第二十三章　现代优化算法

现代优化算法是 80 年代初兴起的启发式算法。这些算法包括禁忌搜索（tabu search），模拟退火（simulated annealing），遗传算法（genetic algorithms），人工神经网络（neural networks）。它们主要用于解决大量的实际应用问题。目前，这些算法在理论和实际应用方面得到了较大的发展。无论这些算法是怎样产生的，它们有一个共同的目标－求 NP-hard 组合优化问题的全局最优解。虽然有这些目标，但 NP-hard 理论限制它们只能以启发式的算法去求解实际问题。

启发式算法包含的算法很多，例如解决复杂优化问题的蚁群算法（Ant Colony Algorithms）。有些启发式算法是根据实际问题而产生的，如解空间分解、解空间的限制等；另一类算法是集成算法，这些算法是诸多启发式算法的合成。

现代优化算法解决组合优化问题，如 TSP（Traveling Salesman Problem）问题，QAP（Quadratic Assignment Problem）问题，JSP（Job-shop Scheduling Problem）问题等效果很好。

§1　模拟退火算法

1.1　算法简介

模拟退火算法得益于材料的统计力学的研究成果。统计力学表明材料中粒子的不同结构对应于粒子的不同能量水平。在高温条件下，粒子的能量较高，可以自由运动和重新排列。在低温条件下，粒子能量较低。如果从高温开始，非常缓慢地降温（这个过程被称为退火），粒子就可以在每个温度下达到热平衡。当系统完全被冷却时，最终形成处于低能状态的晶体。

如果用粒子的能量定义材料的状态，Metropolis 算法用一个简单的数学模型描述了退火过程。假设材料在状态 i 之下的能量为 $E(i)$，那么材料在温度 T 时从状态 i 进入状态 j 就遵循如下规律：

（1）如果 $E(j) \le E(i)$，接受该状态被转换。

（2）如果 $E(j) > E(i)$，则状态转换以如下概率被接受：

$$e^{\frac{E(i)-E(j)}{KT}}$$

其中 K 是物理学中的波尔兹曼常数，T 是材料温度。

在某一个特定温度下，进行了充分的转换之后，材料将达到热平衡。这时材料处于状态 i 的概率满足波尔兹曼分布：

$$P_T(x=i)=\frac{e^{-\frac{E(i)}{KT}}}{\sum_{j\in S} e^{-\frac{E(j)}{KT}}}$$

其中 x 表示材料当前状态的随机变量，S 表示状态空间集合。

显然

$$\lim_{T\to\infty}\frac{e^{-\frac{E(i)}{KT}}}{\sum_{j\in S} e^{-\frac{E(j)}{KT}}}=\frac{1}{|S|}$$

其中$|S|$表示集合S中状态的数量。这表明所有状态在高温下具有相同的概率。而当温度下降时，

$$\lim_{T\to 0}\frac{e^{-\frac{E(i)-E_{\min}}{KT}}}{\sum_{j\in S}e^{-\frac{E(j)-E_{\min}}{KT}}}=\lim_{T\to 0}\frac{e^{-\frac{E(i)-E_{\min}}{KT}}}{\sum_{j\in S_{\min}}e^{-\frac{E(j)-E_{\min}}{KT}}+\sum_{j\notin S_{\min}}e^{-\frac{E(j)-E_{\min}}{KT}}}$$

$$=\lim_{T\to 0}\frac{e^{-\frac{E(i)-E_{\min}}{KT}}}{\sum_{j\in S_{\min}}e^{-\frac{E(j)-E_{\min}}{KT}}}=\begin{cases}\dfrac{1}{|S_{\min}|} & 若i\in S_{\min}\\ 0 & 其它\end{cases}$$

其中$E_{\min}=\min_{j\in S}E(j)$且$S_{\min}=\{i\mid E(i)=E_{\min}\}$。

上式表明当温度降至很低时，材料会以很大概率进入最小能量状态。

假定我们要解决的问题是一个寻找最小值的优化问题。将物理学中模拟退火的思想应用于优化问题就可以得到模拟退火寻优方法。

考虑这样一个组合优化问题：优化函数为$f:x\to R^+$，其中$x\in S$，它表示优化问题的一个可行解，$R^+=\{y\mid y\in R,y>0\}$，S表示函数的定义域。$N(x)\subseteq S$表示x的一个邻域集合。

首先给定一个初始温度T_0和该优化问题的一个初始解$x(0)$，并由$x(0)$生成下一个解$x'\in N(x(0))$，是否接受x'作为一个新解$x(1)$依赖于下面概率：

$$P(x(0)\to x')=\begin{cases}1 & 若f(x')<f(x(0))\\ e^{-\frac{f(x')-f(x(0))}{T_0}} & 其它\end{cases}$$

换句话说，如果生成的解x'的函数值比前一个解的函数值更小，则接受$x(1)=x'$作为一个新解。否则以概率$e^{-\frac{f(x')-f(x(0))}{T_0}}$接受$x'$作为一个新解。

泛泛地说，对于某一个温度T_i和该优化问题的一个解$x(k)$，可以生成x'。接受x'作为下一个新解$x(k+1)$的概率为：

$$P(x(k)\to x')=\begin{cases}1 & 若f(x')<f(x(k))\\ e^{-\frac{f(x')-f(x(k))}{T_0}} & 其它\end{cases}\qquad(1)$$

在温度T_i下，经过很多次的转移之后，降低温度T_i，得到$T_{i+1}<T_i$。在T_{i+1}下重复上述过程。因此整个优化过程就是不断寻找新解和缓慢降温的交替过程。最终的解是对该问题寻优的结果。

我们注意到，在每个T_i下，所得到的一个新状态$x(k+1)$完全依赖于前一个状态$x(k)$，可以和前面的状态$x(0),\cdots,x(k-1)$无关，因此这是一个马尔可夫过程。使用马尔可夫过程对上述模拟退火的步骤进行分析，结果表明：从任何一个状态$x(k)$生成x'的概率，在$N(x(k))$中是均匀分布的，且新状态x'被接受的概率满足式（1），那么经过有限次的转换，在温度T_i下的平衡态x_i的分布由下式给出：

$$P_i(T_i)=\frac{e^{-\frac{f(x_i)}{T}}}{\sum_{j\in S} e^{-\frac{f(x_i)}{T_i}}} \qquad (2)$$

当温度T降为 0 时，x_i的分布为：

$$P_i^*=\begin{cases}\dfrac{1}{|S_{\min}|} & \text{若} x_i\in S_{\min}\\ 0 & \text{其它}\end{cases}$$

并且

$$\sum_{x_i\in S_{\min}} P_i^*=1$$

这说明如果温度下降十分缓慢，而在每个温度都有足够多次的状态转移，使之在每一个温度下达到热平衡，则全局最优解将以概率 1 被找到。因此可以说模拟退火算法可以找到全局最优解。

在模拟退火算法中应注意以下问题：

（1）理论上，降温过程要足够缓慢，要使得在每一温度下达到热平衡。但在计算机实现中，如果降温速度过缓，所得到的解的性能会较为令人满意，但是算法会太慢，相对于简单的搜索算法不具有明显优势。如果降温速度过快，很可能最终得不到全局最优解。因此使用时要综合考虑解的性能和算法速度，在两者之间采取一种折衷。

