基于python演示的模拟退火算法

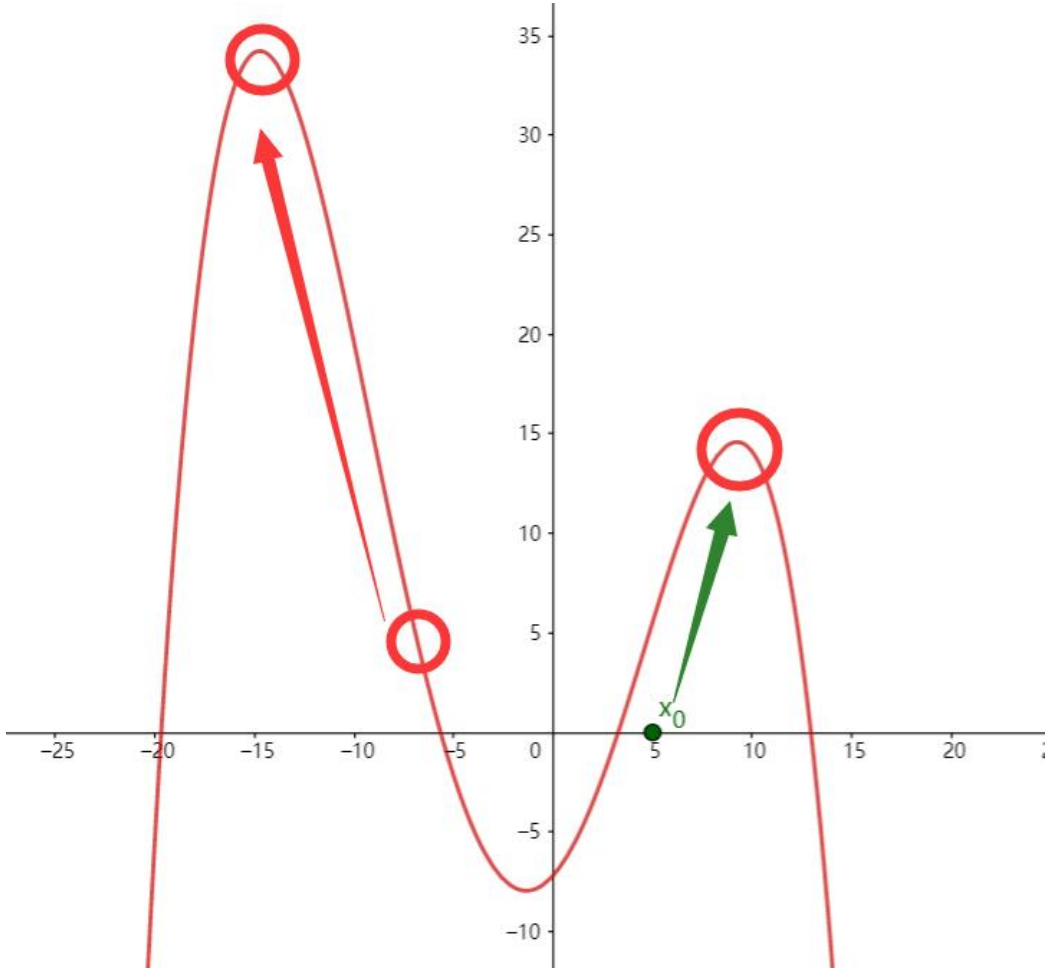
B21031703郑雨涵

目录

1. 算法思想

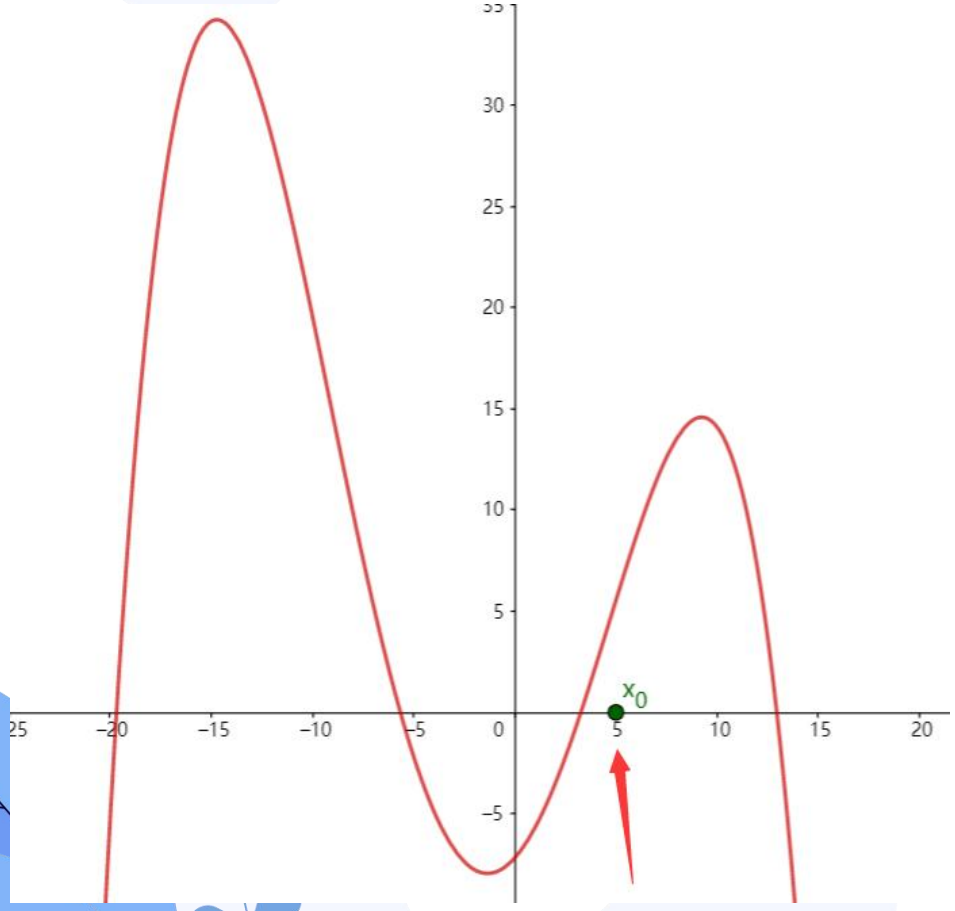
模拟退火算法是一种基于概率的全局优化算法，其灵感来源于固体退火过程。在物理学中，退火是指将材料加热至高温，然后缓慢冷却以减少材料内部应力的过程。在这个过程中，材料的微观结构会因为温度的变化而重组，从而达到更稳定的状态。

