模拟退火算法
Simulated Annealing

随机梯度下降法

给定数据集$\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{m}$,其中$\boldsymbol{x}_i = (\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \cdots, \boldsymbol{x}_{in})$

最小二乘拟合模型:

$$\boldsymbol{h}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m} (\boldsymbol{w}_1 \boldsymbol{x}_{i1} + \boldsymbol{w}_2 \boldsymbol{x}_{i2} + \cdots + \boldsymbol{w}_n \boldsymbol{x}_{in} - \boldsymbol{y}_i)^2$$

梯度下降法求解：

$$w_{\mathrm{j}}^{k+1} = w_j^k - \alpha \frac{\partial h(w^k, x, y)}{\partial w_j}$$

$$= w_j^k - \alpha \sum_{i=1}^{m} (w_1^k x_{i1} + w_2^k x_{i2} + \cdots + w_n^k x_{in} - y_i) x_{ij}$$

$$j = 1,2,\cdots,n$$

计算速度较慢

随机梯度下降法求解:随机选一个样本的梯度

$$w_{\mathrm{j}}^{k+1} = w_j^k - \alpha(w_1^k x_{i\,1} + w_2^k x_{i2} + \cdots + w_n^k x_{in} - y_i)x_{ij}$$

$$j = 1,2,\cdots,n$$

速度快，不易收敛到全局最优解

小批量梯度下降法

随机选K个样本的梯度

$$w_{\mathrm{j}}^{k+1} = w_j^k - \sum_{i=1}^{K} \alpha(w_1^k x_{l_i 1} + w_2^k x_{l_i 2} + \cdots + w_n^k x_{l_i n} - y_{l_i}) x_{l_i j}$$

$$j = 1,2,\cdots,n$$

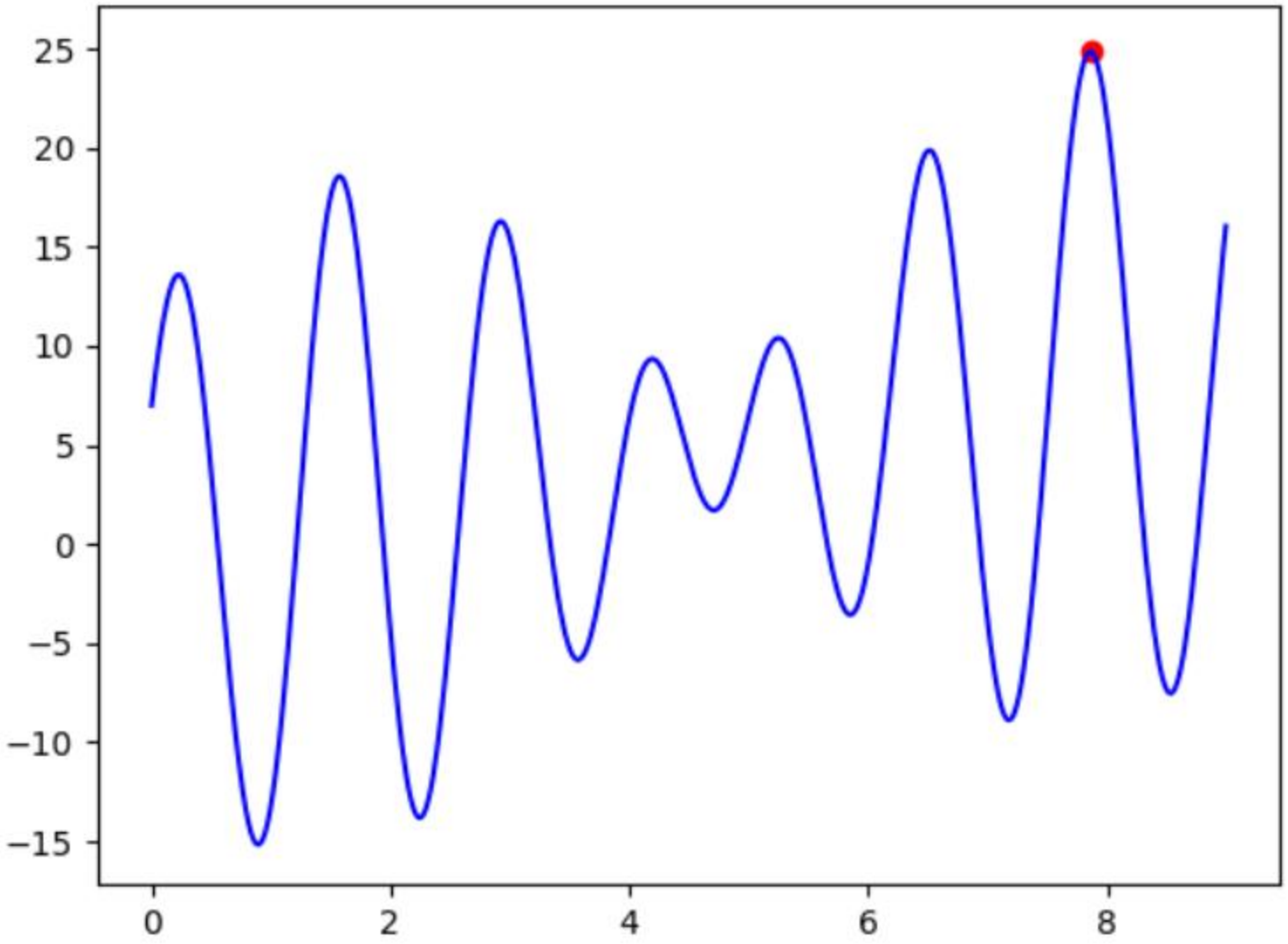
折中算法

作业

给定以下样本集，采用y=ax+b线性拟合，
使用随机梯度下降法
更新权重(至少迭代一步)

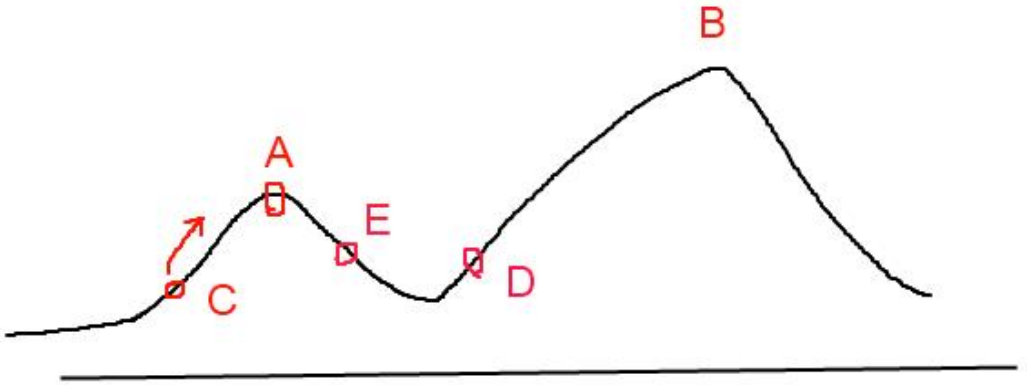
初始：$\boldsymbol{w}^0 = (1,1,1,1,1), \eta = 1$

$$\boldsymbol{x}_1 = (1,3,5,6,4), \boldsymbol{y}_1 = 12$$
$$\boldsymbol{x}_2 = (2,4,6,7,3), \boldsymbol{y}_2 = 15$$
$$\boldsymbol{x}_3 = (3,5,6,8,2), \boldsymbol{y}_3 = 16$$
$$\boldsymbol{x}_4 = (4,3,2,5,1), \boldsymbol{y}_4 = 18$$


25
20
15
10
5
0
−5
−10
−15
0
2
4
6
8

爬山算法

如图所示：假设C点为当前解，爬山算法搜索到A点这个局部最优解就会停止搜索，因为在A点无论向那个方向小幅度移动都不能得到更优的解。



模拟退火算法

在搜索到局部最优解A后，会以一定的概率接受到E的移动。也许经过几次这样的不是局部最优的移动后会到达D点，于是就跳出了局部最大值A。

模拟退火算法

- **算法的提出**

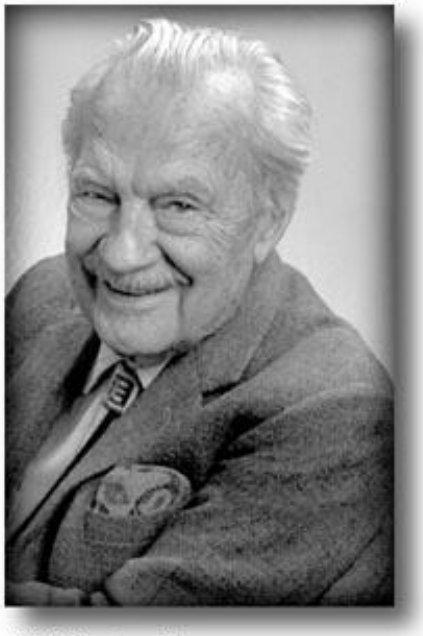
  模拟退火算法最早的思想由Metropolis等（1953）提出，1983年Kirkpatrick等将其应用于组合优化。

- **算法的目的**

  解决具有NP(Non-deterministic Polynomial)复杂性问题；

  克服优化过程陷入局部极小；

  克服初值依赖性。



Nick Metropolis

算法的基础

1、源于对固体退火过程的模拟；

2、采用Metropolis接受准则；

退火工艺：

退火是将金属和合金加热到适当温度，保持一定时间，然后缓慢冷却的热处理工艺。

退火的目的：

①降低钢的硬度，提高塑性，以利于切削加工及冷变形加工。
②细化晶粒，消除因铸、锻、焊引起的组织缺陷，均匀钢的组织和成分，改善钢的性能或为以后的热处理作组织准备。
③消除钢中的内应力，以防止变形和开裂

