工业过程智能优化技术
课程报告

基于改进的遗传算法的城市交通信号优化分析

姓　　名　曾康慧

学　　号　20211003337

班　　级　220211

2024 年 7 月 9 日

目　　录

1 摘要

随着人民生活水平的不断提高，汽车进入寻常百姓家中也已成为现实，随之而来的城市交通问题则日益突现出来。因此，采用现代科学手段，研究一些智能化的方法来解决城市交通管理问题，就成为当务之急。为了缓解城市交通拥挤，本章在分析了城市道路单交叉路口交通流特性的基础上，首先建立了以车辆平均延误时间最短，以相位有效绿灯时间和饱和度为约束条件的非线性函数模型，利用改进的遗传算法对模型进行求解，得到在固定周期下的最优配时方案，仿真结果表明获得了理想的效果。

关键词：遗传算法，交叉口，交通信号控制，优化算法

2 遗传算法基本理论

2.1 遗传算法简介

遗传算法（Genetic Algorithm）是一类借鉴生物界的进化规律（适者生存，优胜劣汰遗传机制）演化而来的随机优化搜索方法，由美国的 J. Holland 教授 1975 年首先提出 [1]。

遗传学认为，遗传是作为一种指令遗传码封装在每个细胞中，并以基因的形式包含在染色体中，每个基因有特殊的位置并控制某个特殊的性质。每个基因产生的个体对环境有一定的适应性。基因杂交和突变可能产生对环境适应性强的后代，通过优胜劣汰的自然选择，适应度值高的基因结构就保存下来。

遗传算法借鉴“适者生存”的遗传遗传学理论，将优化问题的求解表示成“染色体”的“适者生存”过程，通过“染色体”群的一代代复制、交叉、变异的进化，最终得到的是最适应环境的个体，从而得到问题的最优解或者满意解。这是一种高度并行、随机和自适应的通用的优化算法。

遗传算法主要特点是直接对结构对象进行操作，不存在求导和函数连续性的限定；具有内在的隐并行性和更好的全局寻优能力；采用概率化的寻优方法，能自动获取和指导优化的搜索空间，自适应地调整搜索方向，不需要确定的规则。遗传算法的一系列优点使它近年来越来越受到重视，在解决众多领域的优化问题中得到了广泛的应用，其中也包括在交通领域的成功应用。在遗传算法中，将 n 维决策向量 $X=[x_1,x_2...,x_n]^T$ 用 n 个记号符 $X=X_1X_2...X_n \Rightarrow X=[x_1,x_2...,x_n]^T$ 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，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 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无的目标函数值相关联，灭越接近于目标函数的最优点，其适应度越大。

在遗传算法 GA 中，决策变量（待求未知量）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。

生物的进化是以集群为主体的。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，它所对应的表现型将达到或接近于问题的最优解 X^*。

生物的进化过程主要是通过染色体之间的交叉和染色体的变异来完成的。与此相对应，遗传算法中最优解的搜索过程也模仿生物的这个进化过程，使用所谓的遗传算子作用于群体 $p(t)$，进行下述遗传操作，从而得到新一代的群体 $p(t+1)$。

遗传算法包括三个基本操作：选择、交叉和变异。

这些基本操作又有许多不同的方法，使得遗传算法在使用时具有不同的特色。

（1）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。

（2）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。

（3）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。

2.2 遗传算法的特点

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。是一类可用于复杂系统优化计算的鲁棒搜索算法，与其他一些优化算法相比，它具有很多特点。

传统的优化算法主要有三种：枚举法、启发式算法和搜索算法。

1. 枚举法

枚举法在可行解集合内枚举所有可行解，以求出精确最优解。对于连续函数，该方法要求先对其进行离散化处理，这样就可能因离散处理而永远达不到最优解。此外，当枚举空间比较大时，该算法的求解效率非常低，极其耗时。

2. 启发式算法

启发式算法是寻求一种能产生可行解的启发式规则，以找到一个最优解或近似最优解。启发式算法的求解效率比较高，但对每一个需求解的问题必须找出其特有的启发式规则，这个启发式规则一般无通用性，不适合于其他问题。

3. 搜索算法

搜索算法在可行解集合的一个子集内进行搜索操作，以找到问题的最优解或者近似最优解。搜索算法虽然保证不了一定能够得到向题的最优解，但若适当的利用一些启发知识，就可在近似解的质量和效率上达到一种较好的平衡。

遗传算法的主要特点为以下几点：

（1）遗传算法是对要寻优参数的编码进行操作，而不是对参数本身。

（2）遗传算法是从“群体”出发（多个初始解个体）开始的并行操作，而不是从一个点开始。因而可以有效地防止搜索过程收敛于局部最优解，而且有较大的可能求得全局最优解。

（3）遗传算法采用目标函数来确定基因的遗传概率，而不需要其他的推导和附属信息，从而对问题的依赖性较小。所以遗传算法对于待寻优的函数基本无限制，它既不要求函数连续，更不要求可微，即可以是数学解析式所表达的显函数（大多数问题），也可以是其他方式的隐函数（用数值解描述的函数方程）甚至是神经网络（例如第 6 章遗传算法优化的 BP 神经网络）等隐函数。

（4）遗传算法的操作均使用随机概率的方式，而不是确定性的规则。

（5）遗传算法在解空间内不是盲目地穷举或完全随机测试，而是一种启发式搜索，其搜索效率往往优于其他方法。

（6）遗传算法更适合大规模复杂问题的优化。

2.3 基本遗传算法的工作流程

基本遗传算法的工作流程和结构形式如图2-1所示，它的运行过程是一个典型的迭代过程，其必须完成的工作和基本步骤如下：

（1）选择编码策略，把参数集合空间转化为编码后的个体空间：

（2）根据实际问题定义适应度函数；

（3）确定遗传策略，包括种群大小，选择、交叉和变异方法，以及确定选择概率、交叉概率和变异概率等遗传参数；

（4）随机初始化生成初始群体；

（5）计算当前种群中个体编码串解码后的适应度；

（6）按照遗传策略，运用选择、交叉和变异算子作用于群体，形成下一代种群；

（7）判断种群性能是否满足某一指标，或者已完成预定迭代次数满足则输出最佳个体，退出。不满足则返回（6）。

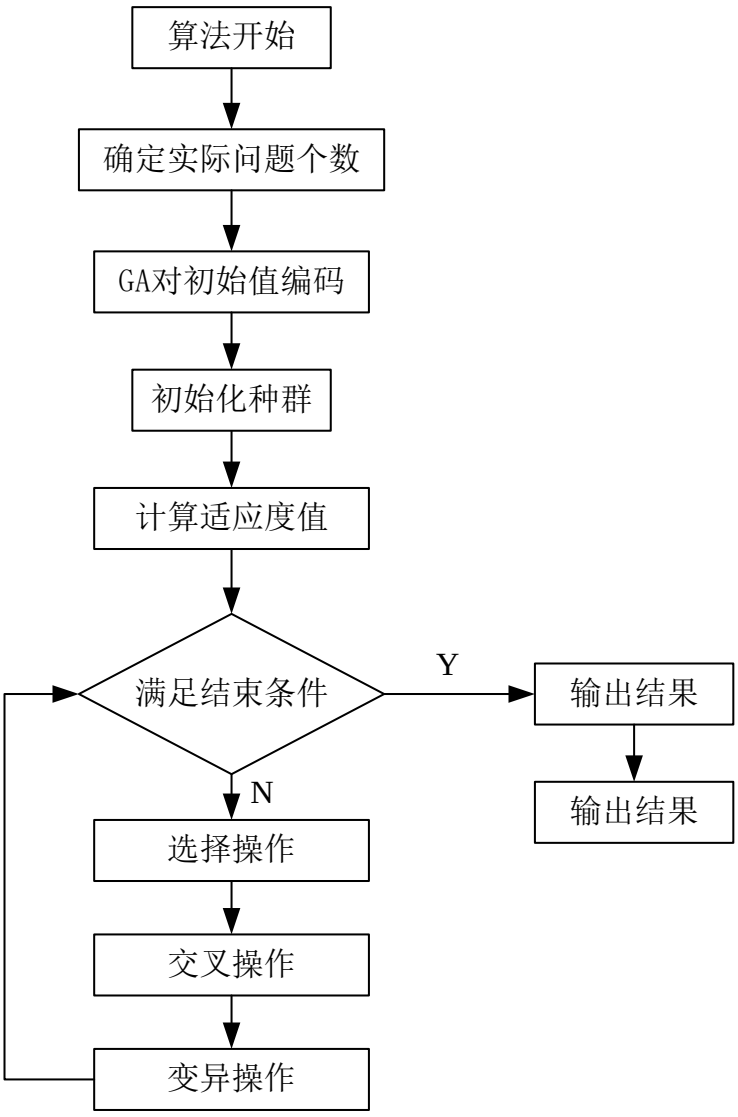

图 2-1　基本遗传算法的流程

3 遗传算法的基本要素

在应用遗传算法时，需要解决以下几个基本要素：编码、适应度函数、遗传算子、控制参数的选择和约束条件的处理。

3.1 编码问题

遗传算法不能直接处理问题空间的参数，而需要把问题的可行解从其解空间转换到遗传算法所能处理的搜索空间中，这一转换方法就称为编码。一般来说，由于遗传算法的鲁棒性，它对编码的要求并不苛刻。但由于编码的方法对于个体的染色体排列形式，以及个体从搜索空间的基因型到解空间的表现型的转换和遗传算子的运算都有很大影响，因此编码方法在很大程度上决定了如何进行群体的遗传进化运算及遗传进化运算的效率。因此，作为遗传算法流程中第一步的编码技术，是遗传算法理论与应用研究中需要首先认真解决的课题。