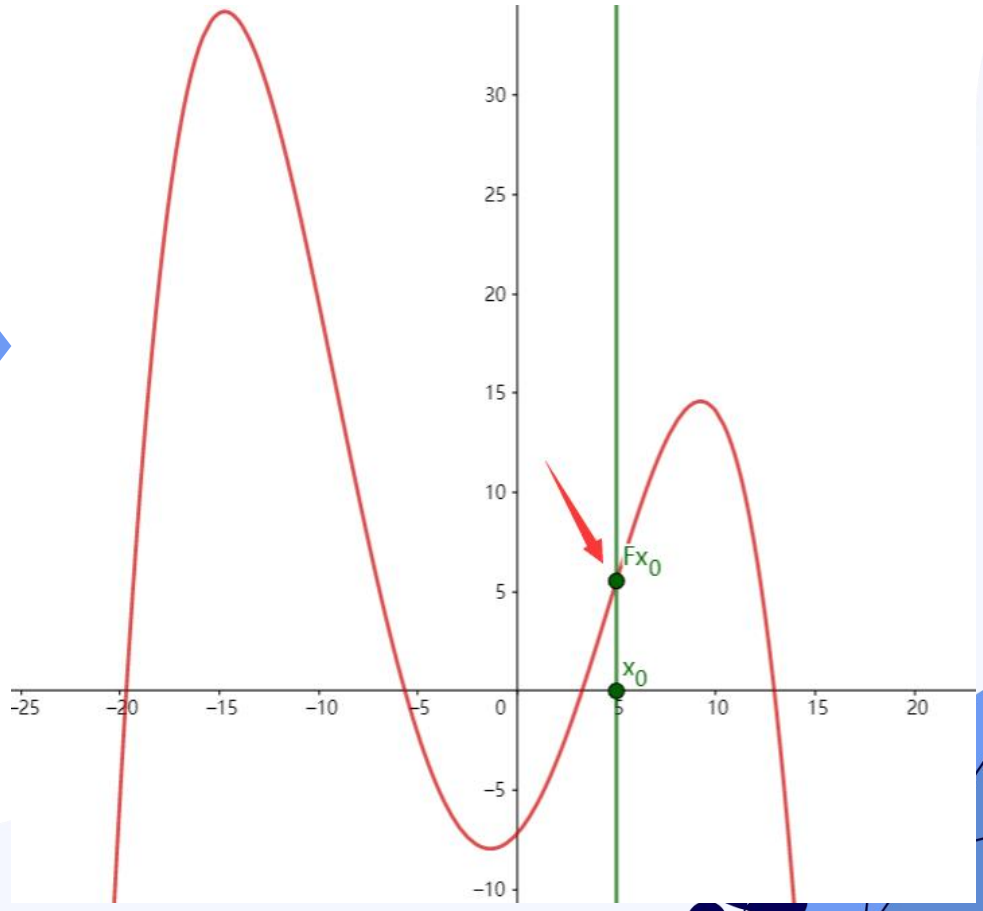
模拟退火算法的核心思想是在搜索过程中接受比当前解差的解，以跳出局部最优解。



目标：找到函数的最大值。

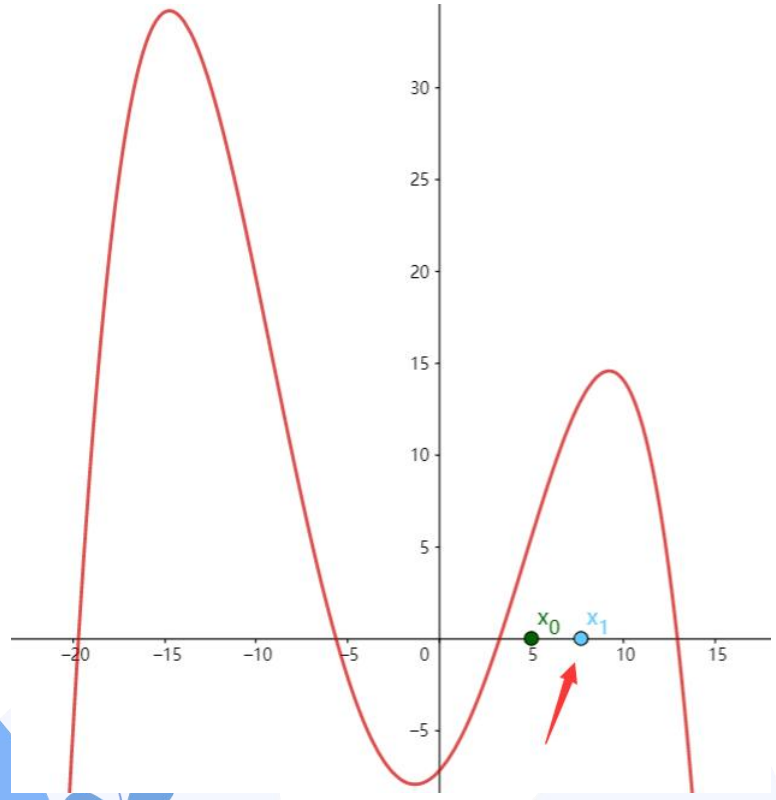
