$$\sum_k y_{ki}=1, i=1,2,\cdots,N \tag{6-15}$$

$$\sum_i x_{ijk}=y_{kj}, j\in V, \forall k \tag{6-16}$$

$$\sum_j x_{ijk}=y_{ki}, i\in V, \forall k \tag{6-17}$$

$$\sum_k y_{0k}=k, \forall k \tag{6-18}$$

$$s_0=0 \tag{6-19}$$

$$s_i+t_i+t_{ij}=s_j, i,j=0,1,\cdots,N; i\neq j \tag{6-20}$$

$$a_j\leqslant s_j\leqslant b_j, j=1,2,\cdots,N \tag{6-21}$$

$$t_i=\max\{a_i-s_i,0\} \tag{6-22}$$

在模型中，式（6-13）为目标函数，表示最小运输成本；式（6-14）定义了车辆的容量约束；式（6-15）表示每个客户能被一辆车服务；式（6-16）、式（6-17）表示 0～1 变量间的关系；式（6-18）表示由仓库发出了 k 辆车；式（6-19）表示车辆在仓库出发的时刻为 0；式（6-20）表示车辆从客户 i 直接行驶至客户 j 的时间关系；式（6-21）定义了客户服务的时间窗约束；式（6-22）定义了车辆在客户 i 的等待时间。如果 VRP 问题是硬时间窗问题，则必须满足式（6-21）；如果为软时间窗问题，则该项任务提前到达或延后到达，就给予一定的惩罚。

6.4.2 自适应遗传算法求解车辆路径问题

1. 算法编码

本节采用基于自然数的编码方式，采用 i_{kj} 表示第 i_{kj} 个客户，由于车辆数 m 已事先估计，则 VRP 问题的一条染色体可以表示为 $(0,i_{11},i_{12},\cdots,i_{1s},0,i_{21},i_{22},\cdots,i_{2t},0,i_{m1},\cdots,i_{mw},0)$。例如染色体（0 1 2 0 3 4 0 5 6 7 0）表示由 3 辆车完成 7 个顾客的运输任务的路径安排，则该染色体的长度为 $N+m+1=7+3+1=11$，3 个子路径分别如下。

子路径 1：仓库→客户 1→客户 2→仓库。

子路径 2：仓库→客户 3→客户 4→仓库。

子路径 3：仓库→客户 5→客户 6→客户 7→仓库。

从上述染色体样例中，可以得知，0 表示车场，一共有 $(m+1)=3+1=4$ 个，这 4 个 0 把整个染色体分为 3 段，代表 3 个子路径。

2. 约束处理

VRP 及 VRPTW 约束复杂，用遗传算法求解时，可采用惩罚的方法来处理约束，以保证种群中染色体的多样性，使得遗传算法的搜索能够继续下去。这里采用参考文献[186]中的约束方法。

对一般的 VRP，可用目标函数添加惩罚值的方法解决：

$$f=\min(\sum_i\sum_j\sum_k c_{ij}x_{ijk}+M\sum_{k=1}^{m}\max(\sum_i g_i y_{ik}-q,0)) \tag{6-23}$$

其中，$M\sum_{k=1}^{m}\max(\sum_i g_i y_{ik}-q,0)$ 表示若解违反容量约束则处以的惩罚值。这里，M 可取一个适当大的正整数，本节求解问题时，M=10 000。

对于带时间窗的 VRP，可把时间窗也变为惩罚函数作为目标函数的一部分：

$$f=\min(\sum_i\sum_j\sum_k c_{ij}x_{ijk}+M\sum_{k=1}^{m}\max(\sum_i g_i y_{ik}-q,0)+d\sum_j\max(a_j-s_j,0)+e\sum_j\max(s_j-b_j,0)) \tag{6-24}$$

式中，d 表示车辆因提前到达客户处而处以的等待罚款；e 表示车辆因延迟到达而处以的延迟罚款；在软 VRPTW 中，d，e 取相应的单位惩罚值；硬 VRPTW 中，取 d=e=M，即一个适当大的正整数；表示时间窗约束必须满足。

3. 适应度转换

在遗传算法中，对染色体的适应度函数要求非负，并且一般的进化规则是求适应度最大。因为 VRP 及 VRPTW 的目标函数均为正数，所以可以直接取目标函数的倒数作为适应度函数，即 $\text{fitness}_i=1/f_i$。

