：一是车流量突然增大，形成交通瓶颈，这种情况多发生在上下班节假日高峰时段；二是道路突发事件造成道路交通容量的减小或吸引过多的交通容量引起的交通拥堵，如交通事故或大型活动等事件的发生。

3.现状与对策

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。欧美等发达国家大城市对城市交通的研究较早，无论在理论上还是在实践应用中都取得了优秀的成果，总结国外拥堵治理经验，主要包括以下几个方面[1]：一是鼓励和发展公共交通；二是加快城市交通基础配套建设，建立智能交通系统，提高交通管理水平；三是合理规划城市布局，从根本上降低出行率。

二、相关数据

交通数据（详见附录一）

三、要解决的问题

1.问题一：通过各种综合文献资料的查询，给出延误时长、通行能力、车辆停车次数这三个评价指标的计算公式，并基于这三个指标对单点定时交通信号灯控制进行介绍分析，以及从时间复杂度、空间复杂度、收敛性三各方面对单点定时交通信号控制所运用的算法和遗传算法这两种算法进行对比分析。

2.问题二：为了改善城市交通，加强道路通行效果，以武汉市某两个相邻的交叉口 AB为例，根据提供的已知AB交叉口的交通数据，从三个评价指标入手，对交通信号进行配时优化研究， 求解出改善后的交通信 号配时方案并进行仿真检验。

§2 问题的分析

一、问题的总分析

本文的重点在于通过了解影响交通效率的各个指标，来研究分析单电定时交通信号控制系统的具体内容和其实际效果，以及从时间复杂度、空间复杂度、收

敛性三个方面对使 用的算法进行研究分析，除此之外还有对交通信号配时优化并进行仿真检验；文章中使用了理论与实际相结合法，对比分析法，定量计量法，综合分析法、遗传算法等方法，在已知的交通数据的基础上通过给出的多种评价指标的计算公式仿真得出交通现状，针对现状加入遗传算法的应用建立相邻两交叉口协调控制模型，对交通信号控制配时优化。

二、对具体问题的分析

1.对问题一的分析

针对问题一，通过查询资料文献得出四项指标的计算公式，结合交通数据和指标公式呈现交通现状及对其进行分析解说，简要介绍了单点定时交通信号控制模型，并从时间复杂度、空间复杂度、收敛性对算法进行对比分析。

2.对问题二的分析

针对问题二，为提高实际交叉口的通行效果，减少延误改善城市交通现状，对交通信号进行配时优化研究，求解出改善后的交通信号配时方案并进行仿真检验。本文基于遗传算法的运用加上城市相邻交叉口协调控制方法的是哦那个，建立相关模型，为该说你城市交通提出切实可行的方案。

§3 模型的假设

1.对于系统的外部进口道，假定车辆从外进口道进入干道协调系统内部是随机的，其延误可根据经典的 Webster 延误计算方法进行计算；

2.对于干道协调系统内部进口道，由于各交叉口相互距离不远，主线方向的交通流因受上游交叉口信号的影响已不再是随机的。因此在延误计算中可不考虑由交通流随机超饱和造成的随机延误；

3.简化起见，不考虑车辆排队长度的影响，认为停车都发生在停车线处；

4.干道协调系统内部的交通流为非饱和流，这是绿波控制得以实施的前提；

5.排队车辆平均分配到该行驶方向的车道上；

6.黄灯时，车辆不允许通过。

§4 名词解释与符号说明

一、名词解释

1.相位：即信号相位，是指在周期时间内按需求人为设定的，同时取得通行权的一个或几个交通流的序列组。

2.遗传算法：是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。

3.时间复杂度：在计算机科学中，算法的时间复杂度是一个函数，它定量描述了该算法的运行时间。这是一个关于代表算法输入值的字符串的长度的函数。

4.空间复杂度：是对一个算法在运行过程中临时占用存储空间大小的量度。

二、符号说明

序号	符号	符号说明
1	d_{ij}^{r}	为第r间隔由i向到j向车辆平均延误（s）

2	N_{ij}^{r}	为第r间隔单位时间平均滞留车辆数
3	h_{ij}^{r}	为第r间隔由i问到j问车辆平均停车次数
4	c_{ij}	为由i向到j向车道通行能力；
5	$q^{d}(j)$	为第j个时段，下游断面d上预计的车辆到达率
6	F	为车辆离散系数
7	C_{O}	最佳周期长度
8	L	总损失时间
9	g_{i}	第i相位的绿灯时间

§5 模型的建立与求解

一、问题一的分析与求解

1.问题的分析

针对问题一，从时间复杂度、空间复杂度、收敛性对算法进行对比分析。本文首先对单点定时交通信号灯控制模型及其所运用的算法做出简要说明，其次采用非线性优化算法—遗传算法进行线控信号配时优化计算，它对于非线性规划问题有较高的计算精度，适用于复杂数学模型的求解，最后从时间复杂度、空间复杂度、收敛性对这两种算法进行对比分析。

单点定时交通信号灯控制

定时控制是实际交通中可以是实现的最基本的控制方式，也是使用最广泛的一种控制方式。在定时控制中，所有的控制参数，例如：延误时长、通行能力、车辆停车次数、绿信比、排队长度……这些参数都是事先根据交叉路口一定时间的交通流量数据确定。

①相关指标公式[2]

$$d_{ij}^{r}=\frac{C(1-\lambda_{k})^{2}}{2(1-\lambda_{ij}^{r})}+\frac{N_{ij}^{r}x_{ij}^{r}}{q_{ij}^{r}} \qquad (1)$$

式中，d_{ij}^{r}为第r间隔由i向到j向车辆平均延误（s）；C为周期时长（s）；

λ_{k}为k相位绿信比，$\lambda_{k}=\frac{g_{k}}{C}$，$g_{k}$为k相位有效绿灯时间；

q_{ij}^{r}为第r间隔从i向到j向流量(pcu/s)；N_{ij}^{r}为第r间隔单位时间平均滞留车辆数

x_{ij}^{r}为第r间隔从i向到j向饱和度，$x_{ij}^{r}=q_{ij}^{r}/c_{ij}$，$c_{ij}$为由$i$向到$j$向车道通行能力；

y_{ij}^{r}为流量比率，$y_{ij}^{r}=q_{ij}^{r}/s_{ij}$，$s_{ij}$为该方向饱和流量$(pcu/s)$；

$$N_{ij}^{r}=\begin{cases}\dfrac{c_{ij}}{4}\left[(x_{ij}^{r}-1)+\sqrt{(x_{ij}^{r}-1)^{2}+\dfrac{12(x_{ij}^{r}-x_{ijo})}{c_{ij}}}\right] & x_{ij}>x_{ijo}\\ 0 & x_{ij}\le x_{ijo}\end{cases}\qquad(2)$$