① 随机在函数上取一点x_0，找到其对应的函数值$F(x_0)$。

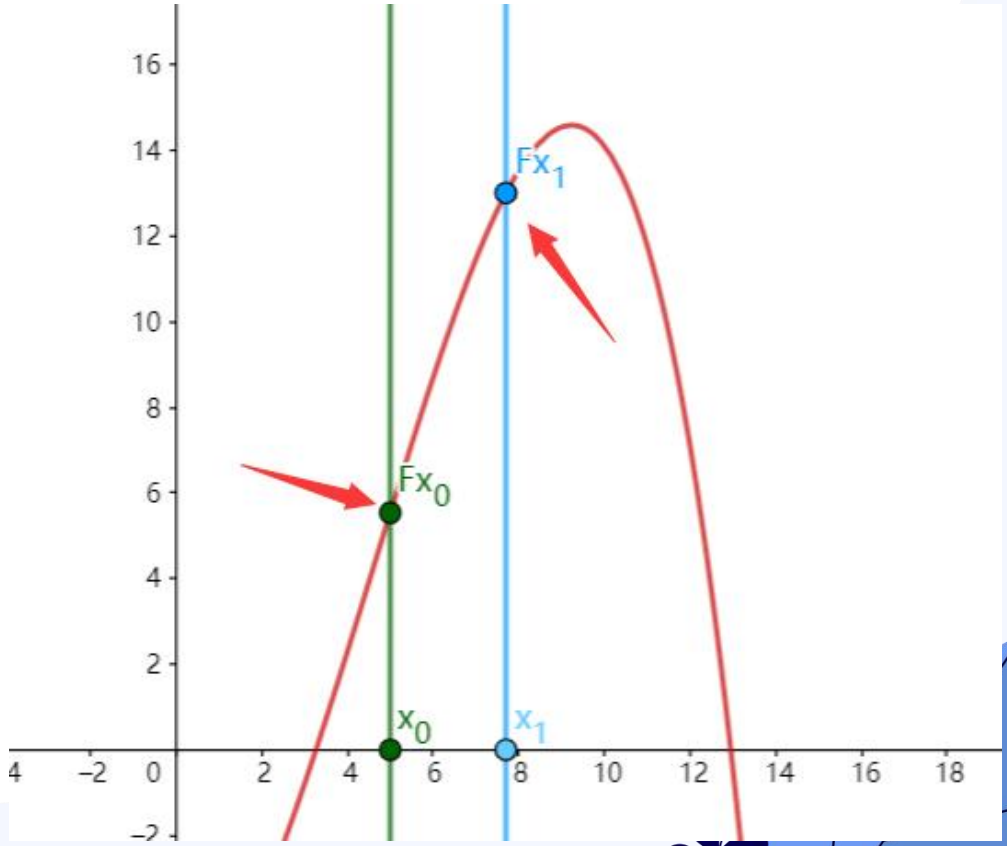




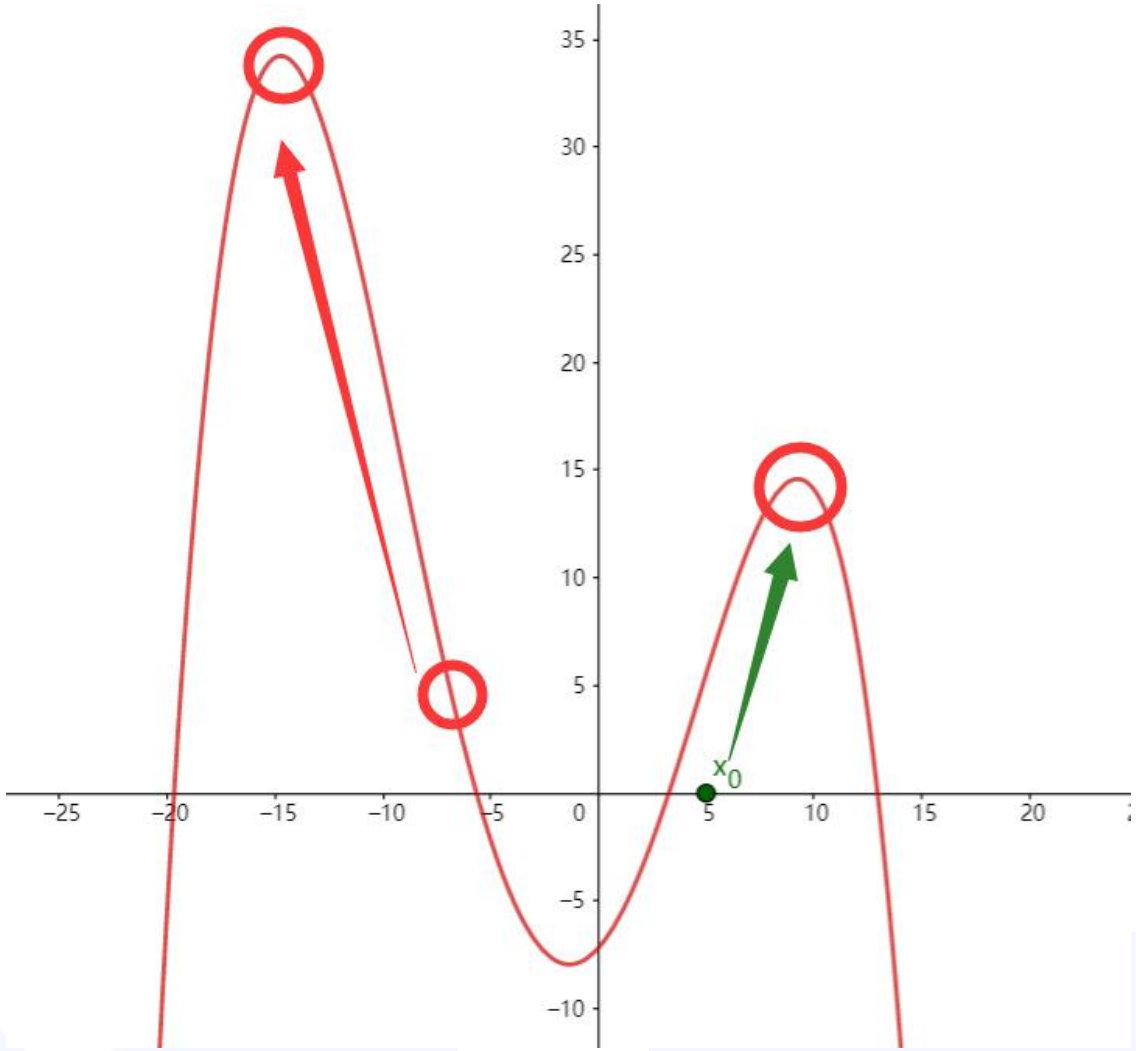
② x_0模拟粒子无序运动，假设x_0向右移动到了x_1。