为什么退火处理？

热力学中的退火现象指物体逐渐降温时发生的物理现象：

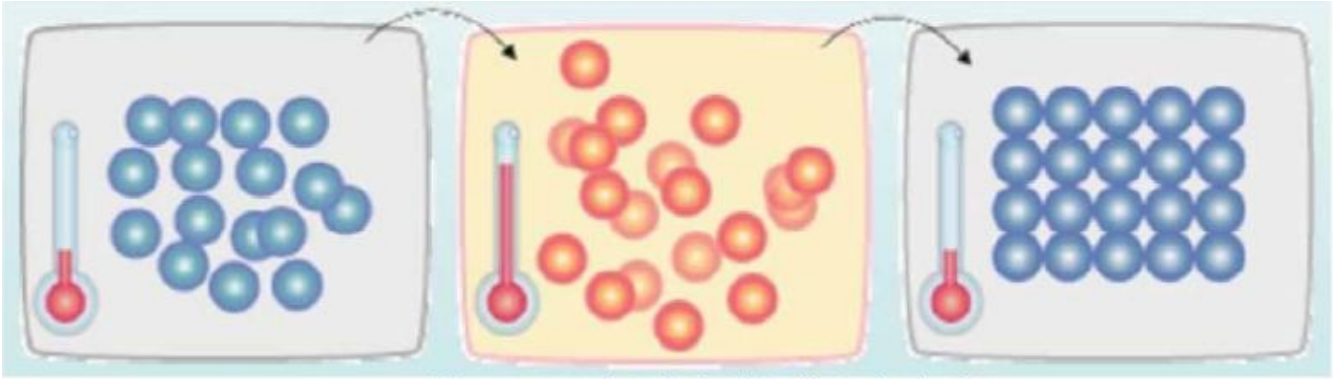
1. 加温时，
固体内部粒子随温度升高变为无序状，内能增大，
用原子或晶格的移动来释放内部残留应力，
通过这些原子排列重组的过程来消除材料中的差。

2. 缓缓降温，
使分子在每一温度时，能够有足够时间找 到安顿位置，
温度越低，物体的能量状态越低，在结晶状态时，系统的能量状态最低，系统最稳定。

3.如果快速 降温，会导致不是最低能态的非晶形。

慢工出细活

物理退火过程



物理退火过程的发展阶段

1. 加温过程。其目的是增强粒子的热运动，使其偏离平衡位置。当温度足够高时，固体将溶解为液体，溶解过程与系统的熵增过程联系，系统能量也随温度的升高而增大，使得每一粒子的状态都具有充分的随机性。

2. 等温过程。物理学的知识告诉我们，对于与周围环境交换热量而温度不变的封闭系统，系统状态的自发变化总是朝自由能减少的方向进行，当自由能达到最小时，系统达到平衡态。

3. 冷却过程。目的是使粒子的热运动减弱并渐趋有序，系统能量逐渐下降，从而得到低能的晶体结构。在常温时达到基态，内能减为最小。

Metropolis准则（1953）——以概率接受新状态

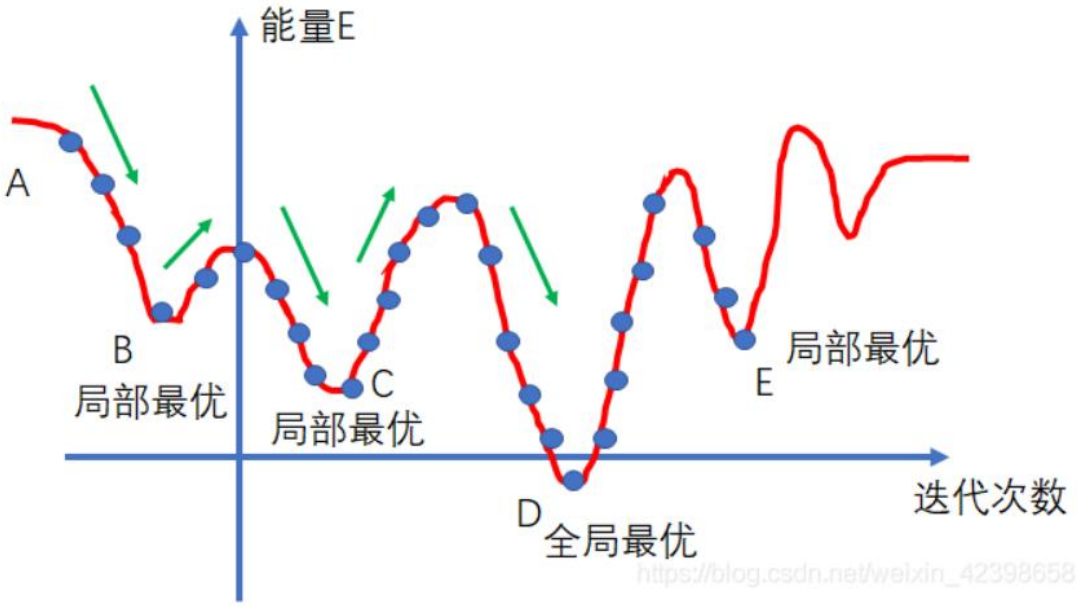
在温度t，初始状态i，该状态的能量为 E_i，

随机选取某个粒子的位移随机地产生一微小变化，得到一个新状态j，新状态的能量为 E_j

$$\begin{cases} E_j < E_i & \text{则接受新状态j} \\ E_j > E_i & \text{则考虑到热运动的影响} \end{cases}$$

$$r = \exp(-\frac{E_j - E_i}{kt}) \qquad \xi \in [0,1) \text{ 随机数}$$

若 $r > \xi$，则接受j，否则接受i。



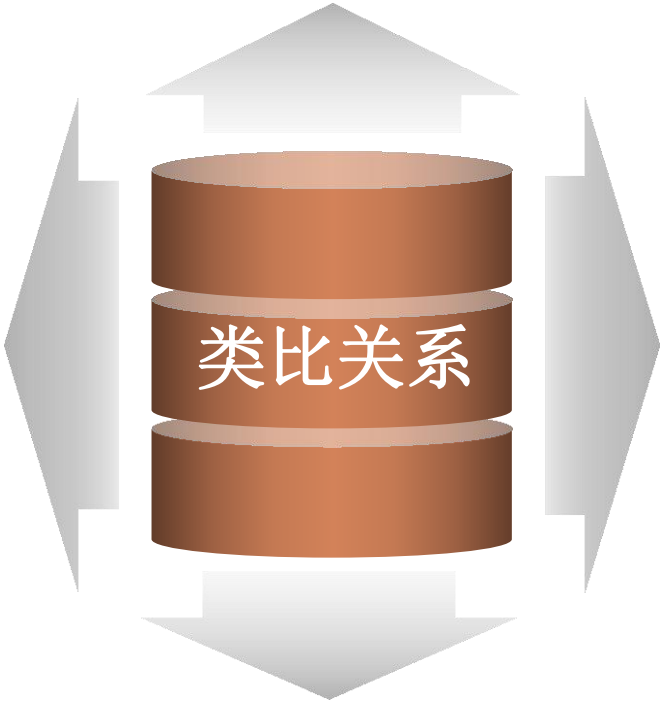
1. 假设开始状态在A，随着迭代次数更新到B的局部最优解，B<A 说明接近最优解了，
2. 状态到达B后，发现下一步能量上升了，(如果是梯度下降则是不允许继续向前的)
3. 而模拟退火准则以一定的概率跳出这个坑

   这个概率的设计十分重要

物理退火过程

物理退火过程

- 物体内部的状态
- 状态的能量
- 温度
- 熔解过程
- 退火冷却过程
- 状态的转移
- 能量最低状态



模拟退火算法

- 问题的解空间
- 解的质量
- 控制参数
- 设定初始温度
- 控制参数的修改
- 解在邻域中的变化
- 最优解

模拟退火算法

算法基本步骤：

1. 确定退火过程的温度初值 T_0 ,随机生成该温度下的粒子状态及一个解 x_0 ，计算目标函数值f即该状态下的内能 E_{x_0} 。

2. 进入下一个冷却温度。

3. 对当前的解 x_i 添加扰动，及产生个新的解 x_j ，计算相应的内能 E_{x_i} 及 E_{x_j} ，根据Metropolis准则判断是否接受新的解，更新局部最优解和全局最优解。

4. 在不同的冷却温度 T_i 下，重复m次对x的扰动，即产生m个新的解，并执行2~3.

5. 判断T是否已经到达了最低温度限制 T_{limit} ，一般设定为 $0.01 \sim 5$ ，是则终止算法；否继续执行2~4.

算法实质分为两层循环:

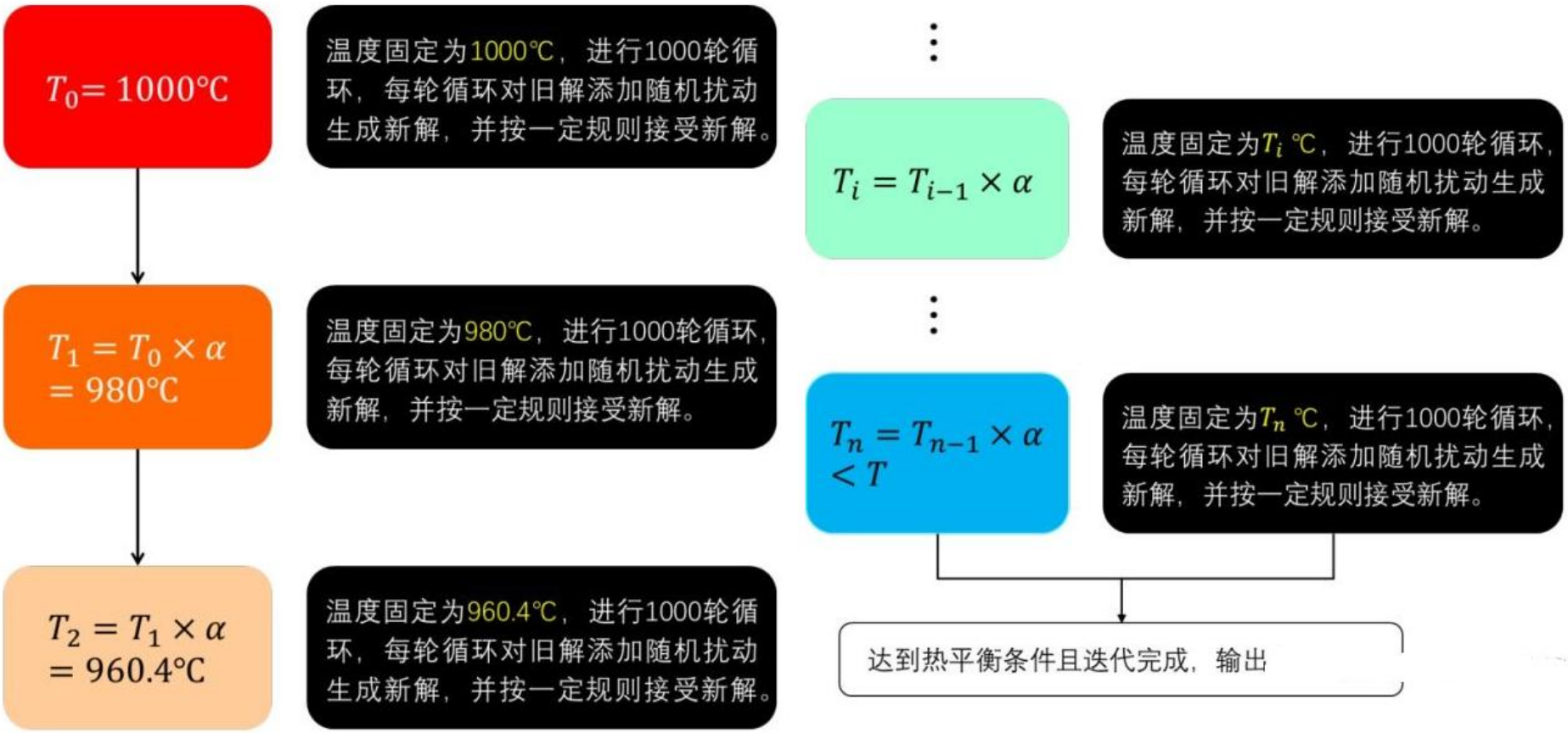
1.内层循环：

在固定温度下随机扰动产生新解，计算目标函数值的变化，决定是否接受。

2.外层循环：

通过缓慢地降低温度，算法可能收敛到全局最优解。

假设初始温度为1000℃，温度衰减系数α = 0.98，热平衡条件为温度小于T℃。



外层循环（彩色模块）控制温度由高向低变化，温度计算公式， 为取值在[0, 1]上的温度衰减系数，如0.95；

内层循环（黑色模块）中，温度固定，对旧解添加随机扰动得到新解，并按一定规则接受新解。

内层循环的迭代次数称为马尔科夫链长度，如上图中的马尔科夫链的长度为1000.

例2 模拟退火求解背包问题

已知背包的装载量为c=8，现有n=5个物品，它们的重量和价值分别是(2, 3, 5, 1, 4)和(2, 5, 8, 3, 6)。试使用模拟退火算法求解该背包问题，写出关键的步骤。

求解：假设问题的一个可行解用0和1的序列表示，例如i=(1010)表示选择第1和第3个物品，而不选择第2和第4个物品。用模拟退火算法求解此例的关键过程如图所示：

运行步骤

已知：
物体个数：n=5
背包容量：c=8
重量 w = (2, 3, 5, 1, 4)
价值 v = (2, 5, 8, 3, 6)

第一步：初始化。假设初始解为i=(11001)，初始温度为T=10。计算$f(i)$=2+5+6=13，最优解s=i

第三步：降温，假设温度降为T=9。如果没有达到结束标准，则返回第二步继续执行

假设在继续运行的时候，从当前解i=(10110)得到一个新解j=(00111)，这时候的函数值为$f(j)$=8+3+6=17，这是一个全局最优解。可见上面过程中接受了劣解是有好处的。

第 二步：在T温度下局部搜索，直到“平衡”，　假设平衡条件为执行了3次内层循环。
（2-1）产生当前解i的一个邻域解j（如何构造邻域根据具体的问题而定，这里假设为随机改变某一位的0/1值或者交换某两位的0/1值），假设j=(11100)
要注意产生的新解的合法性，要舍弃那些总重量超过背包装载量的非法解

（2-2）$f(j)$ = 2+5+8=15 > 13=$f(i)$，所以接受新解j
i=j; $f(i)$=$f(j)$=15; 而且s=i;
要注意求解的是最大值，因此适应值越大越优

（2-3）返回（2-1）继续执行。
(a) 假设第二轮得到的新解j=(11010)，由于$f(j)$ = 2+5+3=10 < 15=$f(i)$，所以需要计算接受概率$P(T)$=exp(($f(j)$-$f(i)$)/T) = exp(-0.5) = 0.607，假设random(0,1)>$P(T)$，则不接受新解

(b) 假设第三轮得到的新解j=(10110)，由于$f(j)$ = 2+8+3=13 < 15=$f(i)$，所以需要计算接受概率$P(T)$=exp(($f(j)$-$f(i)$)/T) = exp(-0.3) = 0.741，假设random(0,1)<$P(T)$，则接受新解
按照一定的概率接受劣解，也是跳出局部最优的一种手段

（2-4）这时候，T温度下的“平衡”已达到（即已经完成了3次的邻域产生），结束内层循环

几点说明

1、新解的产生

要求尽可能地遍及解空间的各个区域，这样，在某一恒定温度下，不断产生新解时，就可能跳出局部最优解。

2、收敛的一般条件：

- 初始温度足够高；
- 热平衡时间足够长；
- 终止温度足够低；
- 降温过程足够缓慢；

3、参数的选择：

(1) 初始值T_0

T_0越大越好，但为了减少计算量，要根据实际情况选择；

(2) 控制参数T的衰减函数

常用$T_{k+1} = aT_k$, a 的取值范围：0.5~0.99；

(3) Markov链长度

$L_k = 100n$， n为问题规模。

- **程序基本步骤**

给定初温$t=t_0$，随机产生初始状态$s=s_0$，令$k=0$；

Repeat

　　Repeat

　　产生新状态s_j=Genete(s)；

　　if min{1,exp[-($C(s_j)$-$C(s)$)/t_k]}>=randrom[0,1]　$s=s_j$;

　　Until 抽样稳定准则满足；

　　退温t_{k+1}=update(t_k)并令$k=k+1$；

Until 算法终止准则满足；

输出算法搜索结果。

- **影响优化结果的主要因素**

给定初温$t=t_0$，随机产生初始状态$s=s_0$，令$k=0$；

Repeat

　　Repeat

　　产生新状态s_j=Genete(s)；

　　if min{1,exp[-($C(s_j)$-$C(s)$)/t_k]}>=randrom[0,1]　$s=s_j$;

　　Until 抽样稳定准则满足；

　　退温t_{k+1}=update(t_k)并令$k=k+1$；

Until 算法终止准则满足；

输出算法搜索结果。

三函数两准则

初始温度

三函数两准则

- 状态产生函数
- 状态接受函数
- 退温函数
- 抽样稳定准则
- 退火结束准则

1 状态产生函数（邻域函数）

设计的出发点：

尽可能保证产生的候选解遍布全部解空间。

两部分
- 产生候选解的方式
- 候选解产生的概率分布

2 状态接受函数 $\min[1, \exp(-\Delta C / t)]$

目的：尽可能接受优化解

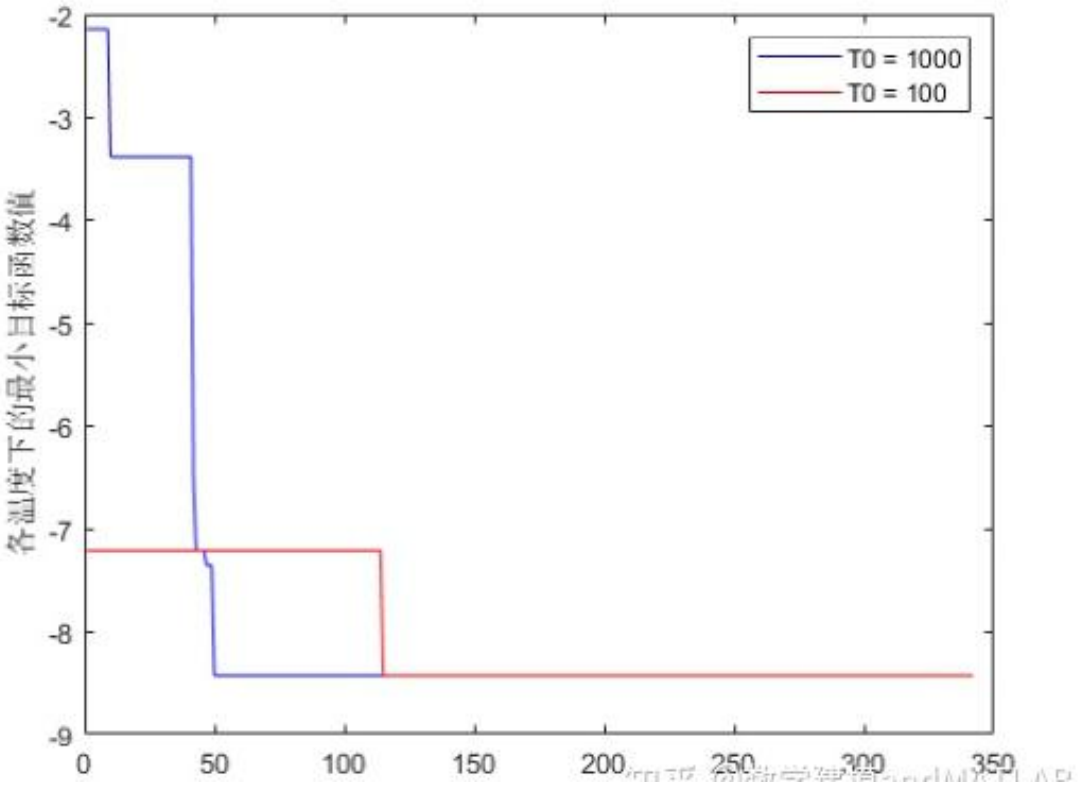
状态接受函数一般以概率的方式给出，不同接受函数的差别主要在于接受概率的形式不同。

3 初温 t_0

实验表明：初温值只要选择充分大，获得高质量解的概率就大！但花费计算时间增加。

✓初温的选择要足够高。

✓初温的确定应折衷考虑优化质量和优化效率。



初始温度100和初始温度1000的都能搜索到全局最小值，
初始温度1000的搜索速度稍快一点。

初始温度

温度更新函数

内循环终止准则

外循环终止准则

退火历程

(annealing schedule)