式中 $x_{ijo}=0.67+\dfrac{s_{ij}g_{k}}{600}$。则第 r 间隔所有流向总流量 $q^{r}=\sum_{j=1}^{n-1}\sum_{i=1}^{n}q_{ij}^{r}$；

第 r 间隔车辆平均延误 $d^{r}=\dfrac{(\sum_{j=1}^{n-1}\sum_{i=1}^{n}d_{ij}^{r}q_{ij}^{r})}{q^{r}}$；

时段 T 车辆平均延误 $\bar{d}=\dfrac{\sum_{r=1}^{w}d^{r}q^{r}}{\sum_{r=1}^{w}q^{r}}$

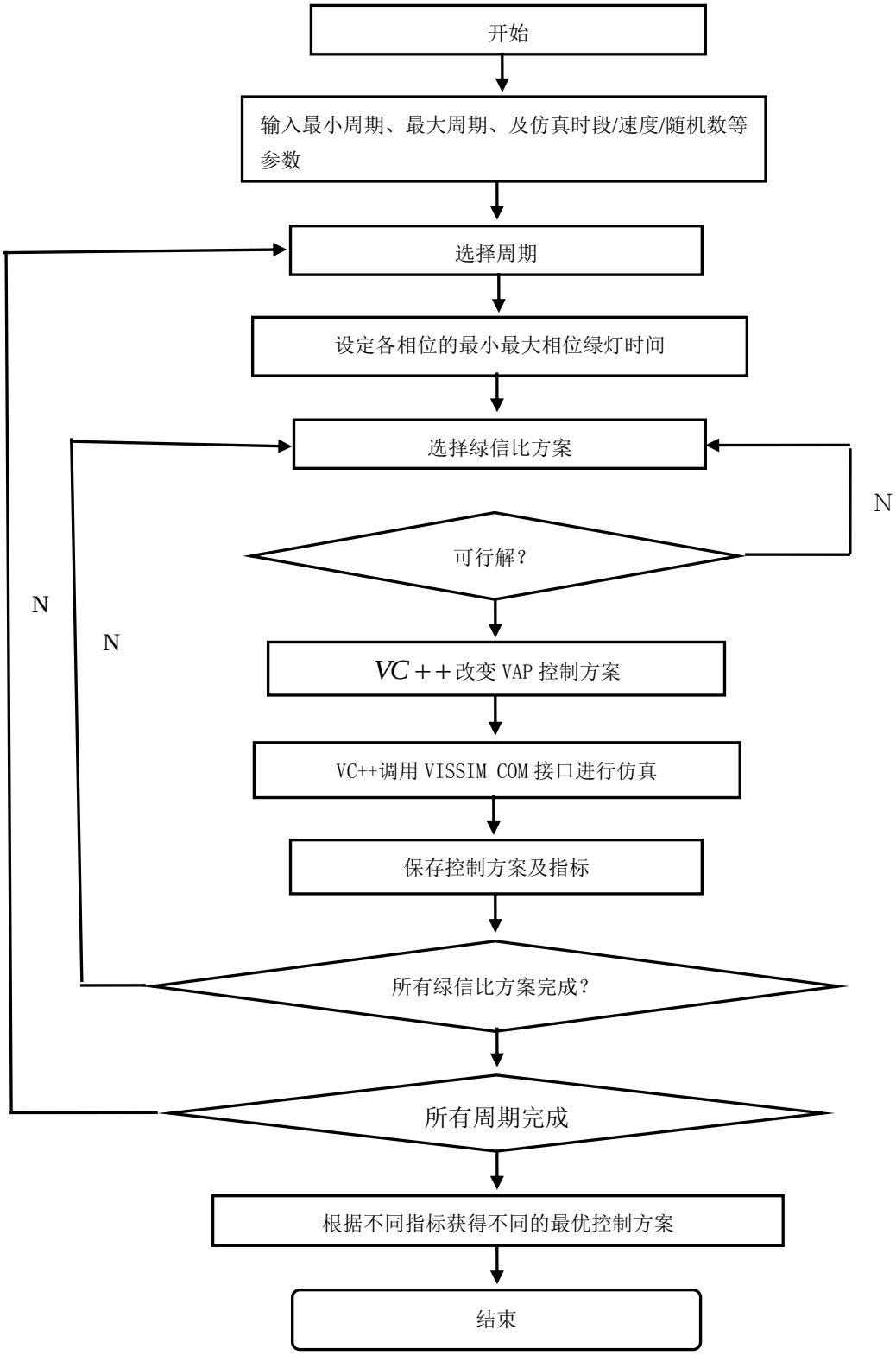
交叉口通行能力 $c_{ij}=s_{ij}\times\lambda_{k}$，则总通行能力 $c=\sum_{j=1}^{n=1}\sum_{i=1}^{n}c_{ij}$　　（3）

$$h_{ij}^{r}=f\left(\frac{1-\lambda_{k}}{1-y_{ij}^{r}}+\frac{N_{ij}^{r}}{q_{ij}^{r}C}\right)\qquad(4)$$

式中：其中，h_{ij}^{r} 为第 r 间隔由 i 问到 j 问车辆平均停车次数；f 为考虑部分车辆只经历不完全停车的情况，对停车率采用的矫正系数，通用取 $f=0.9$。