针对一个具体应用问题，应用最为广泛的是二进制编码和浮点数（十进制）编码。

1，二进制编码二进制编码方法是遗传算法中最常用的一种编码方法，它使用的编码符号集是由二进制符号 0 和 1 所组成的二值符号集 0，1，它所构成的个体基因型是一个二进制编码符号串，其符号串的长度与问题所要求的精度有关。其主要优点在于编码和解码操作简单，交叉和变异等遗传操作便于实现，而且便于利用模式定理进行理论分析等。其缺点在于不便于反映所求问题特定知识，对于一些连续函数的优化问题等，也由于遗传算法的随机特性而使得其局部搜索能力较差，对于一些多维和高精度要求的连续函数优化，二进制编码存在着连续函数离散化时的映射误差，个体编码串较短时，可能达不到精度要求；而个体编码串的长度较长时，虽然能提高精度，但却会使算法的搜索空间急剧扩大，增加了计算复杂性，降低了运算效率。

2. 十进制编码

针对二进制编码方法的这些缺点，人们提出了个体的浮点数编码方法。所谓浮点数编码方法，是指个体的每个基因值用某一范围内的一个浮点数来表示，个体的编码长度等于其决策变量的个数。例如，若某一个优化问题含有 5 个变量 $X_i(i=1,2,...,n)$，每个变量都有其对应的上下限 $[U^i{}_{\min}, U^i{}_{\max}]$，则 X：

5.60	6.50	3.50	3.70	5.00

表示一个体的基因型，其对应的表现型是：$x = [5.60, 6.50, 3.50, 3.70, 5.00]^T$

在十进制编码方法中，必须保证基因值在给定的区间限制范围内，遗传算法中所使用的交叉和变异等遗传算子也必须保证其运算结果所产生的新个体的基因值也在这个区间限制范围内。

十进制编码方法有下面几个优点：

（1）适合于在遗传算法中表示范围较大的数。

（2）适合于精度要求较高的遗传算法。

（3）便于较大空间的遗传搜索。

（4）改善了遗传算法的计算复杂性，提高了运算效率。

（5）便于遗传算法与经典优化方法的混合使用。

（6）便于设计针对问题的专门知识的知识型遗传算子。

（7）便于处理复杂的决策变量约束条件。

3.2 适应度函数

度量个体适应度的函数称为适应度函数，它是根据目标函数确定的用于区分群体中个体好坏的标准，是算法演化过程的驱动力，也是进行自然选择的唯一依据。

适应度函数总是非负的，任何情况下都希望其值越大越好（以极大值为目标进行阐述）。而目标函数可能有正有负，因此需要在目标函数与适应度函数之间进行变换。由解空间中某一点的目标函数 $f(x)$ 到搜索空间中对应个体的适应度函数值 $Fit(f(x))$ 的转换方法基本上有以下三种。

（1）直接将待求解的目标函数转化为适应度函数。

（2）将待求解的目标函数做适当处理后再转化为适应度函数。

若目标函数为最小化问题，则

$$Fit\left(f\left(x\right)\right) = \begin{cases} c_{\max} - f\left(x\right), f\left(x\right) < c_{\max} \\ 0, others \end{cases} \tag{3-1}$$

式中，$c_{\max}$ 为一个适当的相对比较大的数，是 $f(x)$ 的最大值估计，可以是一个合适的输入值。

若目标函数为最大化问题，则

$$Fit\left(f\left(x\right)\right)=\begin{cases} f\left(x\right)-c_{\min}, f\left(x\right)>c_{\min} \\ 0, \text{others} \end{cases} \tag{3-2}$$

式中，$c_{\min}$ 为一个适当的相对比较小的数，是 $f(x)$ 的最小值估计，可以是一个合适的输入值。

（3）若目标函数为最小问题，则

$$Fit\left(f\left(x\right)\right)=\frac{1}{1+c+f\left(x\right)} \tag{3-3}$$

（4）若目标函数为最大问题，则

$$Fit\left(f\left(x\right)\right)=\frac{1}{1+c-f\left(x\right)} \tag{3-4}$$

式(3-1) ~ 式(3-4)中，c 为目标函数界线的保守估计值。

3.3 选择算子

选择又称为复制，是在群体中选择生命力强的个体产生新的群体的过程。遗传算法使用选择算子来对群体中的个体进行优胜劣汰操作，根据每个个体的适应度大小选择，适应度较高的个体被遗传到下一代群体中的概率较大；反之亦然。这样就可以使得群体中个体的适应度值不断接近最优解。选择算子的确定的好坏，直接影响到遗传算法的计算结果。

下面介绍几种典型常用的选择算子：

1. 轮盘赌选择

在轮盘赌选择中，一个个体被选中的概率与其适应度相关。如果 f_i 表示第 i 个个体的适应度，总群体适应度为 $F=\sum_{i=1}^{N}f_i$，则第 i 个个体被选中的概率为：

$$P(i)=\frac{f_i}{F}$$

由于这种选择方法是随机操作的原因，误差比较大，有时甚至连适应度较高的个体也选择不上。

2. 随机竞争选择

随机竞争选择与轮盘赌选择基本一样。在随机竞争选择中，每次按轮盘赌选择

机制选取一对个体，然后让这两个个体进行竞争，适应度高的被选中，如此反复，直到选满为止。

3．随机遍历选择

随机遍历选择提供了零偏差和最小个体扩展。设定 npointer 为需要选择的个体数目，等距离选择个体，选择指针的距离为 $\frac{1}{\text{npointer}}$，第一个指针的位置由 $[1, \frac{1}{\text{npointer}}]$ 区间的均匀随机数决定。

4．排序选择

排序选择的主要思想是对群体中的所有个体按其适应度大小进行排序，基于这个排序来分配各个个体被选中的概率。

在前面所介绍的一些选择操作方法中，其选择依据主要是各个个体适应度的具体数值。一般要求它取非负值，这就使得我们在选择操作之前，必须先对一些负的适应度进行变换处理。而排序选择方法的主要着眼点是个体适应度之间的大小关系。对个体适应度是否取正值或负值以及个体适应度之间的数值差异程度并无特别要求。

排序选择方法具体操作过程如下。

首先对群体中的所有个体按其适应度大小进行降序排序，然后根据具体求解问题，设计一个概率分配方案，按上述排列次序将概率值分配给各个个体。再以各个个体所分配到的概率值作为其能够被遗传到下一代的概率，基于这些概率值用比例选择的方法来产生下一代种群。

排序选择方法的主要问题就是概率的分配问题，如常用的线性排序法概率分配如下：

种群中按适应度排在第 k 的染色体的选择概率为

$$p_k = q - (k-1) \times r \tag{3-5}$$

式(3-5)中，q 为最好染色体选择概率，q_0 为最坏染色体选择概率，参数 r 按下式确定

$$r = \frac{q - q_0}{pop_{size} - 1} \tag{3-6}$$

5．联赛选择

联赛选择也是一种基于个体适应度之间大小关系的选择法。

其基本思想是：每次选取几个个体之中适应度最高的一个个体遗传到下一代群体中。在联赛选择操作中，只有个体适应度之间的大小比较运算。而无个体适应度之间的算术运算，所以它对个体适应度是取正值还是取负值无特别要求。

联赛选择的具体操作过程是：首先从群体中随机选取 m 个个体进行适应度大小的比较，将其适应度最高的个体遗传到下一代。将上述过程重复 n 次，就可得到下一代群体中的 n 个个体。

3.4 交叉算子

在生物的自然进化过程中，两个同源染色体通过交配而重组，形成新的染色体，从而产生新的个体或物种，交配重组是生物遗传和进化过程中的一个主要环节。

在遗传算法中也使用交叉算子来产生新的个体。交叉运算是遗传算法区别于其他进化算法的重要特征，它在遗传算法中起着关键作用，是产生新个体的主要方法。

下面介绍几种适合于二进制编码个体或十进制编码个体的交叉算子。

1. 单点交叉

单点交叉（One-point Crossover) 又称为简单交叉，是最常用和最基本的交义操作算子。它以二值串中的随机选择点开始，对每一对相互配对的个体，依设定的交叉概率在其交叉点处相互交换两个个体的部分染色体，从而产生出两个新的个体。

2，两点交叉与多点交叉

两点交叉（Two-point Crossover）是指在个体编码串中随即设置两个交叉点，然后再进行部分基因交换。两点交叉的具体过程是：

（1）在相互配对的两个个体编码串中随即设置两个交叉点。

（2）交换两个个体在所设定的两个交义点之间的部分染色体。

可以将两点交叉的概念推广至多点。也就是指在个体编码串中随即设置多个交叉点，然后进行基因交换。多点交叉又称广义交叉，其操作过程与单点和双点交叉类似。

3. 均匀交叉

均匀交叉是指两个配对个体的每个基因座上的基因都以相同的交叉概率进行交换，从而形成两个新的个体。其具体运算可通过设置一屏蔽字来确定新个体的

各个基因如何由哪一个父代个体来提供。

均匀交叉的主要操作过程如下：

首先随机产生一个与个体编码串长度等长的屏蔽字：$W = w_1w_2w_3...w_i...w_L$，其中 L 是个体编码串长度。然后由下述规则从 A 和 B 两个父代个体中产生出两个新的子代个体 A' 和 B'。

(1) 若 $W_i = 0$，则才在第 i 个基因座上的基因值继承 A 的对应基因值，B' 在第 i 个基因座上的基因值继承 B 的对应基因值。

(2) 若 $W_i = 1$，则 A' 在第 i 个基因座上的基因值继承 B 的对应基因值，B' 在第 i 个基因座上的基因值继承 A 的对应基因值。

4. 算术交叉

算术交叉是指由两个个体的线性组合而产生出两个新的个体。为了能够进行线性组合运算，算术交叉的操作对象一般是浮点数编码所表示的个体。

3.5 变异算子

遗传算法中所谓的变异运算，是指将个体染色体编码串中的某些基因座上的基因值用该基因座的其他等位基因来替换，从而形成一个新的个体。变异是遗传算法生成新个体的主要方法之一，变异运算可以使算法在运行过程中维持种群的多样性，有效避免早熟，起到改善遗传算法局部搜索能力的作用。