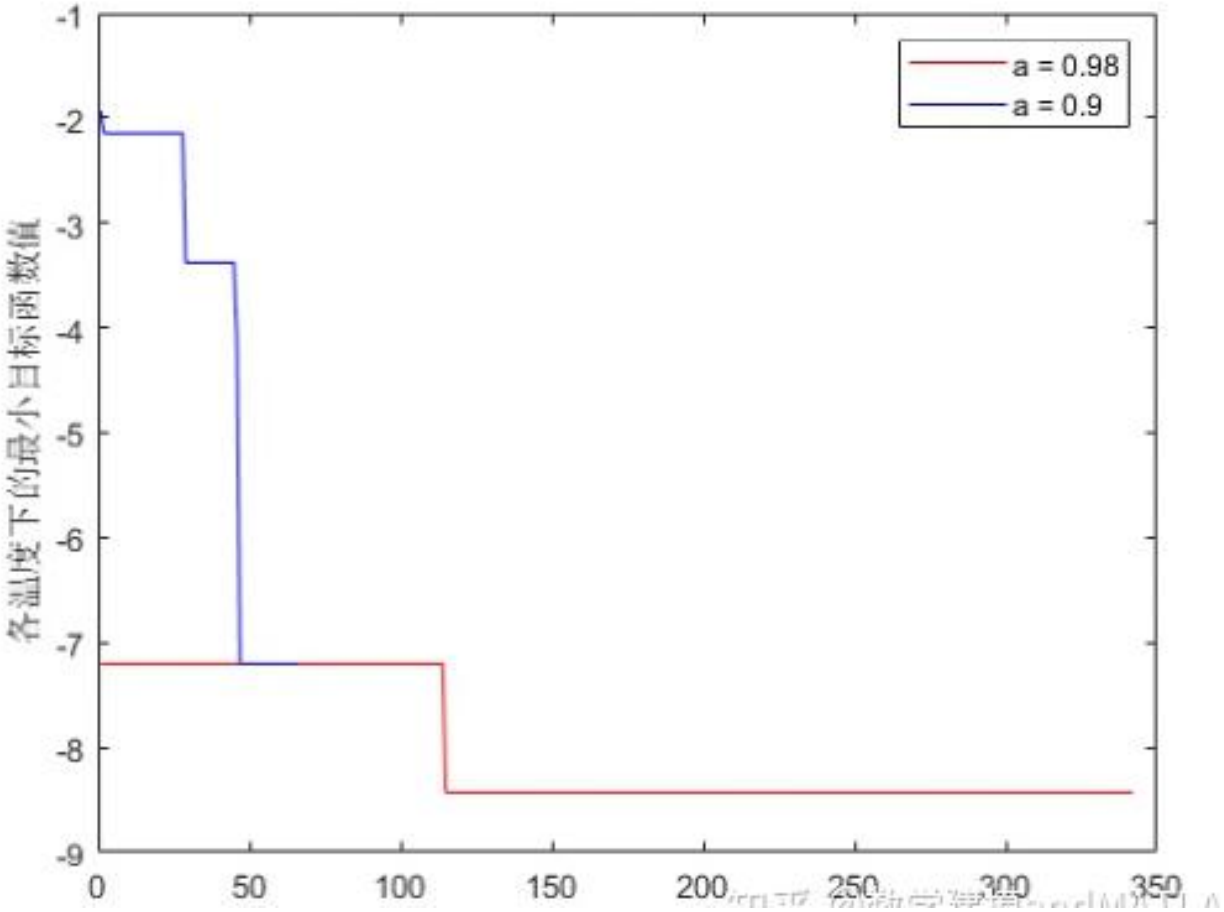
4 温度更新函数

即温度的下降方式，用于在外循环中修改温度值。

衰减量“以小为宜”