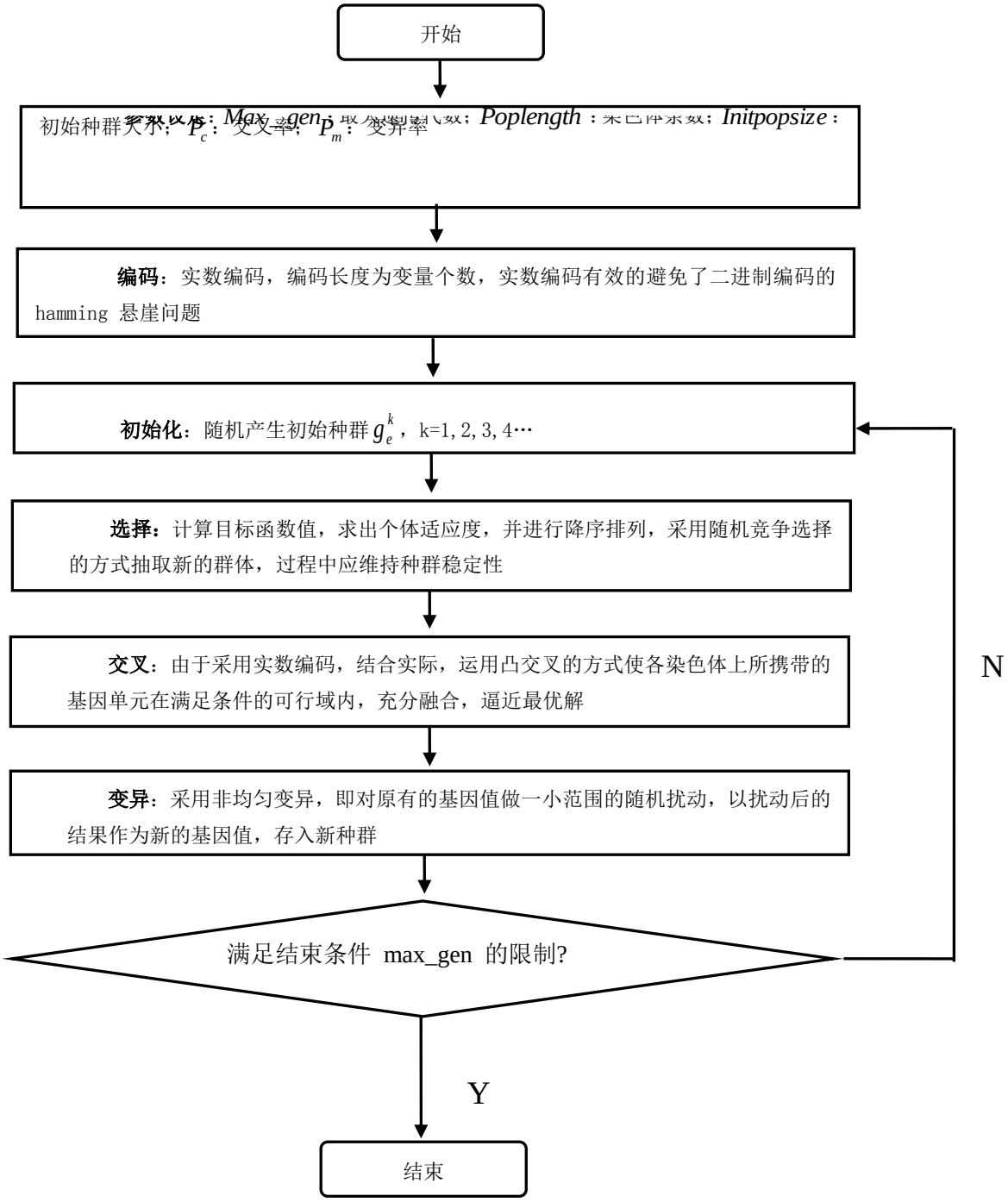
综上得出则第 r 间隔平均停车次数为 $h_{ij}^{r}=\dfrac{(\sum_{j=1}^{n}\sum_{i=1}^{n}h_{ij}^{r}q_{ij}^{r})}{q^{r}}$，时间段 T 所有停车辆的平均停车次数 $\bar{h}=(\sum_{r=1}^{w}h^{r}q^{r})/\sum_{r=1}^{w}q^{r}$。效率控制目标函数为：$\min f_{1}=\dfrac{\bar{d}\bullet\bar{h}}{c}$，具体交通信号控制算法流程如图 1 所示：

图 1 单点定时交通信号灯控制算法流程图



②遗传算法（Genetics Algorithm）简称 GA，是以自然选择和生物遗传理论为基础，将生物进化过程中的“物竞天择，适者生存”的规律与群体内部的染色体的随机信息交换机制相结合，是一种高效的全局寻优搜索算法。具体遗传算法(相关程序见附录程序 1)流程如图 2 所示：

图 2 遗传算法流程图



2.问题的求解

（1）**时间复杂度**

时间复杂度是指执行算法所需要的计算工作量，有以下几种计算方法：

①一般情况下，算法的基本操作重复执行的次数是模块 n 的某一个函数 f(n)，因此，算法的时间复杂度记做：T(n)=O(f(n))

分析：随着模块 n 的增大，算法执行的时间的增长率和 f(n) 的增长率成正比，所以 f(n) 越小，算法的时间复杂度越低，算法的效率越高。

②在计算时间复杂度的时候，先找出算法的基本操作，然后根据相应的各语句确定它的执行次数，再找出 T(n) 的同数量级（它的同数量级有以下：1，log(2)n，n，n log(2)n ，n 的平方，n 的三次方，2 的 n 次方，n!），找出后，f(n) = 该数量级，若 T(n)/f(n) 求极限可得到一常数 c，则时间复杂度 T(n) = O(f(n))

③在 pascal 中比较容易理解，容易计算的方法是：看看有几重 for 循环，只有一重则时间复杂度为 O(n)，二重则为 O(n^2)，依此类推，如果有二分则为 O(log n)，二分例如快速幂、二分查找，如果一个 for 循环套一个二分，那么时间复杂度则为 O(n log n)。

我们按照上述方法对遗传算法进行分析后，得到得到单点定时交通信号灯控制算法的时间复杂度 T(n)= O(n^3)，遗传算法的时间复杂度 T(n)= O(n^2)

（2）空间复杂度

一个程序的空间复杂度是指运行完一个程序所需内存的大小。利用程序的空间复杂度，可以对程序的运行所需要的内存多少有个预先估计。一个程序执行时除了需要存储空间和存储本身所使用的指令、常数、变量和输入数据外，还需要一些对数据进行操作的工作单元和存储一些为现实计算所需信息的辅助空间。程序执行时所需存储空间包括以下两部分。

①固定部分。这部分空间的大小与输入/输出的数据的个数多少、数值无关。主要包括指令空间（即代码空间）、数据空间（常量、简单变量）等所占的空间。这部分属于静态空间。

②可变空间，这部分空间的主要包括动态分配的空间，以及递归栈所需的空间等。这部分的空间大小与算法有关，其计算方法与时间复杂度基本一致。

一个算法所需的存储空间用 f(n)表示。

S(n)=O(f(n))

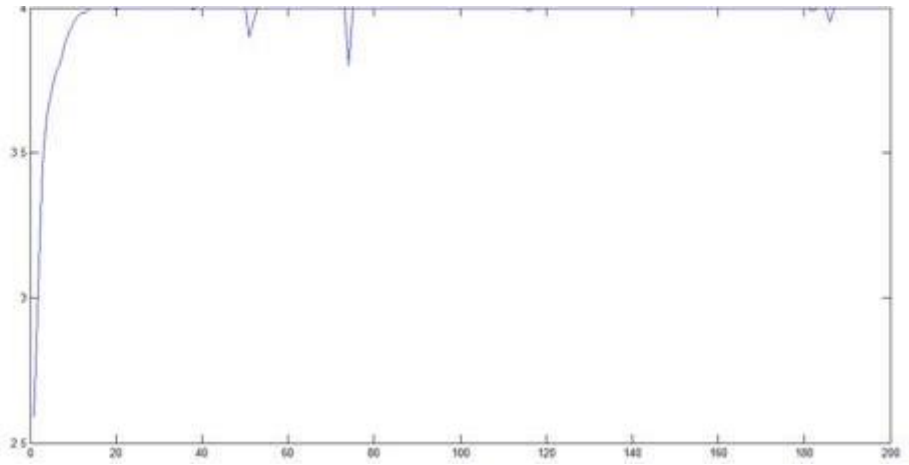
其中 n 为问题的规模，S(n)表示空间复杂度。

我们依据上述所说，对上述两种算法进行分析后，得到单点定时交通信号灯控制算法的空间复杂度 S(n)=O(1)，得到遗传算法的空间复杂度 S(n)=O(1)