（2）要确定在每一温度下状态转换的结束准则。实际操作可以考虑当连续m次的转换过程没有使状态发生变化时结束该温度下的状态转换。最终温度的确定可以提前定为一个较小的值T_e，或连续几个温度下转换过程没有使状态发生变化算法就结束。

（3）选择初始温度和确定某个可行解的邻域的方法也要恰当。

1.2　应用举例

例　已知敌方 100 个目标的经度、纬度如表 1 所示。

表 1　经度和纬度数据表

经度	纬度	经度	纬度	经度	纬度	经度	纬度
53.7121	15.3046	51.1758	0.0322	46.3253	28.2753	30.3313	6.9348
56.5432	21.4188	10.8198	16.2529	22.7891	23.1045	10.1584	12.4819
20.1050	15.4562	1.9451	0.2057	26.4951	22.1221	31.4847	8.9640
26.2418	18.1760	44.0356	13.5401	28.9836	25.9879	38.4722	20.1731
28.2694	29.0011	32.1910	5.8699	36.4863	29.7284	0.9718	28.1477
8.9586	24.6635	16.5618	23.6143	10.5597	15.1178	50.2111	10.2944
8.1519	9.5325	22.1075	18.5569	0.1215	18.8726	48.2077	16.8889
31.9499	17.6309	0.7732	0.4656	47.4134	23.7783	41.8671	3.5667
43.5474	3.9061	53.3524	26.7256	30.8165	13.4595	27.7133	5.0706
23.9222	7.6306	51.9612	22.8511	12.7938	15.7307	4.9568	8.3669
21.5051	24.0909	15.2548	27.2111	6.2070	5.1442	49.2430	16.7044
17.1168	20.0354	34.1688	22.7571	9.4402	3.9200	11.5812	14.5677
52.1181	0.4088	9.5559	11.4219	24.4509	6.5634	26.7213	28.5667
37.5848	16.8474	35.6619	9.9333	24.4654	3.1644	0.7775	6.9576
14.4703	13.6368	19.8660	15.1224	3.1616	4.2428	18.5245	14.3598
58.6849	27.1485	39.5168	16.9371	56.5089	13.7090	52.5211	15.7957
38.4300	8.4648	51.8181	23.0159	8.9983	23.6440	50.1156	23.7816
13.7909	1.9510	34.0574	23.3960	23.0624	8.4319	19.9857	5.7902

40.8801	14.2978	58.8289	14.5229	18.6635	6.7436	52.8423	27.2880
39.9494	29.5114	47.5099	24.0664	10.1121	27.2662	28.7812	27.6659
8.0831	27.6705	9.1556	14.1304	53.7989	0.2199	33.6490	0.3980
1.3496	16.8359	49.9816	6.0828	19.3635	17.6622	36.9545	23.0265
15.7320	19.5697	11.5118	17.3884	44.0398	16.2635	39.7139	28.4203
6.9909	23.1804	38.3392	19.9950	24.6543	19.6057	36.9980	24.3992
4.1591	3.1853	40.1400	20.3030	23.9876	9.4030	41.1084	27.7149

我方有一个基地，经度和纬度为（70,40）。假设我方飞机的速度为 1000 公里/小时。我方派一架飞机从基地出发，侦察完敌方所有目标，再返回原来的基地。在敌方每一目标点的侦察时间不计，求该架飞机所花费的时间（假设我方飞机巡航时间可以充分长）。

这是一个旅行商问题。我们依次给基地编号为 1，敌方目标依次编号为 2，3，…，101，最后我方基地再重复编号为 102（这样便于程序中计算）。距离矩阵 $D=(d_{ij})_{102\times 102}$，其中 d_{ij} 表示表示 i,j 两点的距离，$i,j=1,2,\cdots,102$，这里 D 为实对称矩阵。则问题是求一个从点 1 出发，走遍所有中间点，到达点 102 的一个最短路径。

上面问题中给定的是地理坐标（经度和纬度），我们必须求两点间的实际距离。设 A,B 两点的地理坐标分别为 (x_1,y_1)，(x_2,y_2)，过 A,B 两点的大圆的劣弧长即为两点的实际距离。以地心为坐标原点 O，以赤道平面为 XOY 平面，以 0 度经线圈所在的平面为 XOZ 平面建立三维直角坐标系。则 A,B 两点的直角坐标分别为：

$$A(R\cos x_1\cos y_1, R\sin x_1\cos y_1, R\sin y_1)$$

$$B(R\cos x_2\cos y_2, R\sin x_2\cos y_2, R\sin y_2)$$

其中 $R=6370$ 为地球半径。

A,B 两点的实际距离

$$d=R\arccos\left(\frac{\overrightarrow{OA}\cdot\overrightarrow{OB}}{\left|\overrightarrow{OA}\right|\cdot\left|\overrightarrow{OB}\right|}\right),$$

化简得

$$d=R\arccos[\cos(x_1-x_2)\cos y_1\cos y_2+\sin y_1\sin y_2]。$$

求解的模拟退火算法描述如下：

（1）解空间

解空间 S 可表为 $\{1,2,\cdots,101,102\}$ 的所有固定起点和终点的循环排列集合，即

$$S=\{(\pi_1,\cdots,\pi_{102})\mid \pi_1=1,(\pi_2,\cdots,\pi_{101})\text{为}\{2,3,\cdots,101\}\text{的循环排列},\pi_{102}=102\}$$

其中每一个循环排列表示侦察 100 个目标的一个回路，$\pi_i=j$ 表示在第 i 次侦察 j 点，初始解可选为 $(1,2,\cdots,102)$，本文中我们使用 Monte Carlo 方法求得一个较好的初始解。

（2）目标函数

此时的目标函数为侦察所有目标的路径长度或称代价函数。我们要求

$$\min f(\pi_1,\pi_2,\cdots,\pi_{102})=\sum_{i=1}^{101}d_{\pi_i\pi_{i+1}}$$

而一次迭代由下列三步构成：

（3）新解的产生

① 2 变换法

任选序号 u,v（$u<v$）交换u与v之间的顺序，此时的新路径为：

$$\pi_1\cdots\pi_{u-1}\pi_v\pi_{v+1}\cdots\pi_{u+1}\pi_u\pi_{v+1}\cdots\pi_{102}$$

② 3 变换法

任选序号u,v和w，将u和v之间的路径插到w之后，对应的新路径为（设$u<v<w$）

$$\pi_1\cdots\pi_{u-1}\pi_{v+1}\cdots\pi_w\pi_u\cdots\pi_v\pi_{w+1}\cdots\pi_{102}$$

（4）代价函数差

对于 2 变换法，路径差可表示为

$$\Delta f=(d_{\pi_{u-1}\pi_v}+d_{\pi_u\pi_{v+1}})-(d_{\pi_{u-1}\pi_u}+d_{\pi_v\pi_{v+1}})$$