遗传算法中变异算子也应根据不同的要求进行选择和设计，下面是几种常用的变异算子。

1. 基本位变异

基本位变异操作是指对个体编码串中以变异概率、随机指定的某一位或某几位基因座上的值做变异运算，其具体操作过程如下：

(1) 对个体的每一个基因座，依变异率 p_m 指定其为变异点。

(2) 对每一个指定的变异点，对其基因值做取反运算或用其他等位基因值代替，从而产生出一个新的个体。

2. 均匀变异

操作是指分别用符合某一范围内均匀分布的随机数，以某一较小的概率来替换个体染色体中各个基因上原有的基因值。

均匀变异的具体操作过程是：

（1）依次指定个体编码中的每个基因座为变异点。

（2）对每一个变异点，以变异概率从对应基因的取值范围内取一随机数来替代原来的基因值。

均匀变异操作特别适合应用于遗传算法的初期运行阶段，它使得搜索点可以在整个搜索空间内自由地移动，从而可以增加种群的多样性。

3. 边界变异

边界变异操作是上述均匀变异操作的一个变形。在进行边界变异操作时，随机地取基因座的两个对应边界基因值之一去替代原有的基因值。当变量的取值范围特别宽，并且无其他约束条件时，边界变异会带来不好的作用。但它特别适用于最优点位于或者接近于可行解的边界时的一类问题。

交叉和变异示意图如下所示：

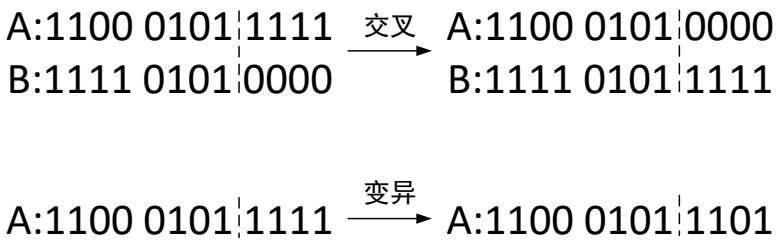

图 3-1　交叉变异示意图

3.6　控制参数的选择

遗传算法中有下面几个参数对遗传算法的运行有很大影响，分别是：个体编码串长度 l、群体大小 M、交叉概率 p_c、变异概率 p_m 和终止代数 T。

（1）编码串长度 l：使用二进制编码表示个体时，编码串长度 l 的选取与问题所要求的求解精度有关；使用浮点数编码来表示个体时，编码串长度 l 与决策变量的个数 n 相等；另外，也可使用变长度的编码来表示个体。

（2）群体大小 M：当 M 取值较小时，可提高遗传算法的运算速度，但却降低了群体的多样性，有可能会引起遗传算法的早熟现象；而当 M 取值较大时，又会使得遗传算法的运行效率降低。一般建议的取值范围是 2~100。

（3）交叉概率 p_c：交叉概率一般取值较大。但如果太小，它会破坏群体中的优良模式，对进化运算不利。一般建议的取值范围是 0.4~0.99。另外，也可使用自适

应的思想来确定交叉概率 p_c。

(4) 变异概率 p_m：若变异概率 p_m 取值太大，则容易破坏群体中的优良模式，使得遗传算法的搜索趋于随机性；若取值过小，则它产生新个体和抑制早熟的能力会较差。

一般建议的取值范围是 0.0001~0.1.

(5) 终止代数 T：终止代数 T 是表示遗传算法运行结束条件的一个参数，一般建议的取值范围是 100~1000。至于遗传算法的终止条件，还可以利用别的判定准则，如：当连续几代个体平均适应度的差异小于某个极小的阀值时或当群体中所有个体适应度的方差小于某一极小的阀值时。

3.7 约束条件处理

在遗传算法中必须对约束条件进行处理，但目前尚无处理各种约束条件的一般方法，根据具体问题可选择下列三种方法，即搜索空间限定法、可行解变换法和罚函数法。

1. 搜索空间限定法

搜索空间限定法的基本思想是对遗传算法的搜索空间的大小加以限制，使得搜索空间中表示一个个体的点与解空间中表示一个可行解的点有一一对应的关系。对一些比较简单的约束条件通过适当编码使搜索空间与解空间一一对应，限定搜索空间能够提高遗传算法的效率。在使用搜索空间限定法时必须保证交叉和变异之后的新个体在解空间中有对应解。

2. 可行解变换法

可行解变换法的基本思想是在由个体基因型到个体表现型的变换中，增加使其满足约束条件的处理过程，即寻找个体基因与个体表现型的多对一变换关系，扩大了搜索空间，使进化过程中所产生的个体总能通过这个变换而转化成解空间中满足约束条件的一个可行解。可行解变换法对个体的编码方法、交叉运算和变异运算等无特殊要求，但运行效率下降。

3. 罚函数法

罚函数法的基本思想是对在解空间中无对应可行解的个体计算其适应度时，减去一个罚函数，从而降低该个体的适应度，使该个体被遗传到下一代群体中的

概率减小。罚函数法可以用式对个体的适应度进行调整:

$$F'(x)=\begin{cases} F(x), \text{if } x \text{ satisfies the constraints} \\ F(x)-P(x), \text{if } x \text{ does not satisfy the constraints} \end{cases} \tag{3-7}$$

式(3-7)中，$F(x)$ 为原适应度函数；$F'(x)$ 为调整后的新适应度函数；$P(x)$ 为罚函数。

4 遗传算法的模式定理

遗传算法通过对群体中多个个体的迭代搜索来逐步找出问题的最优解。这个搜索过程是通过个体之间的优胜劣汰、交叉重组和突然变异等遗传操作来实现，这些操作从本质上而言包含了大量的随机性操作，但实质上，可以从数学机理来分析遗传算法的有效性和合理性。

遗传算法的数学理论基础就是模式定理。模式即种群中个体中的相似模块，它描述了在某些位置上具有相似结构特征的个体编码串的一个了集。

以二进制编码串为例，模式是基于三个字符集（0、1 和 *）的字符串，符号 * 代表任意字符，可以是 0 或 1。若模式 H=*10*，则串 0100、0101、1100 和 1101 是它的子集。

遗传算法的本质是对模式所进行的一系列运算，即通过选择算子将当前群体中的优良模式遗传到下一代群体中，通过交叉算子进行模式的重组，通过变异算子进行模式的突变。通过这些遗传运算，一些较差的模式逐步被淘汰，而一些较好的模式逐步被遗传和进化，最终就可达到问题的最优解，模式定理的定义式表达如下：

$$m(H,t+1) \geq m(H,t) \times \frac{f(H)}{\bar{f}} \times \left[1 - p_c \times \frac{\delta(H)}{l-1} - p_m \times O(H)\right] \tag{4-1}$$

式中 $m(H,t+1)$ 表示在 $t+1$ 代种群中存在模式 H 的个体数目；$m(H,t)$ 表示在 t 代种群中存在模式 H 的个体数目；$f(H)$ 表示在 t 代种群中包含模式 H 的个体平均适应度；$\bar{f}$ 表示在 t 代种群中所有个体的平均适应度：l 表示个体的长度；p_c 表示交叉概率；p_m 表示变异概率；$\delta(H)$ 表示模式 H 的定义距（定义距，第一个确定位置和最后一个确定位置之间的距离）；$O(H)$ 表示模式 H 的模式阶（模式阶，模式 H 中确定位置的个数）。

模式定理可以说明，适应度高、定义距小和模式阶低的模式在后代中可以快速增长。在搜索后期，大部分染色体都具有相同或类似的模式，简单的交叉操作对定义距小和模式阶低的模式改变不大；另一方面，如果变异率很小，则对这样的模式不会产生影响。所以，在后期局部搜索时，染色体进化速度很慢，即使繁殖了很多代，也可能达不到显著的进化效果。因此，遗传算法的一个大的缺点就是全部搜索能力不强。

模式定理是遗传算法的基本理论，为遗传算法的有效性提供了合理的理论依据。模式定理保证了较优的模式（遗传算法的较优解）的数目呈指数增长，同时也给出了模式在选择、交叉和变异作用下子代中产生个体数目的下限值。

5 遗传算法的改进

尽管遗传算法有许多优点，也有许多专家学者对遗传算法进行不断的研究，但目前存在的问题依然很多，如：

（1）适应度值标定方式多种多样，没有一个简洁和通用的方法，不利于对遗传算法的使用。

（2）遗传算法的早熟（即很快收敛到局部最优解而不是全局最优解）现象是迄今为止最难处理的关键问题。

（3）快要接近最优解时在最优解附近左右摆动，收敛较慢。

自从 1975 年 J.H.Holland 系统提出遗传算法的完整结构和理论以来，众多学者一直致力于推动遗传算法的发展，对编码方式、控制参数的确定和交叉机理等进行深入的研究，提出了各种变形的遗传算法。其基本途径概括起来主要有以下几个方面 [2]：

（1）改进遗传算法的组成成分或者使用技术，如选用优化控制参数和适合问题特性的编码技术等。

（2）采用混合遗传算法。

（3）采用动态自适应技术，在进化过程中调整算法控制参数和编码精度。

（4）采用非标准的遗传操作算子。

（5）采用并行计算。

5.1 适应度值标定

初始群体中可能存在特殊个体的适应度值超常（如很大）。为了防止其统治整个群体并误导群体的发展方向而使算法收敛于局部最优解，需限制其繁殖。在计算临近结束，遗传算法逐渐收敛时，由于群体中个体适应度值比较接近，继续优化选择较为困难，造成在最优解附近左右摇摆。此时应将个体适应度值加以放大，以提高选择能力，这就是适应度值的标定。