（3）收敛性

算法的收敛性体与算法的迭代过程相关，用 MATLAB[3] 画出算法迭代过程图如下：

图 3 算法迭代过程图



从图中可以看出，随着迭代次数的增加，算法逐渐收敛全局。

综上所述，遗传算法运行效率较高，且有较高的计算精度，是一种高效的全局寻优算法。

二、问题二的分析与求解

1. 问题的分析

针对问题二，对交通信号进行配时优化研究，求解出改善后的交通信号配饰方案并进行仿真检验。在问题一的基础上，我们运用基于遗传算法的城市相邻两交叉口协调控制方法，有效提高实际平面交叉口通行能力和服务水平，减少城市交通延误，改善城市交通现状，最后我们运用 VISSIM[4]对相邻两交叉口协调控制方案进行仿真检验。

2.对问题的求解

模型　相邻两交叉口协调控制模型

(1)模型的思路

交叉口设置单点定时信号控制时，车辆经常遇到红灯，时开时停，造成行车不畅。传统的干道协调控制设计以获取最大绿波宽度为设计目标，目的是为减少车辆，通过协调控制范围内各交叉口的总延误，缩短行程时间，使车辆能在最短时间内通过。本文以将最大绿波宽带与系统的干线车辆行程时间最短相结合为目标，主要考虑车辆趋于停车线处的平均延误时间、有效绿灯时间以及实际平面交叉口通行能力，建立非线性的目标函数模型。

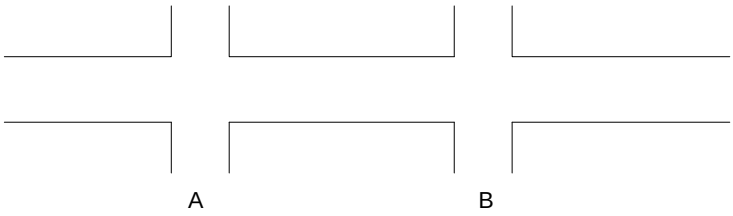
（2）模型的建立

①定义

a.干道协调系统内进口到：主线上不直接与干道协调系统之外的道路相连的进口道，如图 2 中交叉口 A 的主线右进口道，交叉口 B 的左侧主线进口道。

b.干道协调系统外进口道：从外部进入干道协调系统的进口道则称为系统外进口道。次路进口道、主线两端交叉口外侧的两个进口道均属于系统外部进口道 ,如图 2 中交叉口 A 的主线左进口道 ,交叉口 B 的主线右进口道以及所有次要道路的进口道，如图 4：

图 4 武汉市协调控制系统



②相关公式

$$q^d(j)=\sum_{i=1}^{j-t}q^o(i)F(1-F)^{j-t-i} \qquad (5)$$

式中：$q^d(j)$ 为第 j 个时段，下游断面 d 上预计的车辆到达率；$q^o(i)$ 为第 i 个时段，上游出口道的车辆驶出率；F 为车辆离散系数

$$F=1/(1+At) \qquad (6)$$

t 为路段的平均行驶时间的 0.8 倍，以时段计，A 为参数,可实测，无数据资料时可取 A=0.35

$$C_O=(1.5L+5)/(1-Y) \qquad (7)$$

式中：C_O:最佳周期长度（s）；L:总损失时间（s）；Y：所有相位关健车流的流量比之和。

$$L=\sum(I-B+l) \qquad (8)$$

I:信号阶段之间的绿灯时间间隔（s）；B:黄灯时间（s）；l：相位的绿灯损失时间（s）。

在各相位之间，绿灯时间的分配也是以车辆延误最少为原则。当取相等的饱和度时，相位绿信比通常与该相位的交通流量比率大致成正比。单一相位的绿灯时间计算公式为：

$$g_i=\frac{y_i}{Y_i}(C_O-L) \qquad (9)$$

g_i：第 i 相位的绿灯时间；y_i：第 i 相位的流量比

将车流进行离散 ,然后根据上述到达率公式 ,可以得到车辆在交叉口的到达驶离图式 ,见图 5。图 5 中锯齿代表车辆累积到达量 ,斜直线代表该路口的最大放行率 (饱和流率) ,这两条线与时间轴围成的面积即为全部车辆延误时间总和 ,而锯齿线与斜直线的交点的车辆累计到达数则为该路口一个周期的停车次数。

图 6 是交叉口车辆的驶出示意图 ,易见通过主线方向出口的交通量由主

线进口的直行车和相交道路的左、右转车组成 ,因此可分别进行分析。

图 5 交叉口到达驶离图示

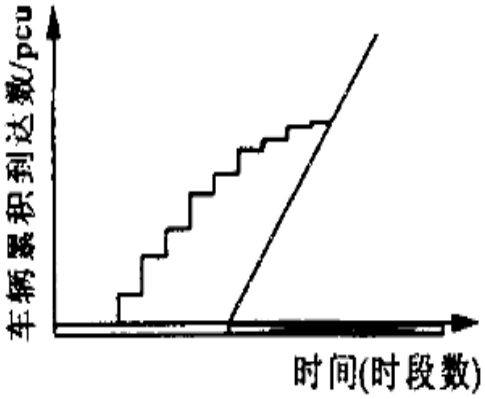
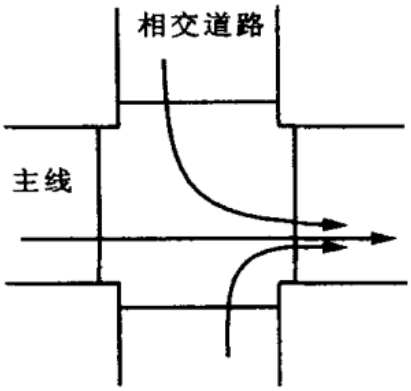


图 6 交叉口主线出口车流组成



(3) 模型的求解

①求解思想

a.以主线出口道的流率模式为基础 ,利用公式 (1)依次推算各交叉口车流在到达下一交叉口进口道 (内部进口)时的流率分布。

b.根据进口道车流到达模式按照图 2 的原理计算延误。

c.将每个交叉口的各个进口 (包括控制系统的内进口、外进口)的延误进行加权相加 ,可得到该交叉口的延误。 将协调控制系统内所有交叉口的延误相加可得到协调控制系统的总延误。

系统的总延误 D 是各交叉口的绿时差 (Oi)的函数 ,即 $D = f^{D}(o1, o2)$,而使 D 获得最小值得一系列绿时差值 ,即为所求的延误最优绿时差系列。

同理 ,按照上述分析求解系统内各交叉口的停车次数 ,进行累加获得系统的总停车次数 S。而使 $S = f^{S}(o1, o2)$ 获得最小值得一系列绿时差值，则为停车最优绿时差系列。