（5）接受准则

$$P=\begin{cases}1 & \Delta f<0\\ \exp(-\Delta f/T) & \Delta f\geq 0\end{cases}$$

如果$\Delta f<0$，则接受新的路径。否则，以概率$\exp(-\Delta f/T)$接受新的路径，即若$\exp(-\Delta f/T)$大于 0 到 1 之间的随机数则接受。

（6）降温

利用选定的降温系数α进行降温即：$T\leftarrow\alpha T$，得到新的温度，这里我们取$\alpha=0.999$。

（7）结束条件

用选定的终止温度$e=10^{-30}$，判断退火过程是否结束。若$T<e$，算法结束，输出当前状态。

我们编写如下的 matlab 程序如下：

```
clc,clear
load sj.txt      %加载敌方 100 个目标的数据，数据按照表格中的位置保存在纯文本文件 sj.txt 中
x=sj(:,1:2:8);x=x(:);
y=sj(:,2:2:8);y=y(:);
sj=[x y];
d1=[70,40];
sj=[d1;sj;d1];
sj=sj*pi/180;
%距离矩阵 d
d=zeros(102);
for i=1:101
    for j=i+1:102
        temp=cos(sj(i,1)-sj(j,1))*cos(sj(i,2))*cos(sj(j,2))+sin(sj(i,2))*sin(sj(j,2));
        d(i,j)=6370*acos(temp);
    end
end
d=d+d';
S0=[];Sum=inf;
rand('state',sum(clock));
for j=1:1000
    S=[1 1+randperm(100),102];
    temp=0;
```

```
        for i=1:101
            temp=temp+d(S(i),S(i+1));
        end
        if temp<Sum
            S0=S;Sum=temp;
        end
end
e=0.1^30;L=20000;at=0.999;T=1;
%退火过程
for k=1:L
    %产生新解
c=2+floor(100*rand(1,2));
c=sort(c);
c1=c(1);c2=c(2);
   %计算代价函数值
   df=d(S0(c1-1),S0(c2))+d(S0(c1),S0(c2+1))-d(S0(c1-1),S0(c1))-d(S0(c2),S0(c2+1));
    %接受准则
   if df<0
   S0=[S0(1:c1-1),S0(c2:-1:c1),S0(c2+1:102)];
   Sum=Sum+df;
   elseif exp(-df/T)>rand(1)
   S0=[S0(1:c1-1),S0(c2:-1:c1),S0(c2+1:102)];
   Sum=Sum+df;
   end
   T=T*at;
    if T<e
        break;
    end
end
 % 输出巡航路径及路径长度
S0,Sum
```

计算结果为 44 小时左右。其中的一个巡航路径如图 1 所示。

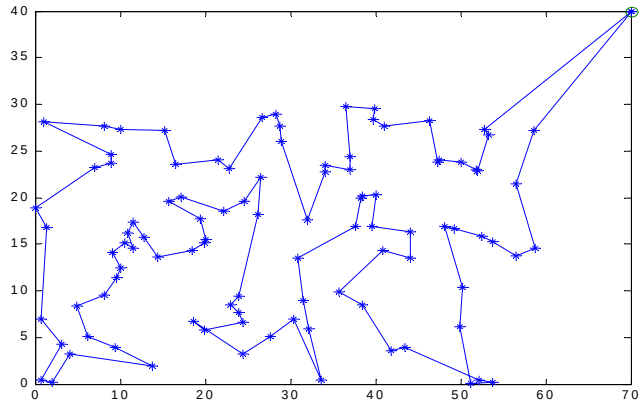

图 1　模拟退火算法求得的巡航路径示意图

§2　遗传算法

2.1　遗传算法简介

遗传算法（Genetic Algorithms，简称 GA）是一种基于自然选择原理和自然遗传机制的搜索（寻优）算法，它是模拟自然界中的生命进化机制，在人工系统中实现特定目标的优化。遗传算法的实质是通过群体搜索技术，根据适者生存的原则逐代进化，最终得到最优解或准最优解。它必须做以下操作：初始群体的产生、求每一个体的适应度、根据适者生存的原则选择优良个体、被选出的优良个体两两配对，通过随机交叉其染色体的基因并随机变异某些染色体的基因后生成下一代群体，按此方法使群体逐代进化，直到满足进化终止条件。其实现方法如下：

（1） 根据具体问题确定可行解域，确定一种编码方法，能用数值串或字符串表示可行解域的每一解。

（2） 对每一解应有一个度量好坏的依据，它用一函数表示，叫做适应度函数，适应度函数应为非负函数。

（3） 确定进化参数群体规模 M 、交叉概率 p_c、变异概率 p_m、进化终止条件。

为便于计算，一般来说，每一代群体的个体数目都取相等。群体规模越大、越容易找到最优解，但由于受到计算机的运算能力的限制，群体规模越大，计算所需要的时间也相应的增加。进化终止条件指的是当进化到什么时候结束，它可以设定到某一代进化结束，也可能根据找出近似最优是否满足精度要求来确定。表 1 列出了生物遗传概念在遗传算法中的对应关系。

表 2 生物遗传概念在遗传算法中的对应关系

生物遗传概念	遗传算法中的作用
适者生存	算法停止时，最优目标值的解有最大的可能被留住
个体	解
染色体	解的编码
基因	解中每一分量的特征
适应性	适应度函数值
种群	根据适应度函数值选取的一组解
交配	通过交配原则产生一组新解的过程
变异	编码的某一分量发生变化的过程

2.2 模型及算法

我们用遗传算法研究 3.1.2 中的问题。

求解的遗传算法的参数设定如下：

种群大小：$M = 50$

最大代数：$G = 1000$

交叉率：$p_c = 1$，交叉概率为 1 能保证种群的充分进化。

变异率：$p_m = 0.1$， 一般而言，变异发生的可能性较小。

（1） 编码策略

采用十进制编码，用随机数列 $\omega_1\omega_2\ldots\omega_{102}$ 作为染色体，其中 $0<\omega_i<1$（$i=2,3,\cdots,101$），$\omega_1=0$，$\omega_{102}=1$；每一个随机序列都和种群中的一个个体相对应，例如一个 9 城市问题的一个染色体为

[0.23，0.82，0.45，0.74，0.87，0.11，0.56，0.69，0.78]

其中编码位置 i 代表城市 i，位置 i 的随机数表示城市 i 在巡回中的顺序，我们将这些随机数按升序排列得到如下巡回：

6－1－3－7－8－4－9－2－5

（2） 初始种群

本文中我们先利用经典的近似算法—改良圈算法求得一个较好的初始种群。即对于初始圈 $C=\pi_1\cdots\pi_{u-1}\pi_u\pi_{u+1}\cdots\pi_{v-1}\pi_v\pi_{v+1}\cdots\pi_{102}$，$2\le u<v\le 101$，$2\le\pi_u<\pi_v\le 101$，交换 u 与 v 之间的顺序，此时的新路径为：

$$\pi_1\cdots\pi_{u-1}\pi_v\pi_{v-1}\cdots\pi_{u+1}\pi_u\pi_{v+1}\cdots\pi_{102}$$