③ 算法倾向于接受更“好”的结果，由于$F(x_1)>F(x_0)$，所以更新x为x_1的值。





④ 假设x_0是向左移动了一点，取到了一个“更差”的结果，那么算法以一定概率接受更差的状态。



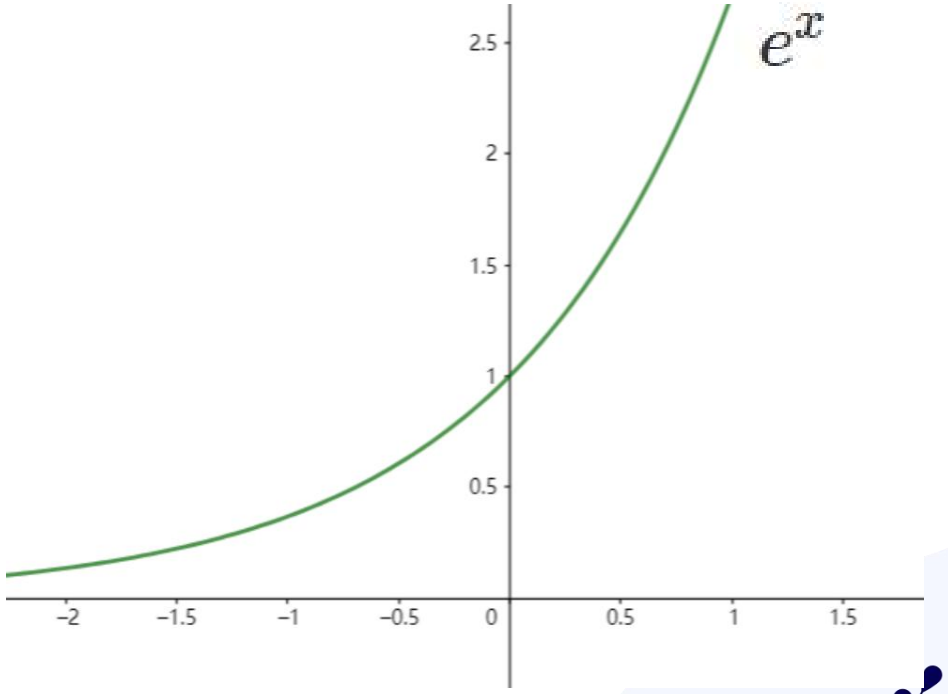
以“一定概率“接受更差的状态。
Metropolis准则