②算法设计

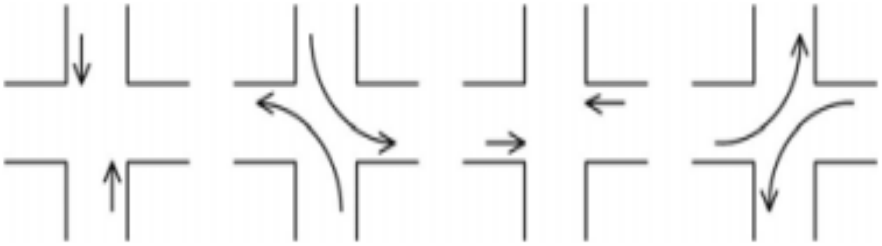
针对城市相邻两交叉口协调控制的特点，结合问题二，算法设计如图 7。

图 7 算法设计流程图

③求解

有信号灯控制下东进口的左转车流、直行车流和西进口的直行车流、左转车流发生严重的干扰，为了避免上述干扰，使车流安全快速的通过交叉口，设计如图 8 所示:

图 8 四相位信号配时图

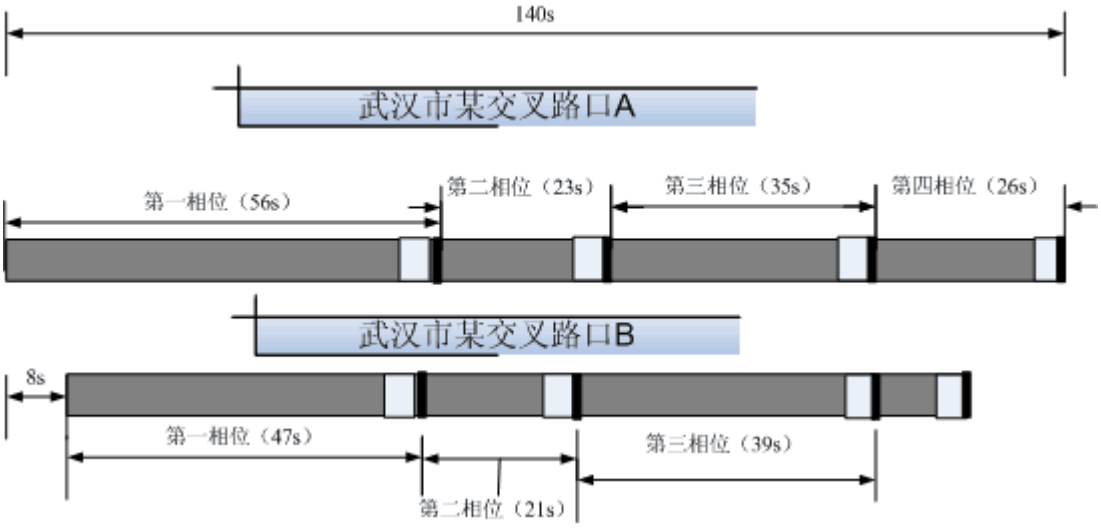


根据题干条件，列出交通基础数据如表 1 所示以及现有两交叉口（武汉市 A 交叉口、武汉市 B 交叉口）配时方案如图 9 所示。

表 1　交通基础数据

交通数据		交通数据（$PCU \cdot h^{-1}$）		
		左转	直行	右转
武汉市 A 交叉口	东进口	366	1394	98
	西进口	295	166	72
	南进口	525	408	300
	北进口	100	394	576
武汉市 B 交叉口	东进口	802	1154	576
	西进口	450	304	329
	南进口	169	420	84
	北进口	132	535	90

图 9 两相邻交叉口现有信号配时方案

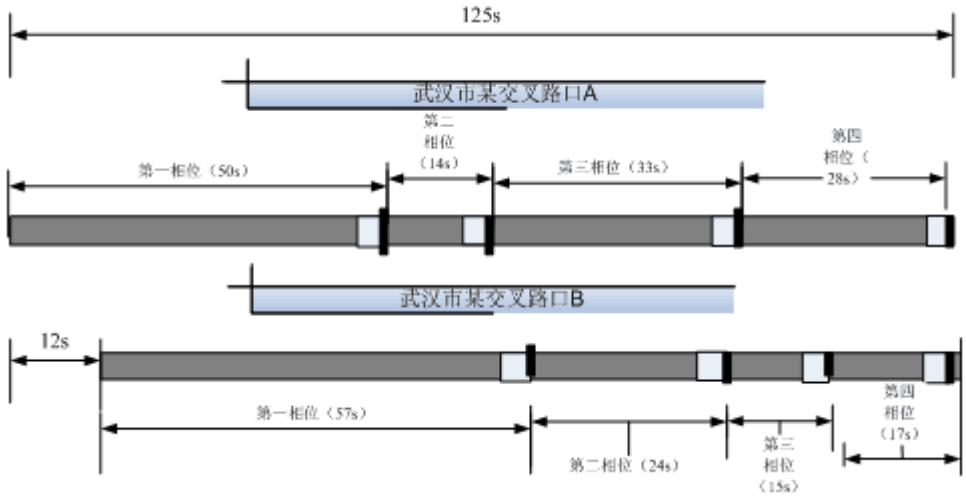


利用上述推导过程，采用 EXCEL、MATLAB 软件对基础数据进行处理获得优化结果。如下表 2 所示：

表 2 优化后信号配时方案

交叉口	周期	相位差	相位 1		相位 2		相位 3		相位 4	
			绿	黄+红	绿	黄+红	绿	黄+红	绿	黄+红
A	125		46	4	10	4	29	4	24	4
B	113		53	4	20	4	9	4	13	4

图 10 两相邻交叉口优化后的信号配时方案



由上可知，优化后武汉市交叉口 A 的信号周期为 125s，第一相位时间 50s，第二相位时间 14s，第三相位时间 33s，第四相位时间 28s。两个交叉口的相位差为 12s，武汉市交叉口 B 的信号周期为 113s，第一相位时间 57s，第二相位时间 24s，第三相位时间 15s，第四相位时间 17s。

为论述上述方法的有效性，运用 VISSIM 软件(具体程序见附录程序 2)，将调查得到的现有的交通数据输入并进行仿真得到结果如表 3 所示：

表 3 优化后仿真结果

结点	进道口	有效绿长	饱和度	通行能力	平均延误时间
	EA	49.8738	0.1327	2192.0758	19.1944
武汉市	WA	14.3072	0.0109	628.8355	0.4754
某交叉路口	SA	33.0971	0.0584	1454.6983	2.6082
A	NA	28.7217	0.044	1262.3902	1.4715
	EB	24.6868	0.2117	3011.2983	72.2974
武汉市	WB	15.3409	0.0387	1288.0079	1.2384
某交叉路口	SB	17.2556	0.0149	800.3964	0.5097
B	NB	9.7463	0.0189	900.2973	0.5632