✓实验表明降温速度越慢，获得高质量解的几率越大，但花费的计算时间将同时增加。

✓温度高时下降的慢些，温度低时下降的快些。



温度衰减系数越小，温度下降越快，对应的迭代次数也就越少，算法搜索的次数也就越少，因此导致了上图中温度衰减系数0.9的模拟退火算法没有搜索到最优解。

$t_{k+1} = \lambda t_k \quad 0 < \lambda < 1$

$\lambda = 0.95$ Kirkpatrick首先提出

被Johnson,Bonomi及Lutton采用

λ 取0.5至0.99之间

Nahar及Skiscim等人把 $[0, t_0]$ 划分成K个小区间，温度更新函数为

$$t_k = \frac{K-k}{K} t_0, k = 1,2,\cdots,K$$

5 内循环终止准则（Metropolis抽样稳定准则）

用于决定在各温度下产生候选解的数目。

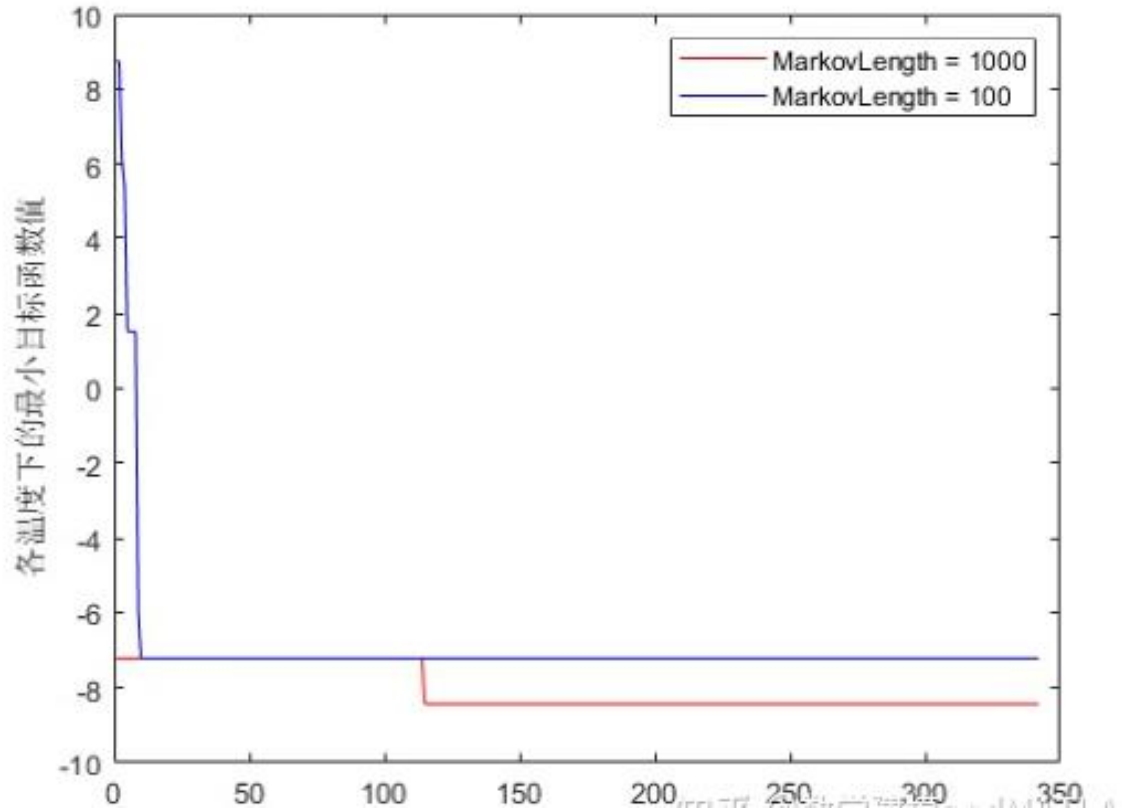
非时齐SAA：每个温度下只产生一个或少量候选解。

时齐算法——常用Metropolis抽样稳定准则包括：

- 检验目标函数值的均值是否稳定
- 连续若干步目标函数值的变化较小
- 按一定的步数抽样

马尔科夫链长度越长，模拟退火算法搜索越充分，更容易搜索到全局最优解，但相应的搜索时间会更长。

实验表明高温时迭代次数越多越好，低温时迭代次数可以适当减少。



6 外循环终止准则（算法终止准则）

用于决定算法何时结束。

✓理论上要求温度终值趋于零 $e_r = \left| \dfrac{f^* - f_{opt}}{f_{opt}} \right|$

✓通常的做法包括：

- 设置终止温度的阀值
- 设置外循环迭代次数(6-50)
- 算法搜索到的最优值连续若干步保持不变
- 检验系统熵是否稳定

SA流程

关键环节

1 初温、初始解

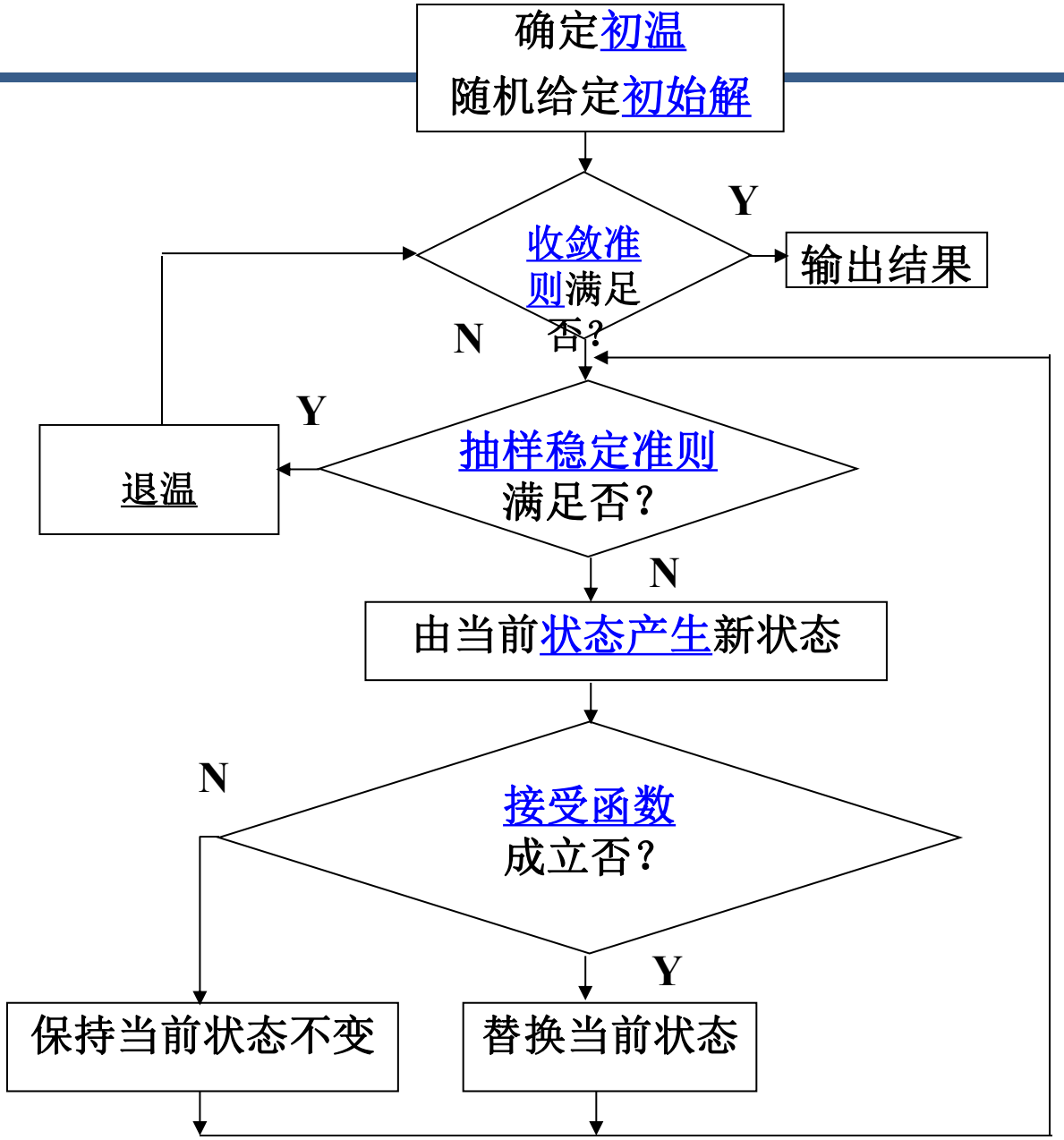
2 状态产生函数

3 状态接受函数

4 退温函数

5 抽样稳定准则

6 收敛准则



模拟退火算法基本要素和设定方法

功能意义	基本要素	设置方法
影响模拟退火算法全局搜索性能的重要因素之一。 实验表明，初温越大，获得高质量解的几率越大，但花费的计算时间将增加。	初始温度	1、均匀抽样一组状态，以各状态目标值的方差定初温 2、随机产生一组状态，以两两状态间最大差值定初温 3、利用经验公式给出初温
状态空间与状态产生函数。 邻域函数（状态产生函数）应尽可能保证产生的候选解遍布全部解空间。	邻域函数	候选解一般采用按照某一概率密度函数对解空间进行随机采样来获得。 概率分布可以是均匀分布、正态分布、指数分布等等
指从一个状态X_k(一个可行解)向另一个状态X_{new}(另一个可行解)的转移概率，通俗的理解是接受一个新解为当前解的概率	接受概率	一般采用Metropolis准则 $P_{ij}^{T}=\begin{cases}1, & \text{if } E(j)\le E(i)\\ e^{-\left(\frac{E(j)-E(i)}{KT}\right)}=e^{-\left(\frac{\Delta E}{KT}\right)}, & \text{otherwise}\end{cases}$
指从某一较高温状态t_0向较低温状态冷却时的降温管理表，或者说降温方式	冷却控制	1、经典模拟退火算法的降温方式 $t_k=\frac{t_0}{\lg(1+k)}$ 2、快速模拟退火算法的降温方式 $t_k=\frac{t_0}{1+k}$
内层平衡也称Metropolis抽样稳定准则，用于决定在各温度下产生候选解的数目	内层平衡	1、检验目标函数的均值是否稳定 2、连续若干步的目标值变化较小 3、预先设定的抽样数目，内循环代数
算法的终止条件	终止条件	1、设置终止温度的阈值 2、设置外循环迭代次数 3、算法搜索到的最优值连续若干步保持不变 4、检验系统熵是否稳定

SA特点

- 可以保证全局最优
- 特别适合组合优化问题
- 可以随机选择初始解
- 简单易行，通用性好

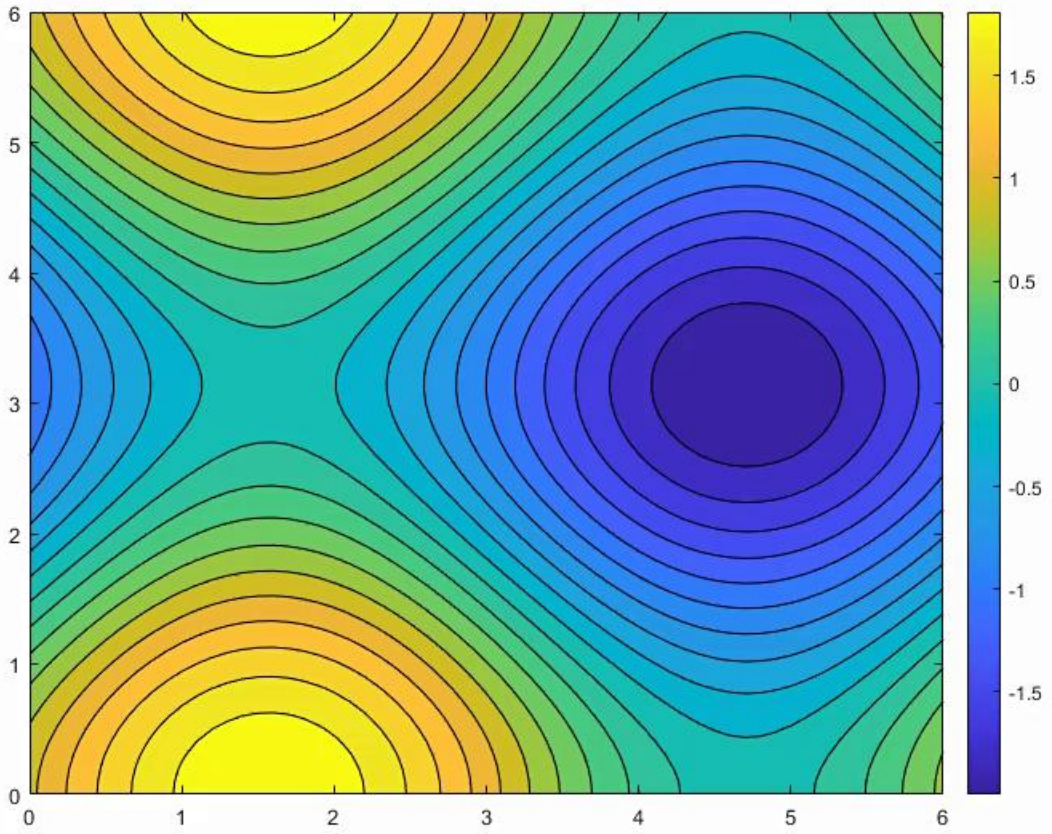
模拟退火算法在求解规模较大的实际问题时，往往存在以下缺点：

（1）收敛速度比较慢。

（2）尽管理论上只要计算时间足够长，模拟退火法就可以保证以概率1收敛于全局最优点。但是在实际算法的实现过程中，由于计算速度和时间的限制，在优化效果和计算时间二者之间存在矛盾，因而难以保证计算结果为全局最优点，优化效果不甚理想。

（3）在每一温度下很难判定是否达到了平衡状态。

寻找目标函数f = sin(x) + cos(y) 在[0, 6]×[0, 6]范围内的最小值。



1、组合优化问题的求解

数学模型

一个商人欲到 n 个城市推销商品，每两个城市 i 和 j 之间的距离为 d_{ij}，如何选择一条道路使得商人每个城市走一遍后回到起点且所走路经最短。

解空间

$$S=\{(\pi_1,\pi_2,\cdots,\pi_n)\,|\,(\pi_1,\pi_2,\cdots,\pi_n)\text{为}(1,2,\cdots,n)\text{的循环排列}\}$$

目标函数

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

初始解

选为 $(1,2,\cdots,n)$

新解的产生 (5 4 1 7 9 8 6 2 3)

1、互换操作(SWAP) (5 8 1 7 9 4 6 2 3)

随机交换两个不同城市的位置。

2、逆序操作(INV) (5 8 9 7 1 4 6 2 3)

两个不同随机位置的城市逆序。

3、插入操作(INS) (5 8 4 1 7 9 6 2 3)

随机选择某个城市插入到不同随机位置。

初值　$t_0 = \dfrac{-\Delta f}{\ln p}$　$p = 0.8$

衰减函数　$t_{k+1} = \lambda t_k$　$0 < \lambda < 1$

马氏链长　$L_k = 20$

停止准则
- 设计终止温度的阀值
- 设计外循环迭代次数
- 算法搜索到的最优值连续若干步保持不变

例1 30城市TSP问题（d^*=423.741 by D B Fogel）

- **TSP Benchmark 问题**

41 94;37 84;54 67;25 62;