记 $\Delta f=(d_{\pi_{u-1}\pi_v}+d_{\pi_u\pi_{v+1}})-(d_{\pi_{u-1}\pi_u}+d_{\pi_v\pi_{v+1}})$，若 $\Delta f<0$，则以新的路经修改旧的路经，直到不能修改为止。

（3） 目标函数

目标函数为侦察所有目标的路径长度，适应度函数就取为目标函数。我们要求

$$\min f(\pi_1,\pi_2,\cdots,\pi_{102})=\sum_{i=1}^{101}d_{\pi_i\pi_{i+1}}$$

（4） 交叉操作

我们的交叉操作采用单点交叉。设计如下，对于选定的两个父代个体 $f_1=\omega_1\omega_2\ldots\omega_{102}$，$f_2=\omega'_1\omega'_2\ldots\omega'_{102}$，我们随机地选取第 t 个基因处为交叉点，则经过交叉运算后得到的子代编码为 s_1 和 s_2，s_1 的基因由 f_1 的前 t 个基因和 f_2 的后 $102-t$ 个基因构成，s_2 的基因由 f_2 的前 t 个基因和 f_1 的后 $102-t$ 个基因构成，例如：

$f_1=[0,\ \ 0.14,\ \ 0.25,\ \ 0.27,|\ \ 0.29,\ \ 0.54,\cdots,0.19,\ \ 1]$

$f_2=[0,\ \ 0.23,\ \ 0.44,\ \ 0.56,|\ \ 0.74,\ \ 0.21,\cdots,0.24,\ \ 1]$

设交叉点为第四个基因处，则

$s_1=[0,\ \ 0.14,\ \ 0.25,\ \ 0.27,|\ \ 0.74,\ \ 0.21,\cdots,0.24,\ \ 1]$

$s_2=[0,\ \ 0.23,\ \ 0.44,\ \ 0.56,|\ \ 0.29,\ \ 0.54,\cdots,0.19,\ \ 1]$

交叉操作的方式有很多种选择，我们应该尽可能选取好的交叉方式，保证子代能继承父代的优良特性。同时这里的交叉操作也蕴含了变异操作。

（5） 变异操作

变异也是实现群体多样性的一种手段，同时也是全局寻优的保证。具体设计如下，按照给定的变异率，对选定变异的个体，随机地取三个整数，满足 $1<u<v<w<102$，把 u,v 之间（包括 u 和 v）的基因段插到 w 后面。

（6） 选择

采用确定性的选择策略，也就是说选择目标函数值最小的 M 个个体进化到下一代，这样可以保证父代的优良特性被保存下来。

2.3 模型求解及结论

编写 MATLAB 程序如下：

```
tic
```

```
clc,clear
load sj.txt            %加载敌方 100 个目标的数据
x=sj(:,1:2:8);x=x(:);
y=sj(:,2:2:8);y=y(:);
sj=[x y];
d1=[70,40];
sj0=[d1;sj;d1];
%距离矩阵 d
sj=sj0*pi/180;
d=zeros(102);
for i=1:101
      for j=i+1:102
            temp=cos(sj(i,1)-sj(j,1))*cos(sj(i,2))*cos(sj(j,2))+sin(sj(i,2))*sin(sj(j,2));
            d(i,j)=6370*acos(temp);
      end
end
d=d+d';L=102;w=50;dai=100;
%通过改良圈算法选取优良父代 A
for k=1:w
      c=randperm(100);
      c1=[1,c+1,102];
      flag=1;
 while flag>0
         flag=0;
    for m=1:L-3
         for n=m+2:L-1
            if d(c1(m),c1(n))+d(c1(m+1),c1(n+1))<d(c1(m),c1(m+1))+d(c1(n),c1(n+1))
                flag=1;
                c1(m+1:n)=c1(n:-1:m+1);
            end
         end
    end
 end
   J(k,c1)=1:102;
end
J=J/102;
J(:,1)=0;J(:,102)=1;
rand('state',sum(clock));
%遗传算法实现过程
A=J;
for k=1:dai   %产生 0～1 间随机数列进行编码
      B=A;
      c=randperm(w);
%交配产生子代 B
      for i=1:2:w
            F=2+floor(100*rand(1));
            temp=B(c(i),F:102);
            B(c(i),F:102)=B(c(i+1),F:102);
            B(c(i+1),F:102)=temp;
      end
```

```
%变异产生子代 C
by=find(rand(1,w)<0.1);
if length(by)==0
      by=floor(w*rand(1))+1;
end
C=A(by,:);
L3=length(by);
for j=1:L3
    bw=2+floor(100*rand(1,3));
    bw=sort(bw);
    C(j,:)=C(j,[1:bw(1)-1,bw(2)+1:bw(3),bw(1):bw(2),bw(3)+1:102]);
end
    G=[A;B;C];
    TL=size(G,1);
    %在父代和子代中选择优良品种作为新的父代
    [dd,IX]=sort(G,2);temp(1:TL)=0;
    for j=1:TL
        for i=1:101
            temp(j)=temp(j)+d(IX(j,i),IX(j,i+1));
        end
    end
       [DZ,IZ]=sort(temp);
       A=G(IZ(1:w),:);
end
path=IX(IZ(1),:)
long=DZ(1)
toc
xx=sj0(path,1);yy=sj0(path,2);
plot(xx,yy,'-o')
```

计算结果为 40 小时左右。其中的一个巡航路径如图 2 所示。

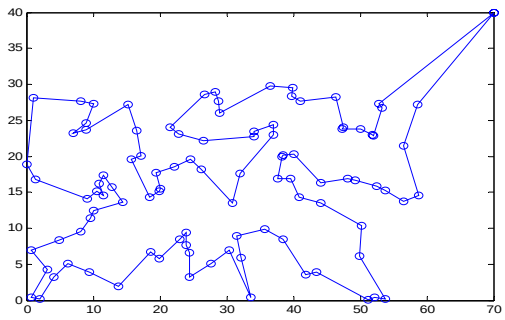

图 2　遗传算法求得的巡航路径示意图

3　禁忌搜索算法

3.1　禁忌搜索算法简介

禁忌搜索算法是组合优化算法的一种，是局部搜索算法的扩展。禁忌搜索算法是人工智能在组合优化算法中的一个成功应用。禁忌搜索算法的特点是采用了禁忌技术。所谓禁忌就是禁止重复前面的工作。禁忌搜索算法用一个禁忌表记录下已经到达过的局部

最优点，在下一次搜索中，利用禁忌表中的信息不再或有选择地搜索这些点。

禁忌搜索算法实现的技术问题是算法的关键。禁忌搜索算法涉及侯选集合、禁忌对象、评价函数、特赦规则、记忆频率信息等概念。

（1）邻域

在组合优化中，距离的概念通常不再适用，但是在一点附近搜索另一个下降的点仍然是组合优化数值求解的基本思想。因此，需要重新定义邻域的概念。

一个组合优化问题可用三参数(D,F,f)表示，其中D表示决策变量的定义域，F表示可行解区域，F中的任何一个元素称为该问题的可行解，f表示目标函数。满足$f(x^*)=\min\{f(x)\mid x\in F\}$的可行解$x^*$称为该问题的最优解。