参照控制延误与服务水平表(表 4)，进一步得到两相邻交叉口现状与优化后的服务水平对比如表 5 所示：

表 4 控制延误与服务水平表

服务等级	控制延误时间/S	交通状况
A	小于 10	无拥挤
B	10~20	轻微拥挤
C	20~35	无重大拥挤
D	35~55	正常情况下无拥挤
E	55~80	恰好拥挤
F	大于 80	超负荷能力、拥挤

表 5 服务水平对比

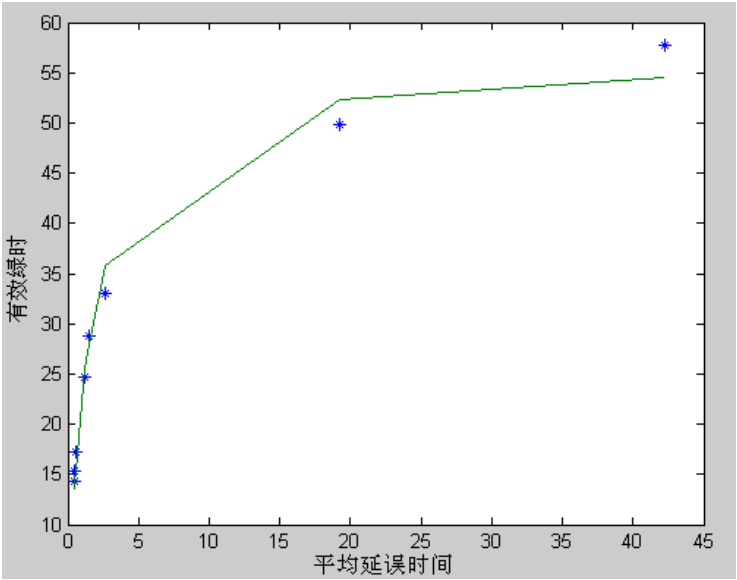
交叉口		交叉口延迟/s	交叉口服务水平
武汉市某交叉路口 A	现状	38.75	D
	优化后	23.75	C
武汉市某交叉路口	现状	45.53	D
	优化后	44.60	D

由表 4 可以得到，优化后武汉市 A 交叉口的延迟时间 38.71%，武汉市 B 交叉口的延迟时间降低了 2.04%，两相邻总交叉口的延迟时间降低了 40.75%，且武汉市 A 交叉口服务水平提高了一个等级。综上可知，新的配时方案是具有可行性的，本文的优化模型和求解释有效的。

为进一步验证优化模型的有效性，我们根据仿真后的数据与题目中的数据探

讨平均延误时间与有效绿时的关系，用 MATLAB 软件编程得出平均延误时间与有效绿时的函数关系图(程序见附录 3)，如图 11 所示。

图 11 平均延误时间与有效绿时的函数关系图



由图可知，有效绿时与平均延误时间成正向关系（$y=\frac{x}{0.0266+0,0177x}$），符合上述模型建立的条件，所以，进一步体现出优化模型的可靠性、准确性。

§6 误差分析与灵敏度分析

一、误差分析

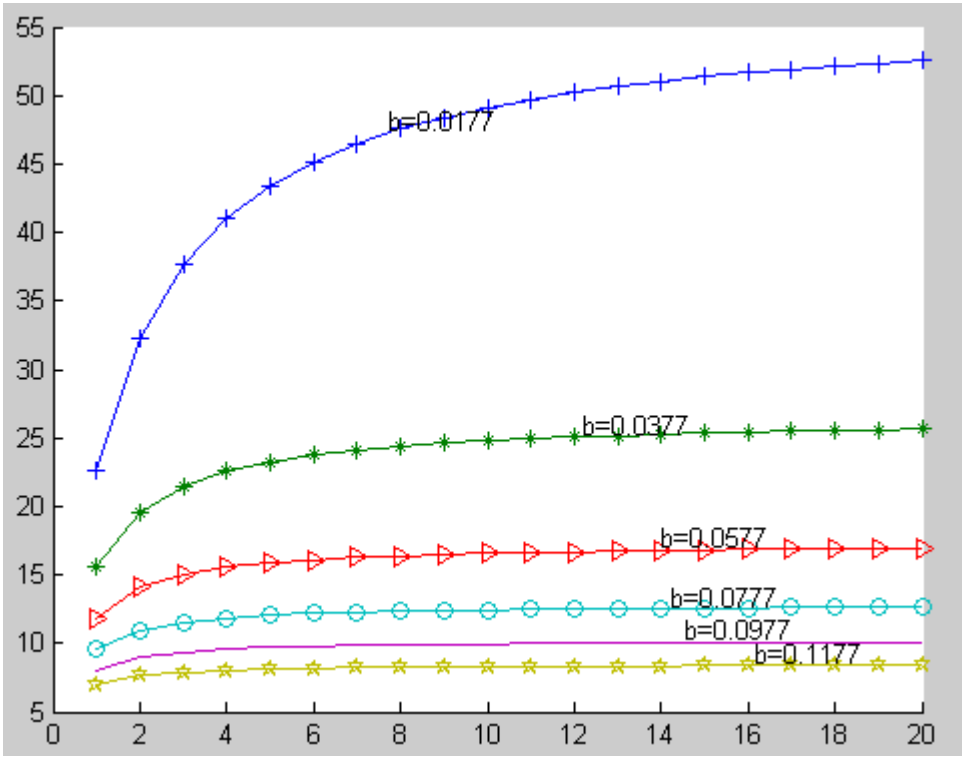
在问题一研究收敛性对比分析中，为了便于计算，我们做了一些细微的处理，对研究的结果与真实结果会有一点误差，但影响并不是很大。在问题二中，我们忽略了一些其他因素，如行人，上下班高峰期等，都会给模型本身真实效果带来一定误差；而问题本身，也只给我们两组数据，由于数据较少，会有一些偶然因素，造成一些错误评估，但是总体来说，影响不是很大。

二、灵敏度分析

在问题二中进行平均延误时间与有效绿时函数进行非线性拟合时，仅仅从仿真检验后的 8 组数据进行考虑，得出的非线性曲线方程：$y=\frac{x}{0.0266+0,0177x}$

数据的不足会在一定程度上影响最终结果，因而可是适当的增减平均延误时间的长度，进行灵敏度分析，针对不同的 d 的不同增加值我们运用 MATLAB 软件进行灵敏度分析(具体程序见附录程序 4)结果如下图 12 所示：

图 12 灵敏度分析图



由灵敏度分析可知，当 d 确定下来时，参变量 b 值变化引起的 y 值变化量相等；当 b 值一定时， y 值随着 d 值的增加而增加，但增加量越来越小。同时，对于不同的 b 值， y 值随着 b 值的增大而减小，且 b 值增大到一定程度后， y 值基本保持不变。则说明参照这两指标建立的优化模型可靠、准确。