7 64;2 99;68 58;71 44;54

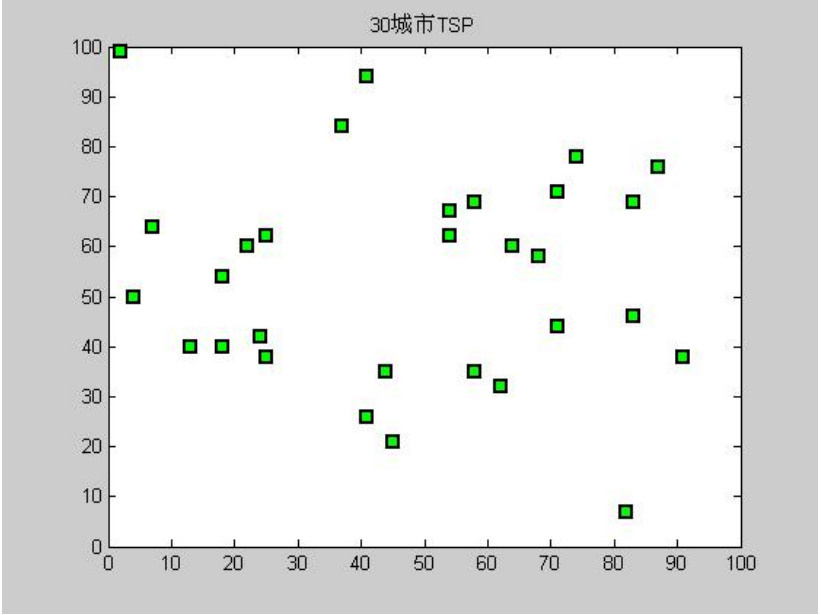
62;83 69;64 60;18 54;22

60;83 46;91 38;25 38;24

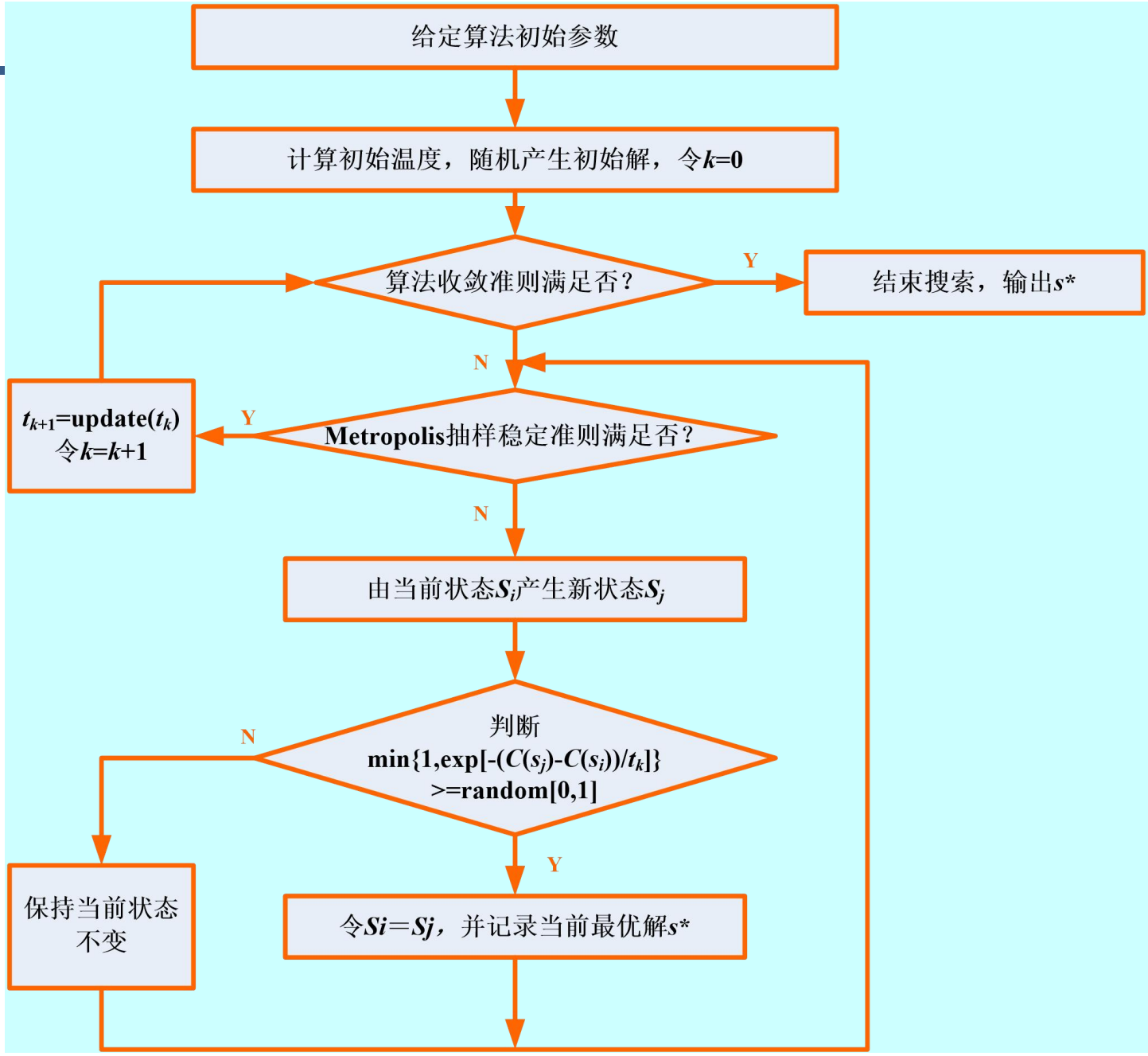
42;58 69;71 71;74 78;87

76;18 40;13 40;82 7;62 32;

58 35;45 21;41 26;44 35;4 50



算法流程

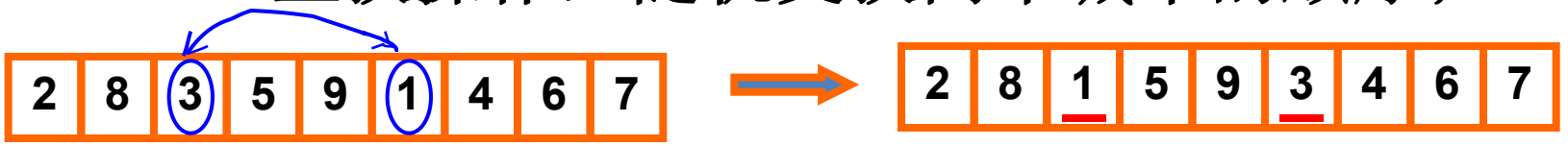

给定算法初始参数
计算初始温度，随机产生初始解，令k=0
算法收敛准则满足否？
Y
结束搜索，输出s*
N
Metropolis抽样稳定准则满足否？
Y
t_{k+1}=update(t_k)
令k=k+1
N
由当前状态S_i产生新状态S_j
判断
min{1,exp[-(C(s_j)-C(s_i))/t_k]}
>=random[0,1]
N
Y
保持当前状态
不变
令Si=Sj，并记录当前最优解s*

- **初始温度的计算**

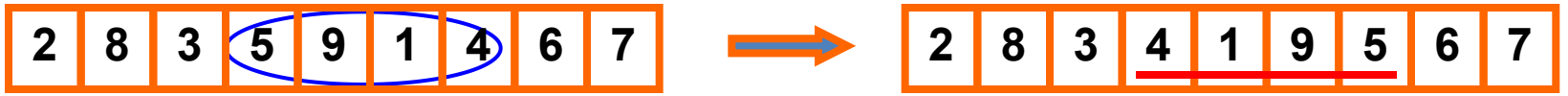
```
for i=1:100
    route=randperm(CityNum);
    fval0(i)=CalDist(dislist,route);
end
t0=-(max(fval0)-min(fval0))/log(0.9);
```

- **状态产生函数的设计**

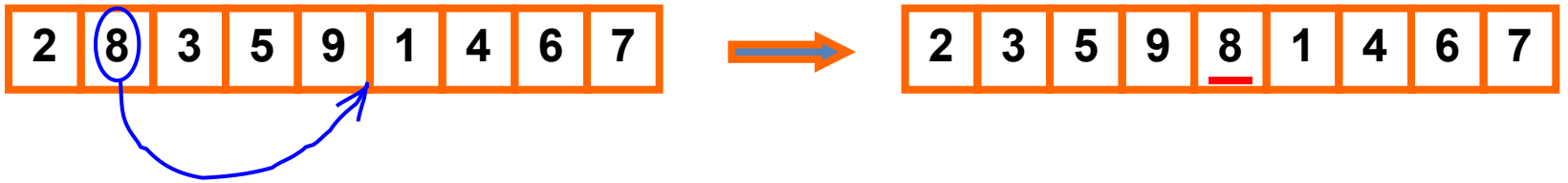
（1）互换操作，随机交换两个城市的顺序；



（2）逆序操作，两个随机位置间的城市逆序；



（3）插入操作，随机选择某点插入某随机位置。



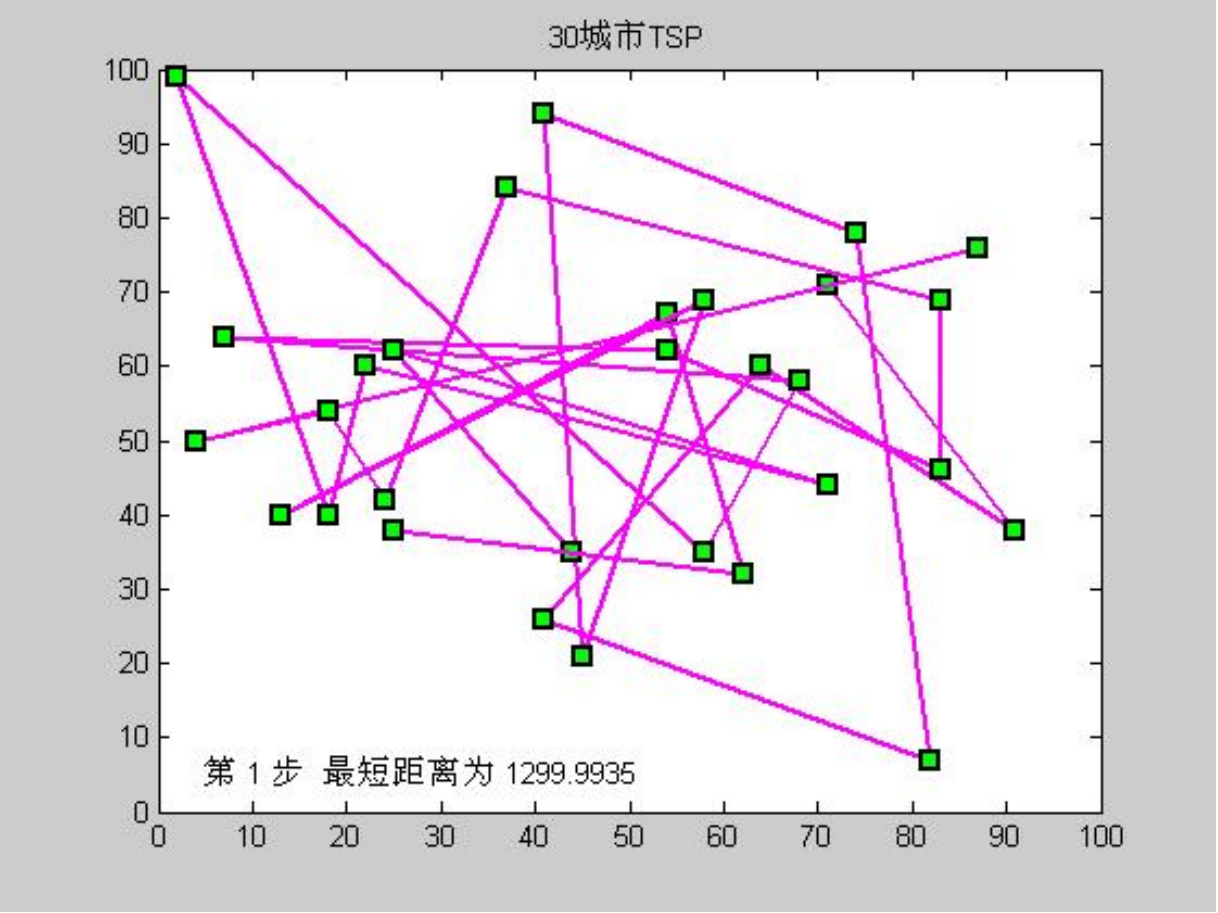
- **参数设定**

  截止温度 tf=0.01;