针对适应度值标定问题本章提出以下计算公式：

$$f' = \frac{1}{f_{\min} + f_{\max} + \delta}\left(f + |f_{\min}|\right) \tag{5-1}$$

式中，f' 为标定后的适应度值，f 为原适应度值，$f_{\max}$ 为适应度值的一个上界，

f_{min} 为适应度值的一个下界，δ 为开区间 (0,1) 内的一个正实数。

若 f_{max} 未知，可用当前代或目前为止的群体中的最大值来代替。若 f_{min} 未知，可用当前代或目前为止的群体中的最小值来代替。取 δ 的目的是防止分母为零和增加遗传算法的随机性。$|f_{min}|$ 是为了保证标定后的适应度值不出现负数。

由图 28 可知，若 f_{max} 与 f_{min} 差值越大，则角度 α 越小，即标定后的适应度值变化范围小，防止超常个体统治整个群体；反之则越大，标定后的适应度值变化范围增大，拉开群体中个体之间的差距，避免算法在最优解附近摆动现象发生。这样就可以根据群体适应度值放大或缩小，变更选择压力。

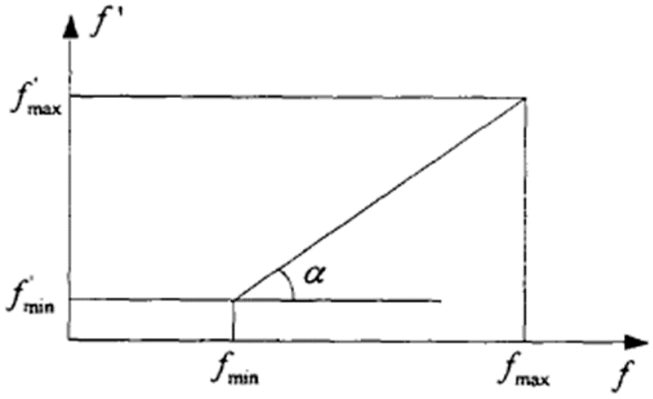

图 5-1　适应度值的标定

5.2　改进的自适应交叉变异率

遗传算法的参数中交叉概率 p_c 和变异概率 p_m 的选择是影响遗传算法行为和性能的关键所在，直接影响算法的收敛性。

对于交义概率 p_c，p_c 越大，新个体产生的速度就越快。然而，p_c 过大时遗传模式被破坏的可能性也越大，使得具有高适应度的个体结构很快就会被破坏；但是如果过小，会使得搜索过程缓慢，以至停滞不前。

对于变异概率 p_m，如果 p_m 过小，就不容易产生新的个体结构；如果 p_m 取值过大，那么遗传算法就变成了纯粹的随机搜索算法。针对不同的优化问题，需要反复实验来确定 p_c 和 p_m，这是一件繁琐的工作，而且很难找到适应于每个问题的最佳值。

对于 GA 中相关参数进行动态调整则称为自适应遗传算法，已有许多学者对自适应遗传算法进行了研究。自适应交叉和变异概率是根据个体的适应度值动态

的调整 p_c 和 p_m。当种群个体适应度值趋于一致时，适当增加 p_c 和 p_m，而当种群适应度值比较分散时，减小 p_c 和 p_m 的值。同时，对于适应度值高于群体平均适应度值的个体，采用较低的 p_c 和 p_m。反而对于适应度值低于群体平均适应度值的个体则采用较高的 p_c 和 p_m。因此，自适应的 p_c 和 p_m，能够提供相对某个解的最佳 p_c 和 p_m。

自适应遗传算法在保持群体多样性的同时，保证遗传算法的收敛性。可用下面两公式动态调整个体的交义变异概率。

$$p_c = \begin{cases} \dfrac{k_1\left(f_{\max} - f'\right)}{f_{\max} - f_{\min}}, f' \geq f_{avg} \\ k_2, f' < f_{avg} \end{cases} \tag{5-2}$$

$$p_m = \begin{cases} \dfrac{k_3\left(f_{\max} - f\right)}{f_{\max} - f_{avg}}, f \geq f_{avg} \\ k_4, f < f_{avg} \end{cases} \tag{5-3}$$

式中，$f_{\max}$ 为群体中最大的适应值；f_{avg} 为每代群体的平均适应值；f' 为要交叉的两个个体中较大的适应值；f 为要变异个体的适应值。

这里，只要设定 k_1, k_2, k_3 和 k_4 在 (0,1) 区间取值，就可以自适应调整了。

（1）当适应度值低于平均适应度值时，说明该个体是性能不好的个体，对它就可以采用较大的交叉率和变异率；如果适应度值高于平均适应度值，说明该个体性能优良，对它就可以根据其适应度值取相应的交叉率和变异率。

（2）当适应度值越接近最大适应度值时，交叉率和变异率就越小。

（3）当适应度值等于最大适应度值时，交叉率和变异率的值为零。

这种调整方法对于群体处于进化后期比较合适，但对于进化初期不利，因为进化初期群体中较优的个体几乎处于一种不发生变化的状态，而此时的优良个体不一定是优化的全局最优解，这容易使进化走向局部最优解的可能性增加。

为此，可做进一步的改进，使群体中最大适应度值的个体的交叉率和变异率不为零，分别提高到 p_{c2} 和 p_{m2}，经过上述改进，p_{c2} 和 p_{m2} 计算表达式如下：

$$p_c = \begin{cases} p_{c1} - \dfrac{\left(p_{c1} - p_{c2}\right)\left(f' - f_{avg}\right)}{f_{\max} - f_{avg}}, f' \geq f_{avg} \\ p_{c1}, f' < f_{avg} \end{cases} \tag{5-4}$$

$$p_m = \begin{cases} p_{m1} - \dfrac{(p_{m1} - p_{m2})(f_{\max} - f_{avg})}{f_{\max} - f}, f' \geq f_{avg} \\ p_{m1}, f < f_{avg} \end{cases} \tag{5-5}$$

6 基于改进遗传算法的道路交通信号优化

对城市道路交叉路口交通信号灯实施合理优化控制，有利于缓解日趋紧张的交通拥挤现象，提高交通效益。对于城市交通，由于道路上的交通车流呈现很大的随机性，车辆行驶过程是一种随机过程，因而实施相位控制也应针对不同的车流情况采取不同的方案。对交叉路口交通信号的优化控制，有以下几种方法：

（1）针对信号周期进行优化；

（2）针对相位信号配时（或绿信比）进行优化；

（3）针对周期和相位信号配时（或绿信比）同时进行优化，甚至还包括相位信号顺序的优化。

配时方案的改变，对各个车道的车流影响很大 [3]。目前，对于城市交通网络的优化控制研究，国内外的一些刊物刊载过一些有关文章，但大多是针对城市交通网路的交通流分配进行优化。也有个别文献提出了针对信号周期或信号时间区间进行优化，而所采用的优化方法大多为传统的优化方法，如黄金分割法、爬山法和网格搜索法等。但对于交叉口多相位交通信号配时优化控制还很少涉及，这一节将针对单交叉路口多相位的交通信号采用改进的遗传算法优化方法配时进行讨论。

6.1 城市交通信号控制优化问题分析

城市交通信号控制系统的控制对象是由各种车辆组成的及在被控制的区域内道路往上行驶的交通流。从控制理论角度来分析，无论是一个交叉口，还是由数以百计的路口组成的被控制区域，都是所谓的被控对象，都可以用同样形式的数学形式来表达其各种变量之间的关系，对一个交叉路口作为被控对象的情况，也是一个非线性的时变系统。

与一般的寻求极值的静态优化不同，交通控制的优化是动态优化问题。在交通控制的优化问题中，无论是状态变量还是控制变量都是随着时间在变化的，这些变量本身也是时间的函数，与静态优化不同，我们要从动态的概念来理解交通控制优化问题。

交通控制优化问题可以表述为：寻求一组随时间变化的控制变量时间 $U(k)$，使得被控区域的交通流从初始状态 $X^*(0)$ 按照使得某种性能指标最优的轨迹运动，在时刻转移到状态 $X^*(k)$。由于系统在优化过程中是不断迭代进行的，所以系统总

是处于在寻找到的优化状态之间不断地转移过程之中，所以时刻的状态表示成上标带 * 号的 $X^*(k)$。

对于非线性时变系统，如果系统的特性总在不断地随时间而变化，那么从理论上讲变量之间的关系就无法确定下来，或者说不是一种单值的确定性的对应关系，而是一种多值的和随机的对应关系。目前，通常采取以下两个办法来近似地研究和处理。

（1）假设时变系统的时变速度比较慢，比系统的输入和输出之间的动态过程慢得多，在这个假设条件下，就可以认为在一个小的时间区间内系统是时不变的。这样就可以借用静态优化算法来研究和处理问题。

（2）降低对问题解的要求，不必寻求最优解，退而求次优或满意解。在这个前提下，就可以放宽原问题的限制，针对与原问题近似的问题来优化，得到对原问题而言是次优解或满意解。

这种处理非线性时变系统思路完全适用于交通信号控制的优化问题，在不得已的情况下，寻求次优解和满意解已成为人们的共识，交通控制的优化问题正好是遇到了理论上解决不了的非线性时变系统的动态优化问题。

在交通控制优化的时间区间内，可以把交通控制系统看成是时不变的系统，可以应用静态优化的方法来研究和处理变量之间的映射关系。这就为解决交通控制优化问题提供了新的途径。

6.2 以车辆平均延误时间最小为目标的单交叉路口优化配时

城市道路单交叉路口一方面是构成线控和区域控制的基础，另一方面，即使将来实现线控和面控，但在线控和面控不能覆盖的区域还会有大量独立控制的交叉口存在，因此，针对单交叉路口信号的合理配时，是实现有效控制的关键，对单交叉路口信号配时的研究具有重要的意义。

在城市道路交叉路口信号优化配时中，关于交叉口交通效益的评价指标，国内外常用的有通行能力、饱和度、延误、服务水平、停车次数、油耗和排队长度等等。其中，车辆延误时间和车辆排队长度是使用频率相对较高的两个性能指标。