§7 模型的评价

一、模型的优点：

1. 本文巧妙地运用流程图，将算法完整清晰的展现出来；

2. 本文建立的模型有成熟的理论基础，又有相应的专业软件支持，可信度较高；

3. 运用多种数学软件（如 MATLAB、EXCEL），取长补短，使计算结果更加精确；

4. 本文建立的模型与实际紧密联系，充分考虑到现实情况的不同阶段，从而使模型更贴近实际，通用性强。

二、模型的缺点：

1. 模型采用原始数据进行计算，变量太多，软件在计算过程中难以收敛，有一定的误差；

2. 对于一些结果的分析我们忽略了一些次要的影响因素，会对结果产生一些误差。

§8 模型的改进与推广

一、模型的该进

1.我们的模型是建立两家插口的信号周期时长相等的基础上，但是现实生活中由于各个路况的不同，每个交叉口的信号周期也不相同，这种假设只是为了解题的方便，所以在日后改进模型时应考虑进这个因素，使得情况更接近实际，模型更具有说服力。

2.在算法上，本文章采用的是遗传算法，该算法克服了一般迭代算法容易陷入部极小的陷阱而陷入死循环的缺点，但是一种全局优化方法。但是遗传算法的早熟现象即很快收敛到局部最优解而不是全局最优解是一个很致命的缺点，因此在改进模型对算法进行改进时可以考虑将遗传算法与模拟退火法相结合的混合遗传算法，再次对目标函数进行优化计算以得到更好的模型。

二、模型的推广

我们研究的模型是基于遗传算法的相邻两交叉口协调控制模型，解决相邻两交叉口交通信号灯配时优化问题。遗传算法是一种高精度的算法，与单指标模型不同的是，我们将多指标巧妙地融合在一起，使得本模型不仅可以解决相邻两交叉口的交通灯配时优化问题，而且可以将模型推广至解决相邻四交叉口、环形交叉口等交通信号灯配饰优化问题，甚至于解决最短路径的旅行商等多种问题。

参考文献

[1]芦娟.郑国华.从国内外治堵经验看解决长沙城市交通拥堵问题的对策[J].企业家天地，2011.（7）：18-20.

[2]Akcelik R, Rouphail N M. Estimation of Delays at Traffic Signals for Variable Demand Conditions[J]. Transportation Research, Part B: Methodological, 1993, 27(2): 109-131.

[3]吴礼斌.经济数学实验与建模.北京：国防工业出版，2013.

[4] 李林,徐建闽.VISSIM 在信号交叉口优化中的应用[J].现代交通科技,2009,6，（5）：62-64.

[5]刘　勇,康立山, 陈毓屏. 非数值并行算法—— 遗传算法[M].北京: 科学出版社, 1995.

附录

附录一：

交通数据

交通数据		交通流量$(PCU \cdot h^{-1})$			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			

程序 1：

用 MATLAB 编程的遗传算法

初始化：

```
%初始化种群
%pop_size: 种群大小
%chromo_size: 染色体长度

function initilize(pop_size, chromo_size)
global pop;
for i=1:pop_size
    for j=1:chromo_size
        pop(i,j) = round(rand);
    end
end
clear i;
clear j;
```

计算适应度：（该函数应该根据具体问题进行修改，这里优化的函数是前述的一维函数）

```
%计算种群个体适应度，对不同的优化目标，此处需要改写
%pop_size: 种群大小
%chromo_size: 染色体长度
```

```
function fitness(pop_size, chromo_size)
global fitness_value;
global pop;
global G;
for i=1:pop_size
    fitness_value(i) = 0.;
end

for i=1:pop_size
    for j=1:chromo_size
        if pop(i,j) == 1
            fitness_value(i) = fitness_value(i)+2^(j-1);
        end
    end
     fitness_value(i)                                                    =
-1+fitness_value(i)*(3.-(-1.))/(2^chromo_size-1);
     fitness_value(i) = -(fitness_value(i)-1).^2+4;
end

clear i;
clear j;
```

对个体按照适应度大小进行排序：

```
%对个体按适应度大小进行排序，并且保存最佳个体
%pop_size: 种群大小
%chromo_size: 染色体长度

function rank(pop_size, chromo_size)
global fitness_value;
global fitness_table;
global fitness_avg;
global best_fitness;
global best_individual;
global best_generation;
global pop;
global G;

for i=1:pop_size
    fitness_table(i) = 0.;
end

min = 1;
temp = 1;
temp1(chromo_size)=0;
```

```
for i=1:pop_size
    min = i;
    for j = i+1:pop_size
        if fitness_value(j)<fitness_value(min);
            min = j;
        end
    end
    if min~=i
        temp = fitness_value(i);
        fitness_value(i) = fitness_value(min);
        fitness_value(min) = temp;
        for k = 1:chromo_size
            temp1(k) = pop(i,k);
            pop(i,k) = pop(min,k);
            pop(min,k) = temp1(k);
        end
    end

end

for i=1:pop_size
    if i==1
        fitness_table(i) = fitness_table(i) + fitness_value(i);
    else
        fitness_table(i) = fitness_table(i-1) + fitness_value(i);
    end
end
fitness_table
fitness_avg(G) = fitness_table(pop_size)/pop_size;


if fitness_value(pop_size) > best_fitness
    best_fitness = fitness_value(pop_size);
    best_generation = G;
    for j=1:chromo_size
        best_individual(j) = pop(pop_size,j);
    end
end


clear i;
clear j;
clear k;
clear min;
```

```
clear temp;
clear temp1;
```

选择操作：

```
%轮盘赌选择操作
%pop_size: 种群大小
%chromo_size: 染色体长度
%cross_rate: 是否精英选择

function selection(pop_size, chromo_size, elitism)
global pop;
global fitness_table;

for i=1:pop_size
    r = rand * fitness_table(pop_size);
    first = 1;
    last = pop_size;
    mid = round((last+first)/2);
    idx = -1;
    while (first <= last) && (idx == -1)
        if r > fitness_table(mid)
            first = mid;
        elseif r < fitness_table(mid)
            last = mid;
        else
            idx = mid;
            break;
        end
        mid = round((last+first)/2);
        if (last - first) == 1
            idx = last;
            break;
        end
   end

   for j=1:chromo_size
        pop_new(i,j)=pop(idx,j);
   end
end
if elitism
    p = pop_size-1;
else
    p = pop_size;
```

```
end
for i=1:p
   for j=1:chromo_size
        pop(i,j) = pop_new(i,j);
   end
end

clear i;
clear j;
clear pop_new;
clear first;
clear last;
clear idx;
clear mid;
```

交叉操作：

```
%单点交叉操作
%pop_size: 种群大小
%chromo_size: 染色体长度
%cross_rate: 交叉概率

function crossover(pop_size, chromo_size, cross_rate)
global pop;
for i=1:2:pop_size
    if(rand < cross_rate)
        cross_pos = round(rand * chromo_size);
        if or (cross_pos == 0, cross_pos == 1)
            continue;
        end
        for j=cross_pos:chromo_size
            temp = pop(i,j);
            pop(i,j) = pop(i+1,j);
            pop(i+1,j) = temp;
        end
    end
end


clear i;
clear j;
clear temp;
clear cross_pos;
```