定义1　对于组合优化问题(D,F,f)，D上的一个映射：

$$N:S\in D\to N(S)\in 2^D$$

称为一个邻域映射，其中2^D表示D的所有子集组成的集合，$N(S)$称为S的邻域，$S'\in N(S)$称为S的一个邻居。

（2）侯选集合

侯选集合由邻域中的邻居组成。常规的方法是从邻域中选择若干个目标值或评价值最佳的邻居入选。

（3）禁忌对象和禁忌长度

禁忌表中的两个主要指标是禁忌对象和禁忌长度。禁忌算法中，由于我们要避免一些操作的重复进行，就要将一些元素放到禁忌表中以禁止对这些元素进行操作，这些元素就是我们指的禁忌对象。禁忌长度是被禁对象不允许选取的迭代次数。一般是给被禁对象x一个数（禁忌长度）t，要求对象x在t步迭代内被禁，在禁忌表中采用$\text{tabu}(x)=t$记忆，每迭代一步，该项指标做运算$\text{tabu}(x)=t-1$，直到$\text{tabu}(x)=0$时解禁。于是，我们可将所有元素分成两类，被禁元素和自由元素。禁忌长度t的选取可以有多种方法，例如$t=$常数，或$t=[\sqrt{n}]$，其中n为邻域中邻居的个数；这种规则容易在算法中实现。

（4）评价函数

评价函数是侯选集合元素选取的一个评价公式，侯选集合的元素通过评价函数值来选取。以目标函数作为评价函数是比较容易理解的。目标值是一个非常直观的指标，但有时为了方便或易于计算，会采用其他函数来取代目标函数。

（5）特赦规则

在禁忌搜索算法的迭代过程中，会出现侯选集中的全部对象都被禁忌，或有一对象被禁，但若解禁则其目标值将有非常大的下降情况。在这样的情况下，为了达到全局最优，我们会让一些禁忌对象重新可选。这种方法称为特赦，相应的规则称为特赦规则。

（6）记忆频率信息

在计算的过程中，记忆一些信息对解决问题是有利的。如一个最好的目标值出现的频率很高，这使我们有理由推测：现有参数的算法可能无法再得到更好的解。根据解决问题的需要，我们可以记忆解集合、被禁对象组、目标值集合等的出现频率。

频率信息有助于进一步加强禁忌搜索的效率。我们可以根据频率信息动态控制禁忌的长度。一个最佳的目标值出现的频率很高，有理由终止计算而将此值认为是最优值。

3.2　模型及求解

我们用禁忌搜索算法研究如下的两个问题：

（1）研究 1.2 中同样的问题。

（2）我方有三个基地，经度、纬度分别为（70,40），（72,45），（68,48）。假设我方所有无人侦察机的速度都为 1000 公里/小时。三个基地各派出一架飞机侦察敌方目标，怎样划分任务，才能使时间最短，且任务比较均衡。

3.2.1　问题（1）的求解

求解的禁忌搜索算法描述如下：

（1）解空间

解空间 S 可表为 $\{1,2,\cdots,101,102\}$ 的所有固定起点和终点的循环排列集合，即

$$S=\{(\pi_1,\cdots,\pi_{102})\,|\,\pi_1=1,(\pi_2,\cdots,\pi_{101})\text{为}\{2,3,\cdots,101\}\text{的循环排列},\pi_{102}=102\}$$

其中每一个循环排列表示侦察 100 个目标的一个回路，$\pi_i=j$ 表示第 i 次侦察 j 点。

（2）目标函数

目标函数为侦察所有目标的路径长度。我们要求

$$\min f(\pi_1,\pi_2,\cdots,\pi_{102})=\sum_{i=1}^{101}w_{\pi_i\pi_{i+1}}$$

（3）候选集合

定义 2　对于路径 $\pi_1\cdots\pi_{u-1}\pi_u\pi_{u+1}\cdots\pi_{v-1}\pi_v\pi_{v+1}\cdots\pi_{102}$，交换 u 与 v 之间的顺序，得到的新路径为

$$\pi_1\cdots\pi_{u-1}\pi_v\pi_{v-1}\cdots\pi_{u+1}\pi_u\pi_{v+1}\cdots\pi_{102},$$

称为原路径二邻域的一个邻居。

定义 3　对于路径 $\pi_1\cdots\pi_u\pi_{u+1}\cdots\pi_{v-1}\pi_v\pi_{v+1}\cdots\pi_w\pi_{w+1}\cdots\pi_{102}$，将 u 和 v 之间的路径插到 w 之后，得到的新路径为

$$\pi_1\cdots\pi_{u-1}\pi_{v+1}\cdots\pi_w\pi_u\cdots\pi_v\pi_{w+1}\cdots\pi_{102},$$

称为原路径三邻域的一个邻居。

如果要考虑当前解的全部二邻域（或三邻域）的邻居，将面临着太大的工作量。因此我们用随机选取的方法每次选取 50 个邻居组成的集合作为侯选集合。而将省下的时间作更多次搜索，这样做同样可以保证较高的精确度，同时可以大大提高算法的效率。

（4）禁忌长度及禁忌对象

对于上述定义的二邻域中的邻居总数为 C_{100}^2，我们的禁忌长度取为 $t=70\approx\sqrt{C_{100}^2}$。

我们把禁忌表设计成一个循环队列，初始化禁忌表 $H=\Phi$。从候选集合 C 中选出一个向量 x，如果 $x\notin H$，并且 H 不满，则把向量 x 添加到禁忌表中；如果 H 已满，则最早进入禁忌表的向量出列，向量 x 进入到出列的位置。

（5）评价函数

可以用目标函数作为评价函数，但是这样每选取一个新的路径都得去计算总时间，计算量比较大。对于上述二邻域中的邻居作为侯选集合，每一个新路径中只有两条边发生了变化，因此将目标函数的差值作为评价函数可以极大地提高算法的效率。评价函数取为

$$\Delta f=(w_{\pi_{u-1}\pi_v}+w_{\pi_u\pi_{v+1}})-(w_{\pi_{u-1}\pi_u}+w_{\pi_v\pi_{v+1}})$$

禁忌搜索算法的流程如下：

STEP1　选定一个初始解 x^{now} 及给以禁忌表 $H=\Phi$；

STEP2　若满足停止条件，停止计算；否则，在 x^{now} 的邻域 $N(H,x^{\text{now}})$ 中选出满足禁忌要求的侯选集 $\text{Can}\_N(x^{\text{now}})$，在 $\text{Can}\_N(x^{\text{now}})$ 中选一个评价值最佳的解 x^{next}，$x^{\text{now}} := x^{\text{next}}$，更新禁忌表 H，重复 STEP2。