本文选用车辆延误时间为性能指标确定单交叉路口的绿信比。考虑到城市多个交义路口协调控制的需要，在同一子控制区内的各交叉口采用相同的信号周期长度，在一天中的某个时间段内其周期是相对固定的，动态调整的可能性很小（一

些路口可以采用双周期，具体是哪些路口要根据实时交通数据进行计算来确定）。因此，本文研究相对固定周期条件下，**针对交叉口交通流的实时变化情况的信号配时方案，以实现信号交叉口的优化控制，提高单交叉路口的车辆通行能力，并减小路口总的车辆延误时间**。

对于延误的计算模型有很多，这里根据本文研究对象的特征，选用 Webster 延误计算模型。Webster 延误计算模型是在统计平衡态理论的基础上得出的，在交叉路口交通量不是十分拥挤的条件下，即交通流饱和度较小的情况下比较准确，随着交通流饱和度由小于 1 逐渐趋近于 1 时，其得到的延误与实际估计的延误差异逐渐增大。这是因为车辆进入道口形成一种稳态的交通流（如泊松分布流）所需要的瞬态时间大于或者等于稳态时间。

6.3 非线性模型的建立

表 6-1 符号说明

符号	说明	单位
d	每辆车在交叉口的平均延误	s
D	4 相位平面交叉口的总延误	s
C	周期时长	s
t	绿灯时长	s
t_i	第 i 相位的绿灯时长	s
L	总的损失时间	s
e	每相位最短绿灯时间	s
g_e	有效绿时	s
λ	绿信比	
λ_i	第 i 相位绿信比	
pcu	Passenger Car Unit（标准车当量数）	
q	实际流量	pcu/h
q_{ij}	第 i 相位第 j 进口道的车流量	pcu/h
s	饱和流量	pcu/h
N	通行能力	pcu/h
y	流量比	
x	饱和度	
x_{ij}	第 i 相位第 j 进口道上的车流饱和度	

6.3.1 目标函数

以实时采集的路上交通流数据为基础，以一个周期内交叉口的车辆平均延误时间最少为目标，建立目标优化函数。采用 Webster 延误估算公式，每辆车在交叉口的平均延误为 d。

$$d=\frac{C(1-\lambda)^2}{2\left(1-\lambda x\right)}+\frac{x^2}{2q\left(1-x\right)}-0.65\left(\frac{C}{q^2}\right)^3 x^{(2+5\lambda)} \tag{6-1}$$

式(6-1)中，d——每辆车的平均延误 (s)；C——周期时长 (s)；λ——绿信比；q——流量 (pcu/h)；x——饱和度。

式(6-1)中的第一项是由于车辆到达的随机性引起的延误，称为随机 (Random) 延误：

$$d_u=\frac{C(1-\lambda)^2}{2\left(1-\lambda x\right)} \tag{6-2}$$

式(6-1)中的第二项是由车辆均衡到达交叉口而引起的延误，称为均匀 (Uniform) 延误：

$$d_r=\frac{x^2}{2q\left(1-x\right)} \tag{6-3}$$

式(6-1)中的第三项数值很小，在实际计算时，可忽略不计。这样就有：

$$d=\frac{C(1-\lambda)^2}{2\left(1-\lambda x\right)}+\frac{x^2}{2q\left(1-x\right)} \tag{6-4}$$

这里以典型的 4 相位平面交叉口为例，可以得出总的延误计算公式，即目标函数：

$$\min D=\sum_{i=1}^{4}\sum_{j=1}^{2}\left\{q_{ij}\left[\frac{C(1-\lambda_i)^2}{2\left(1-\lambda_i x_{ij}\right)}+\frac{{x_{ij}}^2}{2q_{ij}\left(1-x_{ij}\right)}\right]\right\} \tag{6-5}$$

式(6-5)中，q_{ij} 为第 i 相位第 j 进口道的车流量，(pcu/h)；x_{ij} 为第 i 相位第 j 进口道上的车流饱和度：λ_i 为第 i 相位的绿信比，$\lambda_i=\dfrac{t_i}{C}$。

6.3.2 约束条件

城市交通信号的优化过程是针对交叉路口 4 个相位的有效绿灯时间进行实时优化，这里要满足一定的约束条件。$t_1 + t_2 + t_3 + t_4 = C - L$ 考虑交叉口行人过马路时的安全需要，每相位最短绿灯时间不得小于某值 e（这里取最小绿灯时间为 10(s)），因此每一相位的配时须满足条件：

$$e \leq C - L - 3\times 10 \tag{6-6}$$

式(6-6)中，L 为总的损失时间 (s)。

考虑到最大饱和度约束，合理的信号配时设计及某时段内周期的合理给定应能保证在合理正确的信号配时情况下，各相位的饱和度均不过大，避免造成交叉口道出现交通拥堵的现象。

本文假定各相位各交叉路口的饱和度 x 均不大于 0.85:

$$x = \frac{q}{N} = \frac{q}{s\dfrac{g_e}{C}} = \frac{Cq}{sg_e} \leq 0.85 \tag{6-7}$$

式(6-7)中，q 为实际流量 (pcu/h)，N 为通行能力 (pcu/h)，$N = s\dfrac{g_e}{C}$，s 为饱和流量 (pcu/h)，g_e 为有效绿时 (s)，即：

$$g_{e_i} \geq \frac{Cq}{0.85s} = \frac{Cy}{0.85} \tag{6-8}$$

式(6-8)中，y 为流量比，$y = \dfrac{q}{s}$。对每一相位，均将其最大的 y 值代入得到每相位的最小绿灯时间要求：

$$t_i = g_{e_i} \geq \frac{Cy_{i,\max}}{0.85} \tag{6-9}$$

综上所述约束条件如下：

$$\begin{cases} \sum\limits_{i=1}^{4} t_i = C - L \\ e \leq t_i \leq C - L - 3 \times 10 \\ t_i = g_{e_i} \geq \dfrac{Cy_{i,\max}}{0.85} \end{cases} \tag{6-10}$$

6.4 仿真分析

某 4 相位交叉口，各相车流如图6-1所示。为研究问题的简便，假设直行和右转公用一个相位，即红灯时间禁止车辆左转，这种禁止措施对行人安全有利。

4 相位交叉口的信号控制交通流图如图6-1所示：

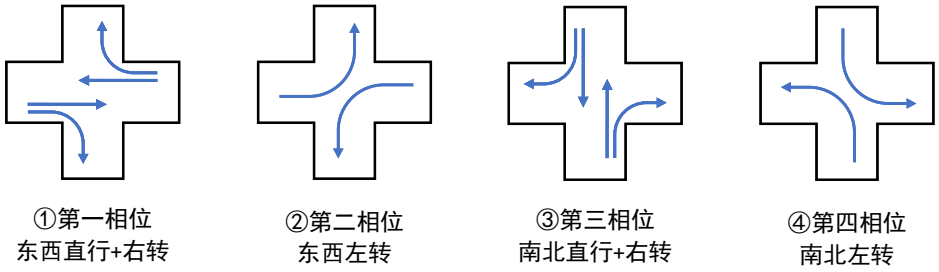

图 6-1 相位信号控制交通流图

4 相位交叉口的信号周期示意图如图6-2所示：

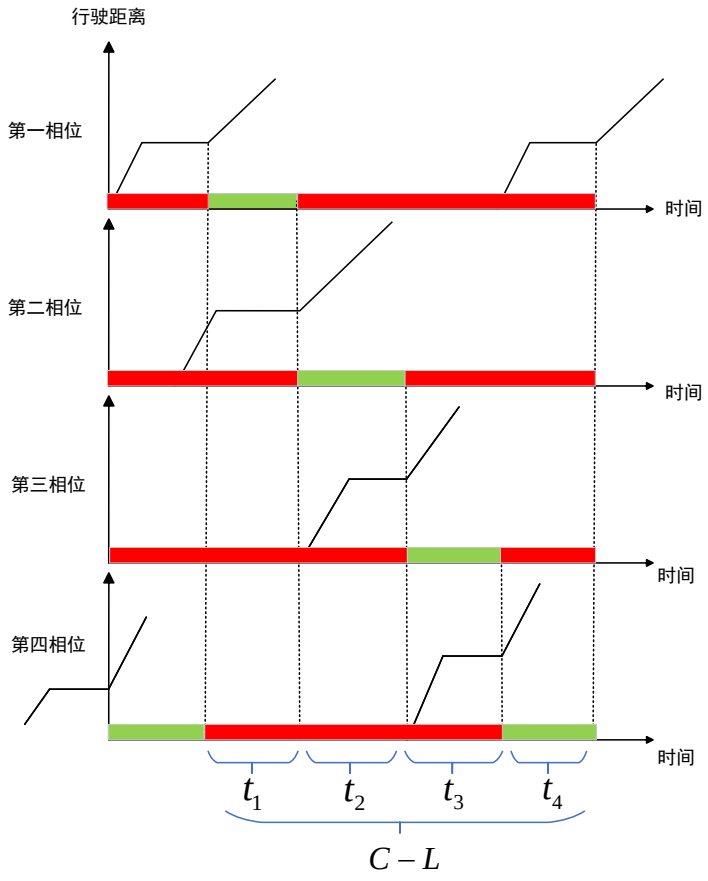

图 6-2 信号周期示意图

某交叉口各进口道车流量有关数据如表6-2所示。

表 6-2　某交叉口各进口道车流量有关数据

相位	进口道	交通流量 q	饱和流量 s	流量比 y
第一相位	东进口	368	2000	0.184
	西进口	462	2000	0.231
第二相位	东进口	152	960	0.158
	西进口	121	960	0.126
第三相位	南进口	311	1800	0.173
	北进口	360	1800	0.200
第四相位	南进口	128	960	0.133
	北进口	115	960	0.120