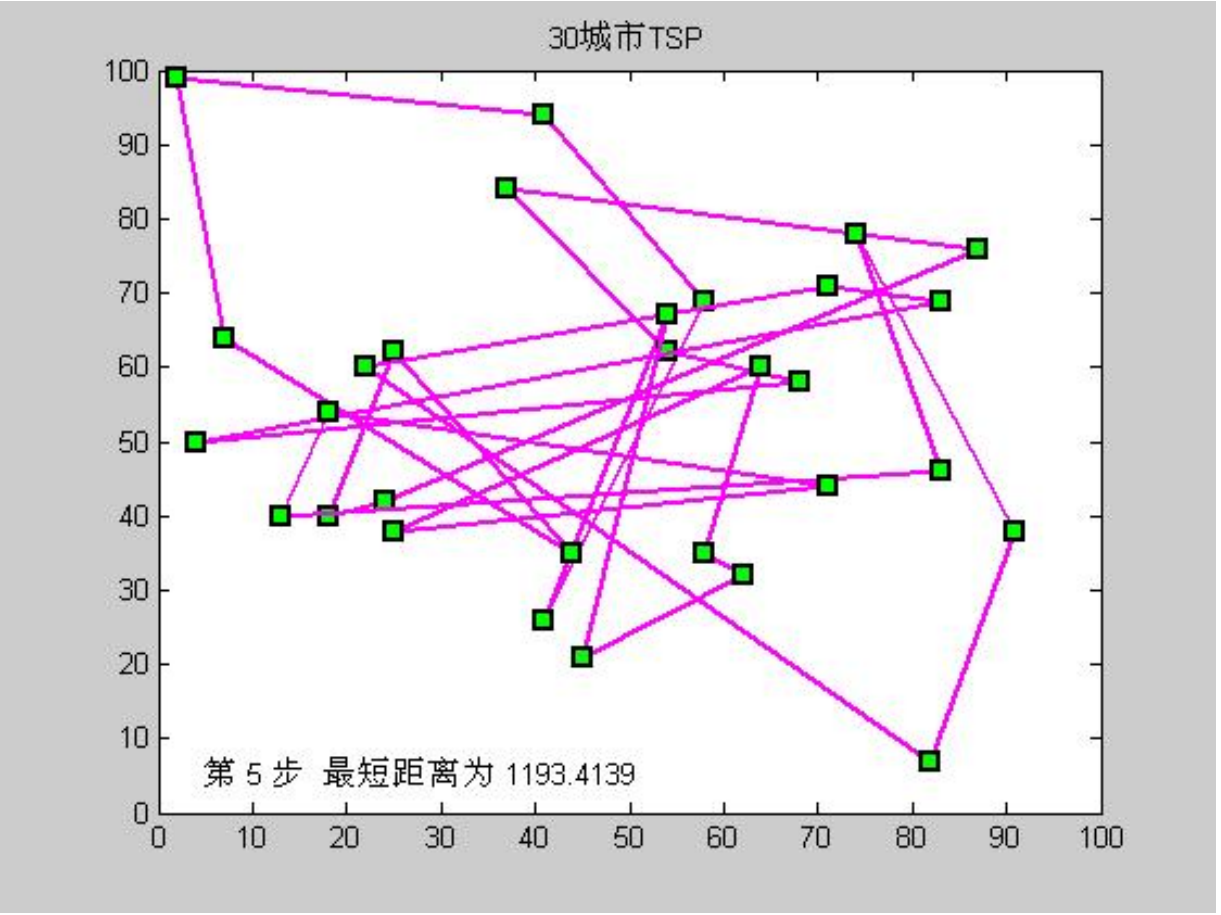
  退温系数 alpha=0.90;

  内循环次数 L=200*CityNum;