利用 Matlab 程序求得，我们的巡航时间大约在 41 小时左右，其中的一个巡航路径如下图所示

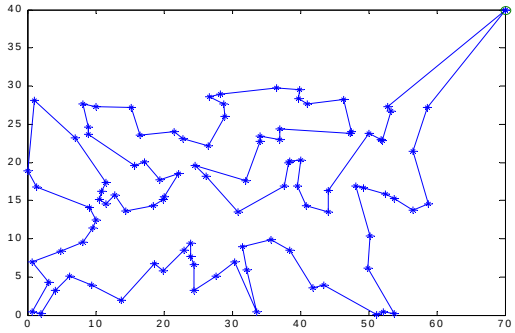

图 3 禁忌搜索算法求得的巡航路径示意图

3.2.2　问题（2）的求解

对于这个问题，我们的基本想法是，先根据敌方基地的分布特点将敌方的基地大体划分在三个区域之内，并使三架侦察机分别对这三个区域的敌军基地进行侦察，求取各自的最短时间。然后对任务不均衡区域之中的点做适当调整。

我们解决问题的步骤如下：

（1）划分子图。要达到比较均衡，应使每架飞机的巡航时间基本相同，由于敌方目标的分布较均匀，可以将敌方目标的地理位置图分成面积基本相同的三部分。如过点（70，40）以斜率 $k_1=\frac{4}{5}$ 作一条斜线，过点（68，48）以斜率 $k_2=\frac{43}{68}$ 作一条斜线，把基地所在的地区划分成三部分 G_1,G_2,G_3（见图 4）。

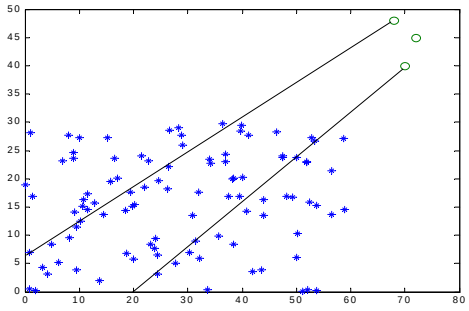

图 4 任务划分示意图

（2）再对每个子图 $G_i(i=1,2,3)$ 分别运用禁忌搜索算法求得其最短侦察路径 ψ_i，和最短侦察时间 $t(\psi_i)\ (i=1,2,3)$。

（3）均衡任务。定义均衡率

$$\eta=\frac{\min\{t(\psi_1),t(\psi_2),t(\psi_3)\}}{\max\{t(\psi_1),t(\psi_2),t(\psi_3)\}}\times 100\%$$

若η接近于 1，则上面划分的任务就可以接受。否则的话，根据$t(\psi_i)$（$i=1,2,3$）的大小用局部搜索算法，调整k_1,k_2的斜率，从而调整各分区内点的个数，直至任务达到均衡。

最后计算结果如下

两直线的斜率分别为$k_1=0.68$，$k_2=0.6484$。

均衡率$\eta=97.57\%$。

三架飞机侦察完所有目标所用的时间分别为

$t(\psi_1)=20.401$小时，$t(\psi_2)=20.910$小时，$t(\psi_3)=20.729$小时。

§4 改进的遗传算法

4.1 引言

无人机航路规划问题实际上是一个组合优化问题，是优化理论中的 NP—hard 问题。因为其解空间不连续，解邻域表达困难，所以难以用通常的算法求解。遗传算法作为现代优化算法之一[1]，其主要特点是对非线性极值问题能以概率 1 跳出局部最优解，找到全局最优解。而遗传算法这种跳出局部最优寻找全局最优特性都基于算法中的交叉和变异。在传统遗传算法的结构中，变异操作在交叉操作基础上进行，强调的是交叉作用，认为变异只是一个生物学背景机制[2]。在具体交叉操作中，人们通常采用断点交叉法（段交叉）多点交叉与均匀交叉，其中断点交叉是指随机地在基因序列中选择一个断点，然后交换双亲上断点右端的所有染色体。在变异操作中，变异算子一般是用 Guassian 分布的随机变异来实现[3,4]。近年来，也有学者尝试用 Cauchy 分布的随机序列来实现变异[5], 希望通过 Cauchy 分布宽大的两翼特性实现更大范围的变异，以利于找到全局最优解。[6]从理论上分析了采用 Cauchy 分布随机变异进化算法的局部收敛性。[7]进一步把二者结合起来, 采用两种分布的线性叠加，但仿真结果显示，算法改进效果并不十分明显。文献[8]将生物进化看成是随机性加上反馈，并指出其中的随机性主要是由系统的内在因素所引起，而不是由外部环境的随机扰动所造成。而混沌系统在其混沌域中表现为随机性，它是确定系统内部随机性的反映，不同于外在的随机特性。本文根据以上特点对基于求解航路规划的遗传算法进行改进，首先将变异操作从交叉操作中分离出来，使其成为独立的并列于交叉的寻优操作，在具体遗传操作中，混沌与遗传操作联系在一起，在交叉操作中，以“门当户对”原则进行个体的配对，利用混沌序列确定交叉点，实行强度最弱的单点交叉，以确保算法收敛精度，削弱和避免寻优抖振问题；在变异操作中，利用混沌序列对染色体中多个基因进行变异，以避免算法早熟。

下面我们研究 1.2 中同样的问题。

4.2 模型及算法

与标准的遗传算法相比，我们做了如下的两点改进。

（1）交叉操作

我们的交叉操作采用改进型交叉。具体设计如下：首先以“门当户对”原则，对父代个体进行配对，即对父代以适应度函数（目标函数）值进行排序，目标函数小的与小的配对，目标函数大的与大的配对。然后利用混沌序列确定交叉点的位置，最后对确定的交叉项进行交叉。例如(Ω_1,Ω_2)配对，他们的染色体分别是$\Omega_1=\omega_1^1\omega_2^1\ldots\omega_{102}^1$，

$\Omega_2 = \omega_1^2\omega_2^2\ldots\omega_{102}^2$，采用 Logistic 混沌序列 $x(n+1)=4x(n)(1-x(n))$ 产生一个 2 到 101 之间的正整数，具体步骤如下：

取一个 $(0,1)$ 随机初始值，然后利用 $x(n+1)=4x(n)(1-x(n))$ 迭代一次产生 1 个 $(0,1)$ 上的混沌值，保存以上混沌值作为产生下一代交叉项的混沌迭代初值，再把这个值分别乘以 100 并加上 2，最后取整即可。假如这个数为 33，那么我们对 (Ω_1,Ω_2) 染色体中相应的基因进行交叉，得到新的染色体 $(\Omega_1^{'},\Omega_2^{'})$

$$\Omega_1^{'} = \omega_1^1\omega_2^1\omega_3^1\omega_4^1\omega_5^1\cdots\omega_{33}^2\omega_{34}^1\cdots\omega_{60}^1\omega_{61}^1\cdots$$

$$\Omega_2^{'} = \omega_1^2\omega_2^2\omega_3^2\omega_4^2\omega_5^2\cdots\omega_{33}^1\omega_{34}^2\cdots\omega_{60}^2\omega_{61}^2\cdots$$