假设信号周期为 C =140(s)，其中总的损失时间为 10(s)，各相位最小绿灯时间为 10(s)。假设各相位黄灯时间和损失时间相等，则有效绿灯时间即为实际的绿灯时间。

在不改变交叉口几何条件的情况下，根据表6-2所示的交通量，利用所建立的数学模型及解法，确定各相位有效绿灯时长。

由最大饱和度限制得到各相位最短绿灯时间：

$$t_i = g_{e_i} \geq \frac{Cy_{i,\max}}{0.85} \tag{6-11}$$

对于第 1 相位，取较大的 y 值代入，得 $g_{e_1} \geq 38(s)$，同样其他各相位取相应的流量比可以得到，$g_{e_2} \geq 26(s)$，$g_{e_3} \geq 33(s)$，$g_{e_4} \geq 22(s)$

将有关数据代入约束条件式，整理后且将等式约束化为不等式约束，得到 4 个相位的绿灯时间的**约束条件**如下：

$$\begin{cases} t_1 + t_2 + t_3 + t_4 = C - L = 130 \\ 38 \leq t_1 \leq 130 - 26 - 33 - 22 = 59 \\ 26 \leq t_2 \leq 130 - 38 - 33 - 22 = 37 \\ 33 \leq t_3 \leq 130 - 38 - 26 - 22 = 44 \\ 22 \leq t_4 \leq 130 - 38 - 26 - 33 = 33 \end{cases} \tag{6-12}$$

在此约束条件下，要使得此交叉路口的总延迟时间最小，故相应的**目标函数**如下：

$$\min D = \sum_{i=1}^{4}\sum_{j=1}^{2}\left\{q_{ij}\left[\frac{C(1-\lambda_i)^2}{2\left(1-\lambda_i x_{ij}\right)} + \frac{{x_{ij}}^2}{2q_{ij}\left(1-x_{ij}\right)}\right]\right\} \tag{6-13}$$

其中，$\lambda_i = \dfrac{t_i}{C}, C = 140$。

决策变量是四个相位的绿灯时间 t_1, t_2, t_3, t_4。

建立优化的数学模型后，分别采用基本遗传算法和改进的遗传算法进行优化。

6.4.1 基本遗传算法

采用基本遗传计算，参数设置如下：

```
%% GA
%% 清空环境变量
clc,clear,close all
warning off
feature jit off
%% 遗传算法参数初始化
maxgen = 50;                          % 进化代数，即迭代次数
sizepop = 50;                         % 种群规模
pcross = [0.7];                        % 交叉概率选择，0和1之间
pmutation = [0.01];                    % 变异概率选择，0和1之间
% 城市交通信号系统参数
C = 140;
L = 10;
load('data.mat')  % 包含交通流量q以及饱和流量xij
q = q./3600;      % 转化为秒s
xij = xij./3600;  % 转化为秒s
%染色体设置
lenchrom=ones(1,3);     % t1、t2、t3
bound=[38,59;26,37;33,44];    % 数据范围
%----------------------------种群初始化
    ------------------------------------
individuals=struct('fitness',zeros(1,sizepop), 'chrom',[]);  %将种
    群信息定义为一个结构体
avgfitness = [];                      %每一代种群的平均适应度
bestfitness = [];                     %每一代种群的最佳适应度
bestchrom = [];                       %适应度最好的染色体
```

Listing 1: 参数设置

相应的目标函数即适应度函数如下：

```
function [f] = fun(x)
% 城市交通信号系统参数
C = 140;  % 信号周期
L = 10;   % 总损失时间
load('data.mat')  % 包含交通流量q以及饱和流量xij
```

```
q = q./3600;        % 转化为秒s
xij = xij./3600;  % 转化为秒s
%该函数用来计算适应度值
t1 = x(1);
t2 = x(2);
t3 = x(3);
t4 = C-L - t1-t2-t3;
lamda(1) = t1/C; % 为第1相位的绿信比
lamda(2) = t2/C; % 为第2相位的绿信比
lamda(3) = t3/C; % 为第3相位的绿信比
lamda(4) = t4/C; % 为第4相位的绿信比
f = 0;          % 适应度值初始化
for i=1:4
    for j=1:2
        f = f + ( C*(1-lamda(i)).^2/2/(1-lamda(i)*xij(i,j)) + xij(i
            ,j).^2/2/q(i,j)/(1-xij(i,j)) )*q(i,j);
    end
end
f = abs(f);
```

Listing 2: 目标函数即适应度函数

初始化种群如下:

```
%% 初始化种群
for i=1:sizepop
    % 随机产生一个种群
    individuals.chrom(i,:)=Code(lenchrom,bound); % 编码 (binary和
        grey的编码结果为一个实数, float的编码结果为一个实数向量)
    x=individuals.chrom(i,:);
    % 计算适应度
    individuals.fitness(i)=fun(x);    % 染色体的适应度
end
```

Listing 3: 初始化种群

迭代循环程序如下:

```
%% 找最好的染色体
[bestfitness bestindex] = min(individuals.fitness);
bestchrom = individuals.chrom(bestindex,:);     % 最好的染色体
% 记录每一代进化中最好的适应度和平均适应度
trace = [bestfitness];

%% 迭代求解最佳初始阀值和权值
```

```
% 进化开始
for i=1:maxgen
    disp(['迭代次数:    ',num2str(i)])
    % 选择
    individuals=Select(individuals,sizepop);
    % 交叉
    individuals.chrom=Cross(pcross,lenchrom,individuals.chrom,
        sizepop,bound);
    % 变异
    individuals.chrom=Mutation(pmutation,lenchrom,individuals.chrom
        ,sizepop,i,maxgen,bound);
    % 计算适应度
    for j=1:sizepop
        x=individuals.chrom(j,:);          % 解码
        individuals.fitness(j)=fun(x);    % 染色体的适应度
    end

    % 找到最小和最大适应度的染色体及它们在种群中的位置
    [newbestfitness,newbestindex]=min(individuals.fitness);
    [worestfitness,worestindex]=max(individuals.fitness);
    % 代替上一次进化中最好的染色体
    if bestfitness>newbestfitness
        bestfitness=newbestfitness;
        bestchrom=individuals.chrom(newbestindex,:);
    end
    individuals.chrom(worestindex,:)=bestchrom; % 剔除最差个体
    trace=[trace;bestfitness]; %记录每一代进化中最好的适应度
end
x = [bestchrom, C-L-sum(sum(bestchrom))]        % 最佳个体值
D = trace(end)        % 延误误差D
E = D./sum(sum(q));  % 平均延误E

%% 遗传算法结果分析
figure('color',[1,1,1])
plot(1:length(trace),trace(:,1),'b--');
title(['适应度曲线  ' '终止代数=' num2str(maxgen)]);
xlabel('进化代数');    ylabel('适应度');
legend('fz最佳适应度');
```

Listing 4: 迭代循环程序

其中，染色体是否合格检验，函数如下：

```
function flag=test(lenchrom,bound,code)
```

```
% lenchrom   input : 染色体长度
% bound      input : 变量的取值范围
% code       output: 染色体的编码值
t=code; %先解码
C = 140;  % 信号周期
L = 10;   % 总损失时间
t4 = C-L - t(1)-t(2)-t(3);
flag=1;
if (t(1)<bound(1,1))||(t(2)<bound(2,1))||(t(3)<bound(3,1))||(t(1)>
    bound(1,2))||(t(2)>bound(2,2))||(t(3)>bound(3,2))||t4<22||t4>33
     flag=0;
end
```

Listing 5: 检验染色体函数

从当前种群中选择适应度高和淘汰适应度低的个体的操作过程叫选择。

选择操作的主要作用是避免有效基因的损失，其目的是以更大的概率使得优化的个体（或解）生存下来，从而提高计算效益和全局收敛性。选择操作是遗传算法中极其重要的一个环节，它是建立在群体中个体的适应度评估基础上进行的。选择操作的实现方式有很多，在遗传算法中一般采取概率选择，概率选择是根据个体的适应度函数的值来进行的，适应度高的个体被选中的概率也大。

相应的选择算子函数如下：

```
function ret=select(individuals,sizepop)
% 该函数用于进行选择操作
% individuals input     种群信息
% sizepop     input     种群规模
% ret         output    选择后的新种群

%求适应度值倒数
fitness1=1./individuals.fitness; %individuals.fitness为个体适应度值

%个体选择概率
sumfitness=sum(fitness1);
sumf=fitness1./sumfitness;

%采用轮盘赌法选择新个体
index=[];
for i=1:sizepop    %sizepop为种群数
    pick=rand;
    while pick==0
        pick=rand;
```

```
    end
    for i=1:sizepop
        pick=pick-sumf(i);
        if pick<0
            index=[index i];
            break;
        end
    end
end

%新种群
individuals.chrom=individuals.chrom(index,:);    %individuals.chrom
    为种群中个体
individuals.fitness=individuals.fitness(index);
ret=individuals;
```

Listing 6: 选择算子函数

遗传算法中的交叉操作的作用是组合出新的个体，方法是将相互配对的染色体按某种方式相互交换其部分基因。遗传算法区别于其他进化算法的重要特征是交叉运算，它在遗传算法中起着关键作用。