变异操作：

```
%单点变异操作
%pop_size: 种群大小
%chromo_size: 染色体长度
%cross_rate: 变异概率
function mutation(pop_size, chromo_size, mutate_rate)
global pop;

for i=1:pop_size
    if rand < mutate_rate
        mutate_pos = round(rand*chromo_size);
        if mutate_pos == 0
            continue;
        end
        pop(i,mutate_pos) = 1 - pop(i, mutate_pos);
    end
end

clear i;
clear mutate_pos;
```

打印算法迭代过程：

```
%打印算法迭代过程
%generation_size: 迭代次数

function plotGA(generation_size)
global fitness_avg;
x = 1:1:generation_size;
y = fitness_avg;
plot(x,y)
```

算法主函数：

```
%遗传算法主函数
%pop_size: 输入种群大小
%chromo_size: 输入染色体长度
%generation_size: 输入迭代次数
%cross_rate: 输入交叉概率
%cross_rate: 输入变异概率
%elitism: 输入是否精英选择
%m: 输出最佳个体
%n: 输出最佳适应度
%p: 输出最佳个体出现代
%q: 输出最佳个体自变量值
```

```
function [m,n,p,q] = GeneticAlgorithm(pop_size, chromo_size,
generation_size, cross_rate, mutate_rate, elitism)

global G ; %当前代
global fitness_value;%当前代适应度矩阵
global best_fitness;%历代最佳适应值
global fitness_avg;%历代平均适应值矩阵
global best_individual;%历代最佳个体
global best_generation;%最佳个体出现代



fitness_avg = zeros(generation_size,1);

disp "hhee"

fitness_value(pop_size) = 0.;
best_fitness = 0.;
best_generation = 0;
initilize(pop_size, chromo_size);%初始化
for G=1:generation_size
    fitness(pop_size, chromo_size);  %计算适应度
    rank(pop_size, chromo_size);  %对个体按适应度大小进行排序
    selection(pop_size, chromo_size, elitism);%选择操作
    crossover(pop_size, chromo_size, cross_rate);%交叉操作
    mutation(pop_size, chromo_size, mutate_rate);%变异操作
end
plotGA(generation_size);%打印算法迭代过程
m = best_individual;%获得最佳个体
n = best_fitness;%获得最佳适应度
p = best_generation;%获得最佳个体出现代

%获得最佳个体变量值，对不同的优化目标，此处需要改写
q = 0.;
for j=1:chromo_size
    if best_individual(j) == 1
            q = q+2^(j-1);
    end
end
q = -1+q*(3.-(-1.))/(2^chromo_size-1);

clear i;
clear j;
```

程序 2：

VISSIM VAP 控制代码如下：

```
PROGRAM renming;
VAP_FREQUENCY 1;
CONST
/*仿真前 VC++修改下边的四个控制参数*/
MAX_STG1=22,
MAX_STG2=3,
MAX_STG3=10,
MAX_STG4=40;
/*ARRAYS*/
/*SUBROUTINES*/
/*PARAMETERS DEPENDENT ON SCJ-PROGRAM*/
/*EXPRESSIONS*/
S00Z001:  IF Any_interstage_active THEN
            GOTO PROG_ENDE
          ELSE
S00Z002:  IF Stage_active(1) THEN
S01Z002             IF StgT(1)<MAX_STG1 THEN
                GOTO PROG_ENDE
          ELSE
S00Z003:    Interstage(1,2);
            GOTO PROG_ENDE
          END
       END
     END
S00Z004: IF Stage_active(2) THEN
S01Z004:   IF StgT(2)<MAX_STG2 THEN
              GOTO PROG_ENDE
         ELSE
S02Z005:   Interstage(2,3);
           GOTO PROG_ENDE
        END
     END
S00Z006:  IF StgT(3)<MAX_STG3 THEN
             GOTO PROG_ENDE
          ELSE
S02Z007:   Interstage(3,4);
            GOTO PROG_ENDE
          END
       END
S00Z008: IF Stage_active(4) THEN
S01Z008:  IF NOT(StgT(4)<MAX_STG4) THEN
```

```
    S02Z009:      Interstage(4,1);
          END
       END
    PROG_ENDE
```

程序 3：

用 MATLAB 绘制有效绿时与平均延误时间的函数关系图

```
x1=[19.19447323
0.475405411
2.608234886
1.471595665
42.29745002
1.238431564
0.509783828
0.563244317];
y1=[49.87388155
14.30720068
33.09714529
28.72177248
57.71655104
24.68681863
15.34093162
17.25569871
];
x=sort(x1);
y=sort(y1);
b0=[0.02,0.02];   %初始参数值
fun=inline('x./(b(1)+b(2)*x)','b','x'); % 定义函数
[b,r,j]=nlinfit(x,y,fun,b0);
b                                          %最佳参数
y2=x./(0.0266+0.0177*x);%根据 b1 写出具体函数
plot(x,y,'*',x,y2,'-')
xlabel('平均延误时间'),ylabel('有效绿时')
```

程序 4：用 MATLAB 进行灵敏度分析

```
x=1:20;
y1=x./(0.0266+0.0177*x)
y2=x./(0.0266+0.0377*x)
y3=x./(0.0266+0.0577*x)
y4=x./(0.0266+0.0777*x)
y5=x./(0.0266+0.0977*x)
y6=x./(0.0266+0.1177*x)
hold on
plot(x,y1,'-+',x,y2,'-*',x,y3,'->',x,y4,'-o',x,y5,'-',x,y6,'-p')
gtext('b=0.0177')
```

```
gtext('b=0.0377')
gtext('b=0.0577')
gtext('b=0.0777')
gtext('b=0.0977')
gtext('b=0.1177')
a5=y6-y5
a4=y5-y4
a3=y4-y3
a2=y3-y2
a1=y2-y1
```

程序 4：

用 MATLAB 进行灵敏度分析

```
x=1:20;
y1=x./(0.0266+0.0177*x)
y2=x./(0.0266+0.0377*x)
y3=x./(0.0266+0.0577*x)
y4=x./(0.0266+0.0777*x)
y5=x./(0.0266+0.0977*x)
y6=x./(0.0266+0.1177*x)
hold on
plot(x,y1,'-+',x,y2,'-*',x,y3,'->',x,y4,'-o',x,y5,'-',x,y6,'-p')
gtext('b=0.0177')
gtext('b=0.0377')
gtext('b=0.0577')
gtext('b=0.0777')
gtext('b=0.0977')
gtext('b=0.1177')
a5=y6-y5
a4=y5-y4
a3=y4-y3
a2=y3-y2
a1=y2-y1
```