很明显这种单点交叉对原来的解改动很小，这可以削弱避免遗传算法在组合优化应用中产生的寻优抖振问题，可以提高算法收敛精度。

（2）变异操作

变异也是实现群体多样性的一种手段，是跳出局部最优，全局寻优的重要保证。在本文具体变异算子设计如下，首先根据给定的变异率(本文选为 0.02)，随机地取两个在 2 到 101 之间的整数，对这两个数对应位置的基因进行变异，具体变异以当前的基因值为初值利用混沌序列 $x(n+1)=4x(n)(1-x(n))$ 进行适当次数的迭代，得到变异后新的基因值，从而得到新的染色体。

4.3　仿真结果对比及算法性能分析

计算的 MATLAB 程序如下：

```
tic
clc,clear
load sj.txt        %加载敌方 100 个目标的数据
x=sj(:,1:2:8);x=x(:);
y=sj(:,2:2:8);y=y(:);
sj=[x y];
d1=[70,40];
sj=[d1;sj;d1];
%距离矩阵 d
sj=sj*pi/180;
d=zeros(102);
for i=1:101
    for j=i+1:102

        temp=cos(sj(i,1)-sj(j,1))*cos(sj(i,2))*cos(sj(j,2))+sin(sj(i,2)
        )*sin(sj(j,2));
        d(i,j)=6370*acos(temp);
    end
end
d=d+d';L=102;w=50;dai=100;
%通过改良圈算法选取优良父代 A
for k=1:w
    c=randperm(100);
    c1=[1,c+1,102];
```

```
        flag=1;
     while flag>0
         flag=0;
       for m=1:L-3
          for n=m+2:L-1
            if
d(c1(m),c1(n))+d(c1(m+1),c1(n+1))<d(c1(m),c1(m+1))+d(c1(n),c1(n+1))
               flag=1;
               c1(m+1:n)=c1(n:-1:m+1);
            end
          end
       end
     end
      J(k,c1)=1:102;
   end
   J=J/102;
   J(:,1)=0;J(:,102)=1;
   rand('state',sum(clock));
   %遗传算法实现过程
   A=J;
   for k=1:dai  %产生 0～1 间随机数列进行编码
       B=A;
       %交配产生子代 B
       for i=1:2:w
           ch0=rand;ch(1)=4*ch0*(1-ch0);
           for j=2:50
               ch(j)=4*ch(j-1)*(1-ch(j-1));
           end
           ch=2+floor(100*ch);
           temp=B(i,ch);
           B(i,ch)=B(i+1,ch);
           B(i+1,ch)=temp;
       end
   %变异产生子代 C
   by=find(rand(1,w)<0.1);
   if length(by)==0
       by=floor(w*rand(1))+1;
   end
   C=A(by,:);
   L3=length(by);
   for j=1:L3
      bw=2+floor(100*rand(1,3));
      bw=sort(bw);
      C(j,:)=C(j,[1:bw(1)-1,bw(2)+1:bw(3),bw(1):bw(2),bw(3)+1:102]);
```

```
end
   G=[A;B;C];
   TL=size(G,1);
   %在父代和子代中选择优良品种作为新的父代
   [dd,IX]=sort(G,2);temp(1:TL)=0;
   for j=1:TL
       for i=1:101
           temp(j)=temp(j)+d(IX(j,i),IX(j,i+1));
       end
   end
     [DZ,IZ]=sort(temp);
     A=G(IZ(1:w),:);
end
path=IX(IZ(1),:)
long=DZ(1)
toc
```

在仿真试验中，我们对文中航路规划问题分别利用断点交叉和换位变异结合的遗传算法，多点交叉和移位变异结合的遗传算法[10]和文中提出的改进算法进行求解比较。表 1 是各种算法种群规模（$M=50$）和迭代次数（$G=100$）都相同时连续 20 次求解的平均值（公里），算法平均运算时间（秒）。

表 3　算法性能比较表

指标	断点交叉算法	多点交叉算法	文中改进算法
平均航路距离	41572	40416	39648
算法执行时间	5.937	6.125	2.985

本文从算法结构到具体的遗传操作都进行了改进，其中变异操作从交叉操作中分离出来，使得遗传算法也可以通过并行计算实现，提高算法实现效率。其次改进后的算法，分别采用变化强度不同的交叉操作和变异操作，其中交叉操作采用强度最弱的单点交叉，保证了算法收敛精度，削弱和避免算法因交叉强度大而产生的寻优抖振问题。当然单一的单点交叉很容易使算法早熟，文中采用较大强度的多个基因变异正好解决早熟问题。从仿真结果可以看到改进后的算法效果较为明显。

§5　蚁群算法

5.1　蚁群算法简介

蚁群是自然界中常见的一种生物，人们对蚂蚁的关注大都是因为“蚁群搬家，天要下雨”之类的民谚。然而随着近代仿生学的发展，这种似乎微不足道的小东西越来越多地受到学者们地关注。1991 年意大利学者 M. Dorigo 等人首先提出了蚁群算法，人们开始了对蚁群的研究：相对弱小，功能并不强大的个体是如何完成复杂的工作的（如寻找到食物的最佳路径并返回等）。在此基础上一种很好的优化算法逐步发展起来。

蚁群算法的特点是模拟自然界中蚂蚁的群体行为。科学家发现，蚁群总是能够发现从蚁巢到食物源的最短路径。经研究发现，蚂蚁在行走过的路上留下一种挥发性的激素，蚂蚁就是通过这种激素进行信息交流。蚂蚁趋向于走激素积累较多的路径。找到最短路径的蚂蚁总是最早返回巢穴，从而在路上留下了较多的激素。由于最短路径上积累了较多的激素，选择这条路径的蚂蚁就会越来越多，到最后所有的蚂蚁都会趋向于选择

这条最短路径。基于蚂蚁这种行为而提出的蚁群算法具有群体合作，正反馈选择，并行计算等三大特点，并且可以根据需要为人工蚁加入前瞻、回溯等自然蚁所没有的特点。

在使用蚁群算法求解现实问题时，先生成具有一定数量蚂蚁的蚁群，让每一只蚂蚁建立一个解或解的一部分，每只人工蚁从问题的初始状态出发，根据“激素”浓度来选择下一个要转移到的状态，直到建立起一个解，每只蚂蚁根据所找到的解的好坏程度在所经过的状态上释放与解的质量成正比例的“激素”。之后，每只蚂蚁又开始新的求解过程，直到寻找到满意解。为避免停滞现象，引入了激素更新机制。

5.2　解决 TSP 问题的蚁群算法描述

现以 TSP 问题的求解为例说明蚁群系统模型。首先引进如下记号：n 为城市的个数；m 为蚁群中蚂蚁的数量；d_{ij} 为两城市 i 和 j 之间距离；$b_i(t)$ 为 t 时刻位于城市 i 的蚂蚁的个数，$m=\sum_{i=1}^{n} b_i(t)$；$\tau_{ij}(t)$ 为 t 时刻边弧 (i,j) 的轨迹强度（即 ij 连线上残留的信息量），且设 $\tau_{ij}(0)=c$（c 为常数），$i,j=1,2,\cdots,n$，$i\neq j$；$\eta_{ij}(t)$ 为 t 时刻边弧 (i,j) 的能见度，反映由城市 i 转移到城市 j 的期望程度。