交叉操作根据交叉算子将种群中的两个个体以一定的概率随机地在某些基因位进行基因交换，从而产生新的个体。其目的是获得下一代的优良个体，提高遗传算法的搜索能力。

交叉算子函数如下：

```
function ret=Cross(pcross,lenchrom,chrom,sizepop,bound)
%本函数完成交叉操作
% pcorss                    input  : 交叉概率
% lenchrom                  input  : 染色体的长度
% chrom      input  : 染色体群
% sizepop                   input  : 种群规模
% ret                       output : 交叉后的染色体
 for i=1:sizepop  %每一轮for循环中，可能会进行一次交叉操作，染色体
    是随机选择的，交叉位置也是随机选择的，%但该轮for循环中是否进行
    交叉操作则由交叉概率决定（continue控制）
     % 随机选择两个染色体进行交叉
     pick=rand(1,2);
     while prod(pick)==0
         pick=rand(1,2);
     end
     index=ceil(pick.*sizepop);
```

```
15      % 交叉概率决定是否进行交叉
16      pick=rand;
17      while pick==0
18          pick=rand;
19      end
20      if pick>pcross
21          continue;
22      end
23      flag=0;
24      while flag==0
25          % 随机选择交叉位
26          pick=rand;
27          while pick==0
28              pick=rand;
29          end
30          pos=ceil(pick.*sum(lenchrom)); %随机选择进行交叉的位置，即
                选择第几个变量进行交叉，注意：两个染色体交叉的位置相同
31          pick=rand; %交叉开始
32          v1=chrom(index(1),pos);
33          v2=chrom(index(2),pos);
34          chrom(index(1),pos)=pick*v2+(1-pick)*v1;
35          chrom(index(2),pos)=pick*v1+(1-pick)*v2; %交叉结束
36          flag1=test(lenchrom,bound,chrom(index(1),:)); %检验染色体
                1的可行性
37          flag2=test(lenchrom,bound,chrom(index(2),:)); %检验染色体
                2的可行性
38          if   flag1*flag2==0
39              flag=0;
40          else flag=1;
41          end    %如果两个染色体不是都可行，则重新交叉
42      end
43  end
44  ret=chrom;
```

Listing 7: 交叉算子函数

变异运算是染色体上某等位基因发生的突变现象，是产生新个体的另一种方法。变异是指染色体编码串以一定概率选择基因在染色体的位置，通过改变基因值来形成新的个体的操作，它改变了染色体的结构和物理形状。变异的主要目的是维持群体的多样性，防止出现未成熟收敛现象，此外还能使遗传算法具有局部的随机搜索能力。

变异算子函数如下:

```
function ret=Mutation(pmutation,lenchrom,chrom,sizepop,num,maxgen,
    bound)
% 本函数完成变异操作
% pcorss                input  : 变异概率
% lenchrom              input  : 染色体长度
% chrom      input  : 染色体群
% sizepop               input  : 种群规模
% opts                  input  : 变异方法的选择
% pop                   input  : 当前种群的进化代数和最大的进化代数
    信息
% bound                 input  : 每个个体的上届和下届
% maxgen                input  : 最大迭代次数
% num                   input  : 当前迭代次数
% ret                   output : 变异后的染色体

for i=1:sizepop   %每一轮for循环中，可能会进行一次变异操作，染色体
    是随机选择的，变异位置也是随机选择的，
    %但该轮for循环中是否进行变异操作则由变异概率决定（continue控
        制）
    % 随机选择一个染色体进行变异
    pick=rand;
    while pick==0
        pick=rand;
    end
    index=ceil(pick*sizepop);
    % 变异概率决定该轮循环是否进行变异
    pick=rand;
    if pick>pmutation
        continue;
    end
    flag=0;
    num = 0;
    chrom1 = chrom(i,:);
    while flag==0&&num<=20
        % 变异位置
        pick=rand;
        while pick==0
            pick=rand;
        end
        pos=ceil(pick*sum(lenchrom));  %随机选择了染色体变异的位
            置，即选择了第pos个变量进行变异
```

```
37
38          pick=rand; %变异开始
39          fg=(rand*(1-num/maxgen))^2;
40          if pick>0.5
41              chrom(i,pos)=chrom(i,pos)+(bound(pos,2)-chrom(i,pos))*
                    fg;
42          else
43              chrom(i,pos)=chrom(i,pos)+(chrom(i,pos)-bound(pos,1))*
                    fg;
44          end   %变异结束
45          flag=test(lenchrom,bound,chrom(i,:));      %检验染色体的可行
                性
46          num = num+1;  % 检验次数设置
47      end
48      if num>20           % 如果大于20次, 则不变异
49          chrom(i,:) = chrom1;
50      end
51  end
52  ret=chrom;
```

Listing 8: 变异算子函数

运行程序输出四个相位的绿灯时间 t_1, t_2, t_3, t_4 和总延迟时间 D 结果如下:

```
x =

   47.5316   26.8240   33.2607   22.3838


D =

   25.8114
```

图 6-3　适应度值变化曲线

得到如图 6-4所示的适应度值变化曲线图。

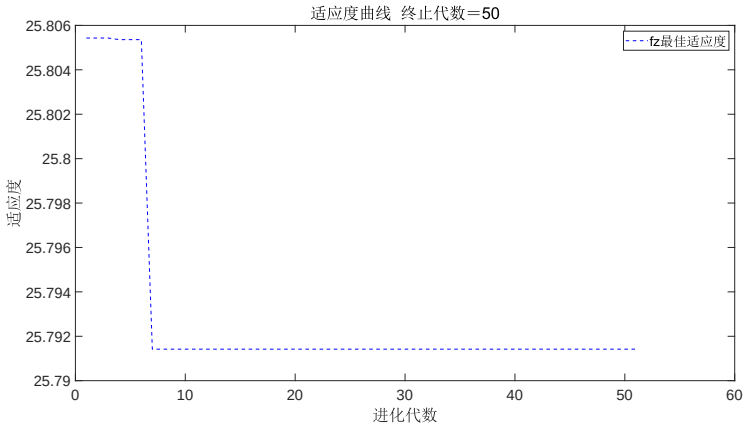

图 6-4　适应度值变化曲线

6.4.2　改进的遗传算法

针对适应度值标定问题本章提出以下计算公式:

$$f' = \frac{1}{f_{\min} + f_{\max} + \delta}(f + |f_{\min}|) \tag{6-14}$$

式中，f' 为标定后的适应度值，f 为原适应度值，$f_{\max}$ 为适应度值的一个上界，$f_{\min}$ 为适应度值的一个下界，δ 为开区间 (0,1) 内的一个正实数。

则程序中适应度变化如下:

迭代循环程序如下:

```
%% 迭代求解最佳初始阀值和权值
% 进化开始
for  i=1:maxgen
```

```
    disp(['迭代次数:    ',num2str(i)])
    % 选择
    individuals=Select(individuals,sizepop);
    % 交叉
    individuals.chrom=Cross(pcross,lenchrom,individuals.chrom,
        sizepop,bound);
    % 变异
    individuals.chrom=Mutation(pmutation,lenchrom,individuals.chrom
        ,sizepop,i,maxgen,bound);

    % 计算适应度
    for j=1:sizepop
        x=individuals.chrom(j,:);          % 解码
        individuals.fitness(j)=fun(x);    % 染色体的适应度
    end
    fmax = max(individuals.fitness);      % 适应度最大值
    fmin = min(individuals.fitness);      % 适应度最小值
    favg = mean(individuals.fitness);  % 适应度平均值
    individuals.fitness = (individuals.fitness + abs(fmin))./(fmax+
        fmin+delta); %适应度标定

    % 找到最小和最大适应度的染色体及它们在种群中的位置
    [newbestfitness,newbestindex]=min(individuals.fitness);
    [worestfitness,worestindex]=max(individuals.fitness);
    % 代替上一次进化中最好的染色体
    if bestfitness>newbestfitness
        bestfitness=newbestfitness;
        bestchrom=individuals.chrom(newbestindex,:);
    end
    individuals.chrom(worestindex,:)=bestchrom; % 剔除最差个体
    trace=[trace;bestfitness]; %记录每一代进化中最好的适应度
end
x = [bestchrom, C-L-sum(sum(bestchrom))]        % 最佳个体值
D = fun(bestchrom)      % 延误误差D
E = D./sum(sum(q));      % 平均延误E

%% 遗传算法结果分析
figure('color',[1,1,1]),
plot(1:length(trace),trace(:,1),'b--');
title(['适应度曲线  ' '终止代数=' num2str(maxgen)]);
xlabel('进化代数');    ylabel('适应度');
legend('fz最佳适应度');
```

Listing 9: 迭代循环程序

自适应遗传算法在保持群体多样性的同时，保证遗传算法的收敛性。可用下面两公式动态调整个体的交义变异概率。

$$p_c = \begin{cases} \dfrac{k_1 (f_{\max} - f')}{f_{\max} - f_{\min}}, f' \geq f_{avg} \\ k_2, f' < f_{avg} \end{cases} \tag{6-15}$$

$$p_m = \begin{cases} \dfrac{k_3 (f_{\max} - f)}{f_{\max} - f_{avg}}, f \geq f_{avg} \\ k_4, f < f_{avg} \end{cases} \tag{6-16}$$