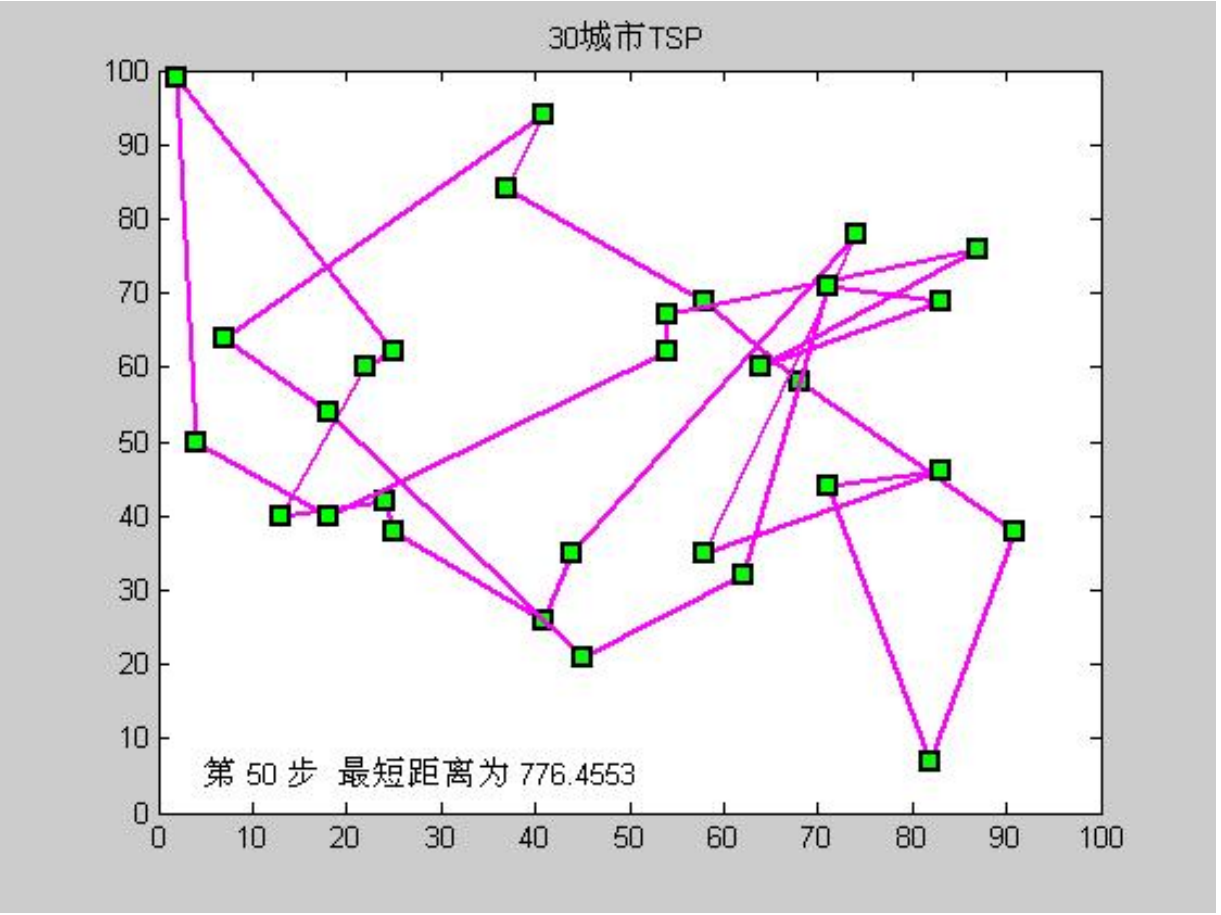
- 运行过程



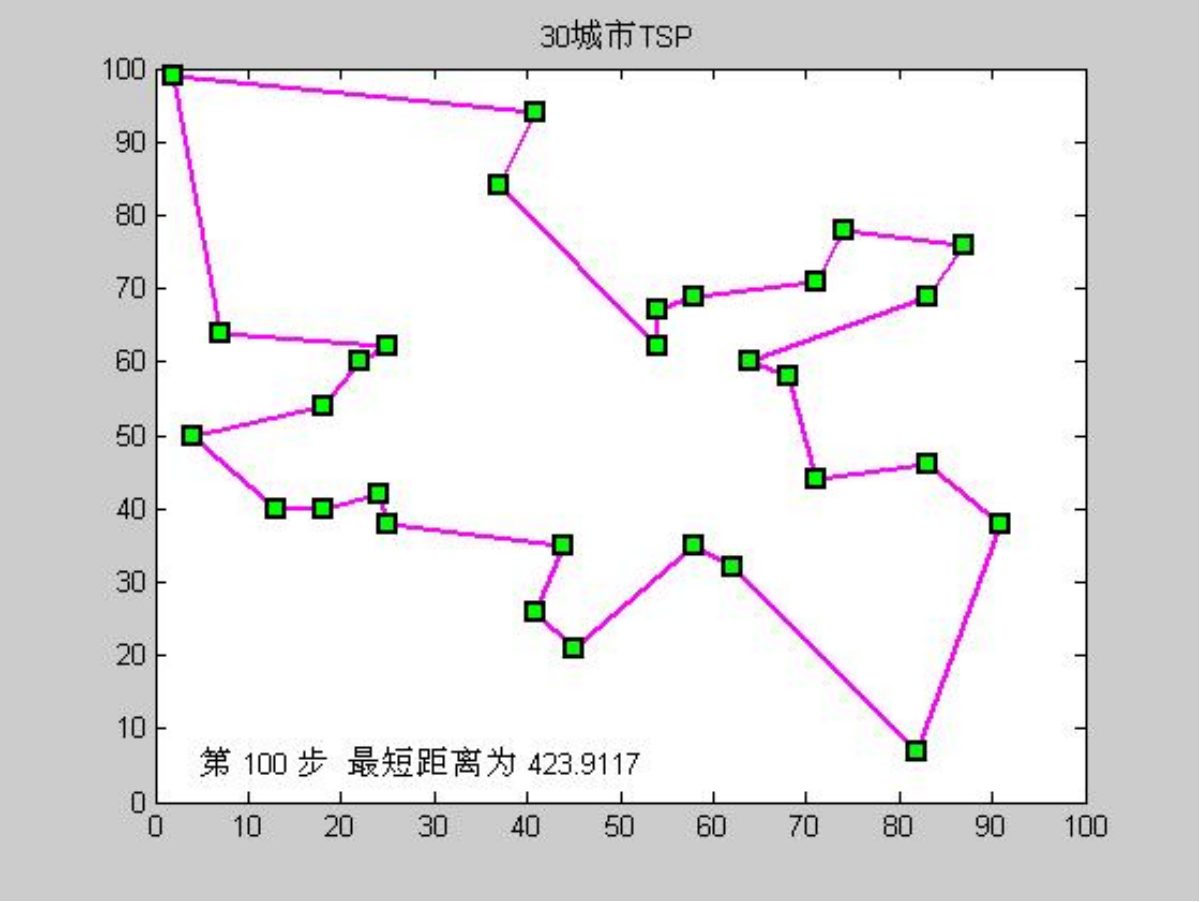
- 运行过程



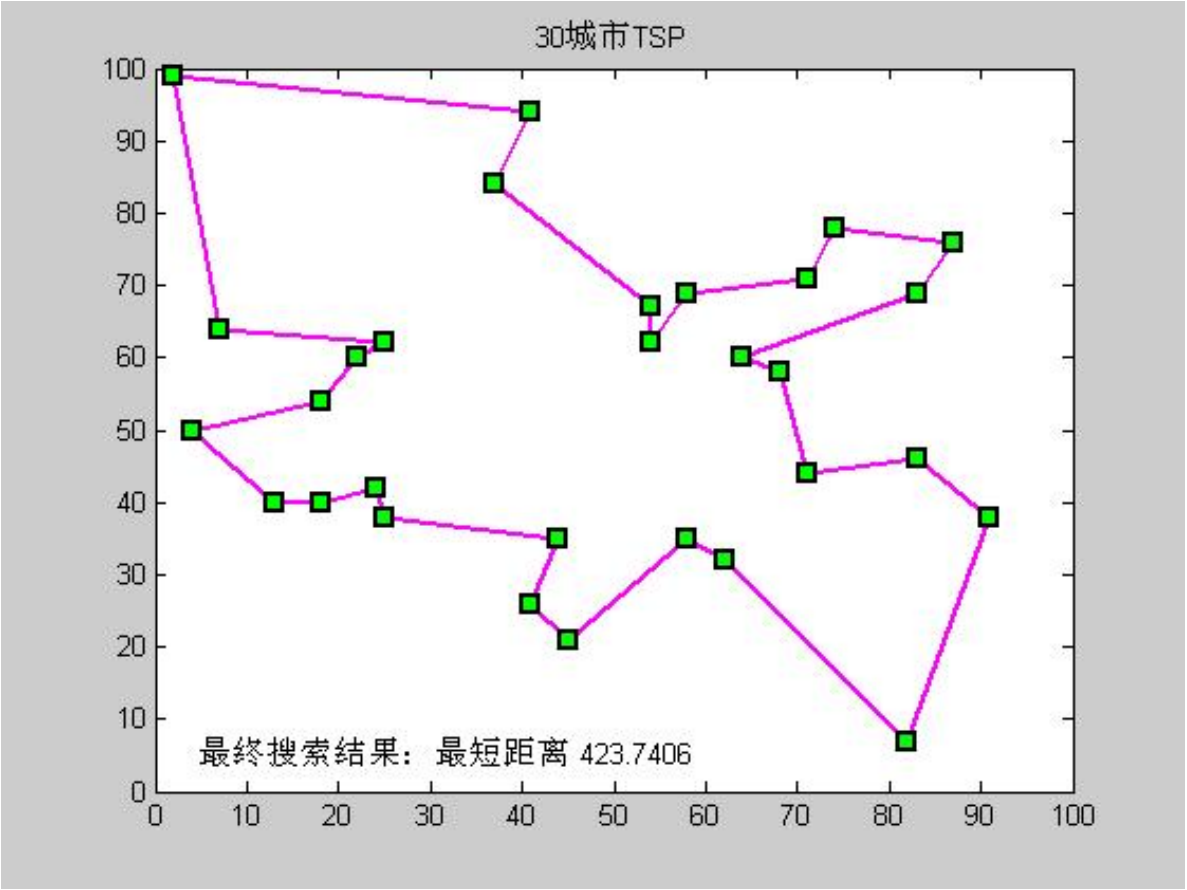
- 运行过程



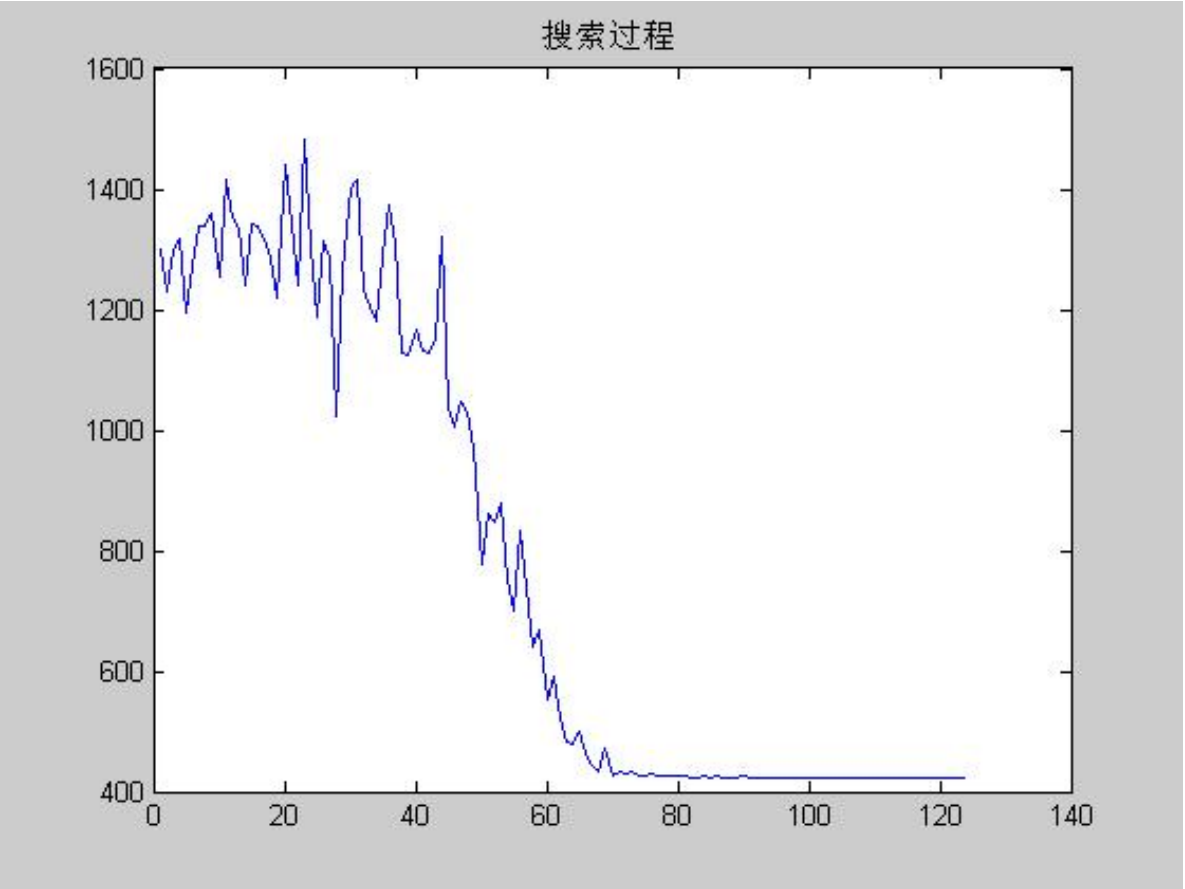
- 运行过程



- 运行过程



- 运行结果



函数优化问题的求解

求解 $f = x + 10sin(5x) + 7cos(4x)$ 在 $x \in [0, 9]$ 上的最大值

```
% 目标函数
function f=target(x)
f=x + 10 * sin(5 .* x) + 7 * cos(4 .* x);
end
```

```
lower_bound = 0;
higher_bound = 9;
```

1参数设置

```
x_new = lower_bound+rand*(higher_bound-lower_bound);
x_new = x_new*0.1;    % 从x较小开始搜索，直至找到最值
```

```
x_current = x_new;      % 初始化局部最优x
x_best = x_new;         % 初始化全局最优x
y_current = inf;        % 初始化局部最优y
y_best = inf;           % 初始化全局最优y
```

```
t = 1000;
tf = 3;
ratio = 0.9;
Markov_length = 20;
```

2 程序内外循环

```
while t>tf
    for i=1:Markov_length
    end
    if flag==0
        break
    else
        t=t*ratio;
    end
end
```

内循环：温度 t不变的情况下，执行 Metropolis准则判断

flag：控制内循环程序停止

外循环：温度 t不断的降低

3 程序内循环：温度 t不变，执行Metropolis采样 准则判断

```
for i=1:Markov_length
```

```
x_new = x_new+new;
if (x_new<lower_bound||x_new>higher_bound)
    x_new = x_new-3*new;
end

y_new=-target(x_new);          % 寻找最大值，
```

解的产生：在其领域内的扰动

```
% Metroplis准则寻找最大值
```

```
flag=y_current;                % 用于控制退出循环条件
if y_new <  y_current
    x_current = x_new;
    y_current = y_new;
    if y_new < y_best
        x_best = x_new;
        y_best = y_new;
    end
else
    if rand<exp(-(y_new-y_current)/t)
        x_current = x_new;
        y_current = y_new;
    else
        x_new=x_current;
    end
end
```

flag：控制内循环程序停止

```
if abs(flag-y_current)<0.001
    flag=0;
    break
end
```

```
end
```

模拟退火与传统迭代最优算法的比较：

（1）从局部最优中跳出是非常可能的，因此不会陷入局部最优。

（2）系统最终状态的总特征可以在较高温度下看到，而状态的好的细节却在低温下表现，因此，模拟退火是自适应的。

传统的优化方法

1、依赖于初始条件。
2、与求解空间有紧密关系，促使较快地收敛到局部解，但同时对解域有约束，如可微或连续。利用这些约束，收敛快。
3、有些方法，如Davison-Fletcher-Powell直接依赖于至少一阶导数；共轭梯度法隐含地依赖于梯度。

全局优化方法

1、不依赖于初始条件；
2、不与求解空间有紧密关系，对解域，无可微或连续的要求。求解稳健，但收敛速度慢。能获得全局最优。适合于求解空间不知的情况

练习作业：写出模拟退火算法求解以下问题最小值的详细步骤

寻找目标函数f = x + 10 sin(3x) + cos(x) 在[0, 9]范围内的最小值。