根据上述原理，蚂蚁 $k(k=1,2,\cdots,m)$ 在运动过程中根据各条路径上的信息量决定转移方向。与真实蚁群系统不同，人工蚁群系统具有一定的记忆功能。随着时间的推移，以前留下的信息逐渐消逝，经 n 个时刻，蚂蚁完成一次循环，各路径上信息量要作调整。由此得到下述的人工蚁群系统模型：

1）设人工蚁群在并行地搜索 TSP 的解，并通过一种信息素做媒介相互通信，在每个结点上且和该结点相连的边上以信息素量做搜索下一结点的试探依据，直到找到一个 TSP 问题的可行解。

2）在时刻 t 人工蚁 k 由位置 i 转移至位置 j 的转移概率为

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{v\in S}\tau_{iv}^{\alpha}(t)\eta_{iv}^{\beta}(t)}, & j\in S\\ 0, & j\notin S\end{cases} \tag{3}$$

其中参数 α 为轨迹的相对重要性（$\alpha\geq 0$）；β 为能见度的相对重要性（$\beta\geq 0$）；S 为可行点集，即蚂蚁 k 下一步允许选择的城市。α,β 分别反映了蚂蚁在运动过程中所积累的信息及启发式因子在蚂蚁选择路径中所起的不同作用。

3）当 m 个人工蚁按（3）式找到了可行解，则将各边的信息量用下式修改。即调整信息量的轨迹强度更新方程为

$$\tau_{ij}(t+1)=\rho\tau_{ij}(t)+\Delta\tau_{ij},\quad \rho\in(0,1) \tag{4}$$

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$$

其中 $\Delta\tau_{ij}^{k}$ 为第 k 只蚂蚁在本次循环中留在路径 (i,j) 上的信息量；$\Delta\tau_{ij}$ 为本次循环中路径 (i,j) 上的信息量的增量；参数 ρ 为轨迹的持久性；$1-\rho$ 为轨迹衰减度，表示信息消逝程度。

对上述系统模型，采用人工蚁群方法求解的算法步骤可归结为：

step 1：$NC\leftarrow 0$（NC 为迭代步数或搜索次数）；各 τ_{ij} 和 $\Delta\tau_{ij}$ 的初始化；将 m 个蚂蚁置于 n 个顶点上。

step 2：将各蚂蚁的初始出发点置于当前解集中；对每个蚂蚁 k（$k=1,2,\cdots,m$）按概率 p_{ij}^k 转移至下一顶点 j；将顶点 j 置于当前解集。

step 3：计算各蚂蚁的目标函数值 $z_k(k=1,\cdots,m)$，记录当前的最好解。

step 4：按更新方程修改轨迹强度。

step 5：$NC \leftarrow NC+1$，若 $NC <$ 预定的迭代次数且无退化行为（即找到的都是相同解），则转 step 2。

若为了简化计算，增加处理较大规模的 TSP 问题的能力，则可将（4）式修改为：

$$\tau_{ij}(t+1)=\rho\tau_{ij}(t)+(1-\rho)\Delta\tau_{ij},\quad \rho\in(0,1)$$

其中

$$\Delta\tau_{ij}^k=\begin{cases}\dfrac{1}{d_{ij}}, & (i,j)\in BE\\ 0, & \text{其它}\end{cases}$$

此处 BE 为本次最优路线上的边集。

5.3　人工蚁群算法性能的讨论

人工蚁群算法是一种基于种群的进化算法。作为一个新兴的研究领域，虽它还远未像 GA、SA 等算法那样形成系统的分析方法和坚实的数学基础，但目前已有一些基本结果。

在 M. Dorigo 三种不同的模型中，循环路径 (i,j) 上信息量的增量 $\Delta\tau_{ij}$ 不同。

1）Ant-quantity system 模型中，

$$\Delta\tau_{ij}^k=\begin{cases}\dfrac{Q}{d_{ij}}, & \text{若第}k\text{只蚂蚁在时刻}t\text{和}t+1\text{之间经过}ij\\ 0, & \text{其它}\end{cases}$$

2）在 Ant-density system 模型中，

$$\Delta\tau_{ij}^k=\begin{cases}Q, & \text{若第}k\text{只蚂蚁在时刻}t\text{和}t+1\text{之间经过}ij\\ 0, & \text{其它}\end{cases}$$

3）在 Ant-cycle system 模型中，

$$\Delta\tau_{ij}^k=\begin{cases}\dfrac{Q}{L_k}, & \text{若第}k\text{只蚂蚁在本次循环中经过}ij\\ 0, & \text{其它}\end{cases}$$

其中 Q 是反映蚂蚁所留轨迹数量的常数，L_k 表示第 k 只蚂蚁在本次循环中所走路径的长度；且 $t=0$ 时，$\tau_{ij}^k(0)=c$，$\Delta\tau_{ij}^k=0$。算法中模型 1)、2）利用的是局部信息，模型 3）利用的是整体信息。

人工蚁群算法中，α,β,Q 等参数对算法性能也有很大的影响。α 值的大小表明留在每个结点上的信息量受重视的程度，α 值越大，蚂蚁选择以前选过的点的可能性越大，但过大会使搜索过早陷于局部极小点；β 的大小表明启发式信息受重视的程度；Q 值会影响算法的收敛速度，Q 过大会使算法收敛于局部极小值，过小又会影响算法的收敛速度，随问题规模的增大 Q 的值也需要随之变化；蚂蚁的数目越多，算法的全局

搜索能力越强，但数目加大将使算法的收敛速度减慢。

参考文献

[1] 邢文训，谢金星. 现代优化算法. 北京：清华大学出版社, 1999.

[2] 张文修，梁怡. 遗传算法的数学基础. 西安：西安交通大学出版社，2002.

[3] Back T，Hoffimeister F and Schwefel H P，“A survey of evolution strategies,”In Proc of the 4th Int. Genetic Algorithms Conference, CA: Morgan Kaufmann Publishers， pp.2-9, 1991.

[4] Fogel D B, “An introduction to simulated evolutionary optimization,” IEEE Transaction Neural Network, vol.5, no.1, pp.3-14, 1994.

[5] Wei C.J，Yao S.S and He Z.Y.，“A modified evolutionary programming,” In Proc 1996 IEEE Int. Evolutionary Computation Conference, NJ, IEEE Press, pp.135-138, 1996.

[6] Rudolph G, “Local convergence rates of simple evolutionary algorithms with Cauchy mutations,” IEEE Transaction Evolutionary Computation, vol.1, no.4, pp.249-258, 1997.

[7] Chellapilla K.，“Combining mutation operators in evolutionary programming,” IEEE Trans on Evolutionary Computation, vol.2, no.3, pp.91-96, 1998.

[8] 吴祥兴, 陈忠. 混沌学导论. 上海: 上海科学技术文献出版社, 1996.

[9] 吴翊，吴梦达，成礼智编著. 数学建模的理论与实践. 长沙：国防科技大学出版社, 1999.

[10] 玄光男，程润伟著，汪定伟等译. 遗传算法与工程设计. 科学出版社，2000.