式中，$f_{\max}$ 为群体中最大的适应值；f_{avg} 为每代群体的平均适应值；f' 为要交叉的两个个体中较大的适应值；f 为要变异个体的适应值。

这里，只要设定 k_1, k_2, k_3 和 k_4 在 (0,1) 区间取值，就可以自适应调整了。

所以交叉算子程序可修改为：

```
function ret=Cross(pcross,lenchrom,chrom,sizepop,bound)
%本函数完成交叉操作
% pcorss                input  : 交叉概率
% lenchrom              input  : 染色体的长度
% chrom      input  : 染色体群
% sizepop               input  : 种群规模
% ret                   output : 交叉后的染色体
k1 = 0.6;   k2 = 0.7;
k3 = 0.001;  k4 = 0.01;
% 计算适应度
for j=1:sizepop
    x=chrom(j,:);     % 解码
    f(j)=fun(x);      % 染色体的适应度
end
fmax = max(f);        % 适应度最大值
fmin = min(f);        % 适应度最小值
favg = mean(f);       % 适应度平均值

 for i=1:sizepop  %每一轮for循环中，可能会进行一次交叉操作，染色体是随机选择的，交叉位置也是随机选择的，%但该轮for循环中是否进行交叉操作则由交叉概率决定（continue控制）
     % 随机选择两个染色体进行交叉
     pick=rand(1,2);
```

```
    while prod(pick)==0
        pick=rand(1,2);
    end
    index=ceil(pick.*sizepop);

    f1 = fun( chrom(index(1),:) );  % 个体适应度值
    f2 = fun( chrom(index(2),:) );  % 个体适应度值
    f3 = max(f1,f2);                % 两者中大者
    if f3>=favg
        pcross = k1*(fmax - f3)./(fmax-favg);
    else
        pcross = k2;
    end

    % 交叉概率决定是否进行交叉
    pick=rand;
    while pick==0
        pick=rand;
    end
    if pick>pcross
        continue;
    end
    flag=0;
    while flag==0
        % 随机选择交叉位
        pick=rand;
        while pick==0
            pick=rand;
        end
        pos=ceil(pick.*sum(lenchrom)); %随机选择进行交叉的位置，即选择第几个变量进行交叉，注意：两个染色体交叉的位置相同
        pick=rand; %交叉开始
        v1=chrom(index(1),pos);
        v2=chrom(index(2),pos);
        chrom(index(1),pos)=pick*v2+(1-pick)*v1;
        chrom(index(2),pos)=pick*v1+(1-pick)*v2; %交叉结束
        flag1=test(lenchrom,bound,chrom(index(1),:)); %检验染色体1的可行性
        flag2=test(lenchrom,bound,chrom(index(2),:)); %检验染色体2的可行性
        if  flag1*flag2==0
            flag=0;
        else flag=1;
```

```
62            end    %如果两个染色体不是都可行，则重新交叉
63        end
64    end
65    ret=chrom;
```

Listing 10: 修改后的交叉算子程序

变异算子函数修改为:

```
1   function ret=Mutation(pmutation,lenchrom,chrom,sizepop,num,maxgen,
        bound)
2   % 本函数完成变异操作
3   % pcorss                input  : 变异概率
4   % lenchrom              input  : 染色体长度
5   % chrom      input  : 染色体群
6   % sizepop               input  : 种群规模
7   % opts                  input  : 变异方法的选择
8   % pop                   input  : 当前种群的进化代数和最大的进化代数
        信息
9   % bound                 input  : 每个个体的上届和下届
10  % maxgen                input  : 最大迭代次数
11  % num                   input  : 当前迭代次数
12  % ret                   output : 变异后的染色体
13  k1 = 0.6;    k2 = 0.7;
14  k3 = 0.001; k4 = 0.01;
15  % 计算适应度
16  for j=1:sizepop
17      x=chrom(j,:);     % 解码
18      f(j)=fun(x);      % 染色体的适应度
19  end
20  fmax = max(f);        % 适应度最大值
21  fmin = min(f);        % 适应度最小值
22  favg = mean(f);       % 适应度平均值
23
24  for i=1:sizepop   %每一轮for循环中，可能会进行一次变异操作，染色体
        是随机选择的，变异位置也是随机选择的，
25      %但该轮for循环中是否进行变异操作则由变异概率决定（continue控
            制）
26      % 随机选择一个染色体进行变异
27      pick=rand;
28      while pick==0
29          pick=rand;
30      end
31      index=ceil(pick*sizepop);
```

```
32
33      f1 = fun( chrom(index(1),:) );  % 个体适应度值
34      f3 = max(f1);                   % 两者中大者
35      if f3>=favg
36          pmutation = k3*(fmax - f3)./(fmax-favg);
37      else
38          pmutation = k4;
39      end
40
41    % 变异概率决定该轮循环是否进行变异
42     pick=rand;
43     if pick>pmutation
44         continue;
45     end
46     flag=0;
47    num = 0;
48     chrom1 = chrom(i,:);
49     while flag==0&&num<=20
50         % 变异位置
51         pick=rand;
52         while pick==0
53             pick=rand;
54         end
55         pos=ceil(pick*sum(lenchrom));  %随机选择了染色体变异的位
               置，即选择了第pos个变量进行变异
56
57         pick=rand; %变异开始
58         fg=(rand*(1-num/maxgen))^2;
59         if pick>0.5
60             chrom(i,pos)=chrom(i,pos)+(bound(pos,2)-chrom(i,pos))*
                   fg;
61         else
62             chrom(i,pos)=chrom(i,pos)-(chrom(i,pos)-bound(pos,1))*
                   fg;
63         end   %变异结束
64         flag=test(lenchrom,bound,chrom(i,:));      %检验染色体的可行
               性
65          num = num+1;  % 检验次数设置
66     end
67     if num>20           % 如果大于20次，则不变异
68         chrom(i,:) = chrom1;
69     end
70 end
```

```
71 ret=chrom;
```

Listing 11: 修改后的变异算子函数

运行程序输出四个相位的绿灯时间 t_1, t_2, t_3, t_4 和总延迟时间 D 结果如下:

```
x =

   45.6082   26.2883   36.0038   22.0997


D =

   25.7926
```

图 6-5　运行程序输出结果

可见使用改进的遗传算法可使延迟 D 更低。

但程序的运行结果和 k_1, k_2, k_3 和 k_4 的设定有关，不同的取值可能导致不一样的效果。

得到如图 6-6所示的适应度值变化曲线图。

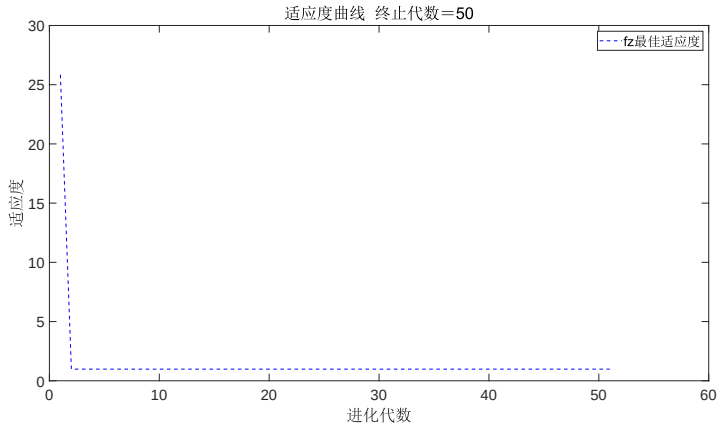

图 6-6　适应度值变化曲线

6.4.3　对比显示

对比简单遗传算法和改进的遗传算法, 程序如下:

```
1 %% 改进的GA
2 %% 清空环境变量
3 clc,clear,close all % 清除变量空间
4 warning off         % 消除警告
```

```
feature jit off  % 加速代码执行
ysw1
hold on
ysw2
%% 改进的遗传算法结果分析
plot(1:length(trace),trace(:,1),'b--');
title(['适应度曲线  ' '终止代数=' num2str(maxgen)]);
xlabel('进化代数');    ylabel('适应度');
legend('fz最佳适应度');
```

Listing 12: 对比显示程序

运行程序输出图形如图 28-6 所示。

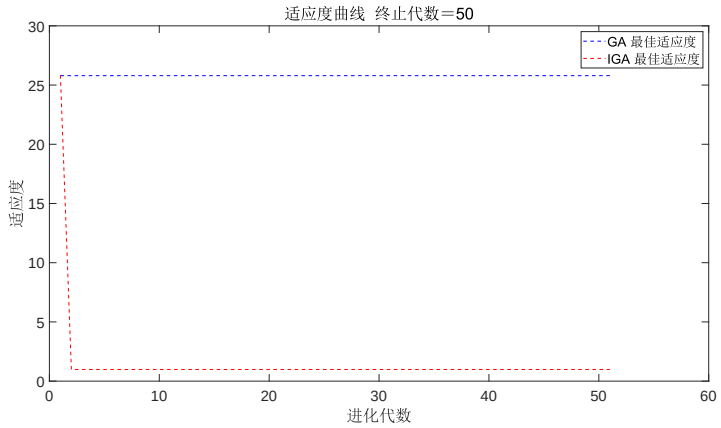

图 6-7　适应度值变化曲线

可见，改进的遗传算法在收敛性好于简单遗传算法，且不易于陷入局部最优，然而改进的遗传算法耗时比较长，这也是改进的同时带来的缺陷，在实际应用中，应该合理均衡算法效果。

7 结论

遗传算法是模拟自然界遗传机制和生物进化论而成的一种随机搜索优化方法,由于其隐含并行性和较强的全局搜索特性,使其具有其他常规优化算法无法拥有的优点。

本研究在分析城市道路单交叉路口交通流特性的基础上，成功建立了以车辆平均延误时间最短为目标的非线性函数模型，并利用改进的遗传算法对模型进行了求解，得到了在固定周期下的最优配时方案。仿真结果表明，该方案在缓解城市交通拥堵方面取得了理想的效果。

本研究的成果为城市交通管理提供了有益的参考，为优化交通系统操作提供了可行性方案。未来的工作可以进一步完善模型，考虑更多的影响因素，如交通信号的协调、实时交通信息的应用等，以进一步提高交通管理的效率和效果。这项研究的成果为构建智慧城市交通管理系统奠定了基础，并有望为城市交通管理领域的研究和实践提供有益的启示。

参考文献

[1] 王小川. *MATLAB* 神经网络 *43* 个案例分析. 北京航空航天大学出版社, 2013.

[2] 余胜威. *MATLAB* 优化算法案例分析与应用. 进阶篇. 清华大学出版社, 2015.

[3] 王伟平. 城市平面交叉口交通信号控制优化方法的研究. Master’s thesis, 山东科技大学, 2005.

虚拟仿真实验:

实验名称	姓名	实验结果	实验成绩	实验开始时间	实验结束时间	实验用时	操作
复杂地质钻进过程智能优化控制虚拟仿真实验	曾康慧	完成	77	2024-07-02 19:34:44	2024-07-02 21:15:44	101	步骤详情 下载报告

1 共 1 条

图 7-1　实验截图