4. 交叉算子与变异算子

由于 VRP 及 VRPTW 的条件约束性，如使用简单的交叉算子，将有很大概率产生大量的不可行解或导致优良基因段丢失，使遗传算法的搜索结果不好。例如，

在父代（0 1|2 3 0 4|6 5 0）中（||内为进行交叉的基因段），2、3 段已为优良基因段，但交叉后可能丢失这段优良基因。这里采用改进的最大保留交叉[187]来增大保持优良基因段的概率。具体做法如下：

（1）随机选择两个相邻的零点为交叉位，按照顺序交叉[188]交叉两个父代交叉段外的元素。

（2）对第一步操作后产生的两个子代进行基因位调整。把交叉段间的元素移到染色体的首位，其余非零元素按照原来的先后顺序排列到其后，最后排列剩余的零元素。

（3）对第二步操作后的两个子代进行插零工作。把子代尾部的各个零元素都插入到交叉段后的各个非零元素当中，并且使各个零都不相邻，并保证染色体末位保留一个零。

经过以上三步操作，就产生了两个满足染色体原始构成结构的子代，并增加了保持优良基因段的可能性。如有两个父代：

old1=（|0 1 8 6 0|5 0 2 3 0 9 7 4 0）

old2=（0 9|0 1 5 4 6 0|7 3 0 8 2 0）

若选择的两个交叉点分别为 old1：1 和 5，old2：3 和 8，我们记为[1, 5]和[3, 8]。则经过第一步变换后，产生的两个子代为：

new1=（0 1 8 6 0 0 9 5 4 7 3 0 2 0）

new2=（8 0 0 1 5 4 6 0 2 3 0 9 7 0）

经过第二步位置变换后的两个后代为：

new1=（0 1 8 6 0 9 5 4 7 3 2 0 0 0）

new2=（0 1 5 4 6 0 8 2 3 9 7 0 0 0）

第三步若选择的插入点分别为第一个子代 new1 的 7、8 位，第二个子代 new2 的 9、10 位，也就是把零元素分别插入到各插入位的元素后，产生子代如下：

new1=（0 1 8 6 0|9 5 0 4 0 7 3 2 0）

new2=（0 1 5 4 6 0|8 2 3 0 9 0 7 0）

变异算子本文选择的是简单的 2-交换变异和 3-交换变异。2-交换变异即随机选择一个染色体的两个非零元素，交换这两个元素的位置生成新的染色体。3-交换变异同理，即随机选择一个染色体的三个非零元素，交换这三个元素的位置生成 6 个新的染色体，从这 6 个染色体中选择适应值最高的一个作为变异后的新染色体。

5. 实验结果

采用 2.2.4 节介绍的改进自适应遗传算法求解 VRPTW 问题，采用上述介绍的基于自然数编码方式，取目标函数的倒数作为适应度函数。对一个 VRPTW 实例进行了实验和实验分析。计算实例取自参考文献[189]，未做数据改动。有 9 个客户，编号为 1，…，9，由仓库 0 派出容量均为 1 的车辆执行货运任务。参考文献[189]部分原始数据如表 6-8 所示。

表 6-8　参考文献[189]部分原始数据

i	0	1	2	3	4
(x_i, y_i)	(99, 39)	(39, 70)	(22, 15)	(97, 79)	(6, 28)
g_i	0	0.102	0.113	0.095	0.131
$[a_i, b_i]$	[0, 0]	[74, 124]	[58, 108]	[15, 65]	[96, 146]
i	5	6	7	8	9
(x_i, y_i)	(62, 79)	(42, 88)	(86, 3)	(87, 61)	(88, 91)
g_i	0.029	0.306	0.532	0.617	0.232
$[a_i, b_i]$	[47, 97]	[85, 135]	[21, 71]	[9, 59]	[37, 87]

取 α=0.85，由表 6-8 中的数据和 α 的值可得所需车辆数 m=3。

遗传算法中参数设置如下：迭代次数为 100；种群规模为 20；交叉概率取 0.8，变异概率取 0.5。

取 d=2，e=3，按式（6-24）求解软时间窗问题。算法运行 15 次，得到最小费用为 459.1751。其中，运输成本为 459.1751，各惩罚值均为 0。表示该解的一个染色体为（0 7 2 4 0 3 9 5 6 1 0 8 0），对应的子路径为：

0→7→2→4→0

0→3→9→5→6→1→0

0→8→0