接受概率：$e^{\frac{\Delta f}{kT}}$

K：一个物理常数；

T：当前温度；

$\Delta f = -|f(x_0) - f(x_1)|$



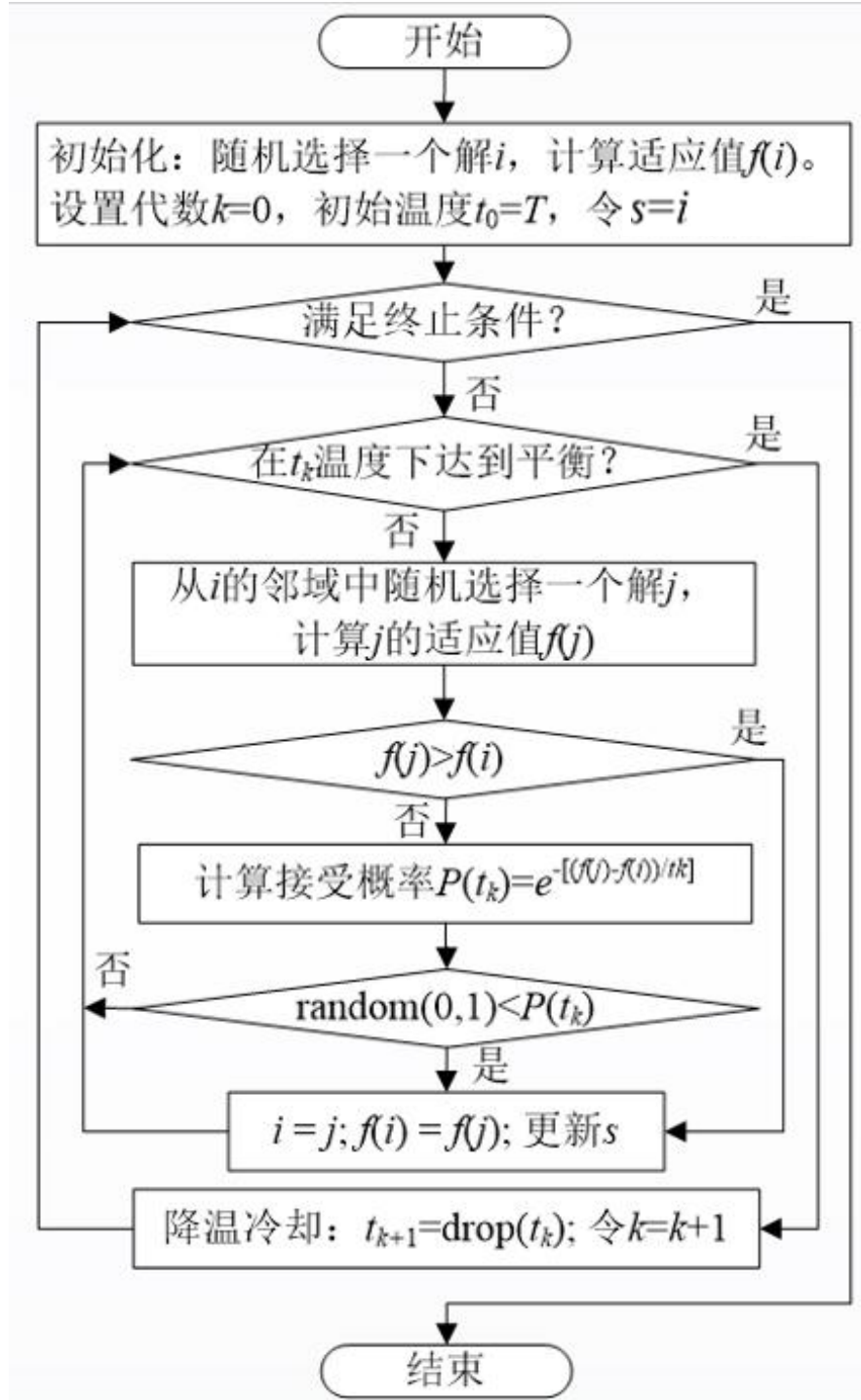
2. 算法流程

退火前的准备：

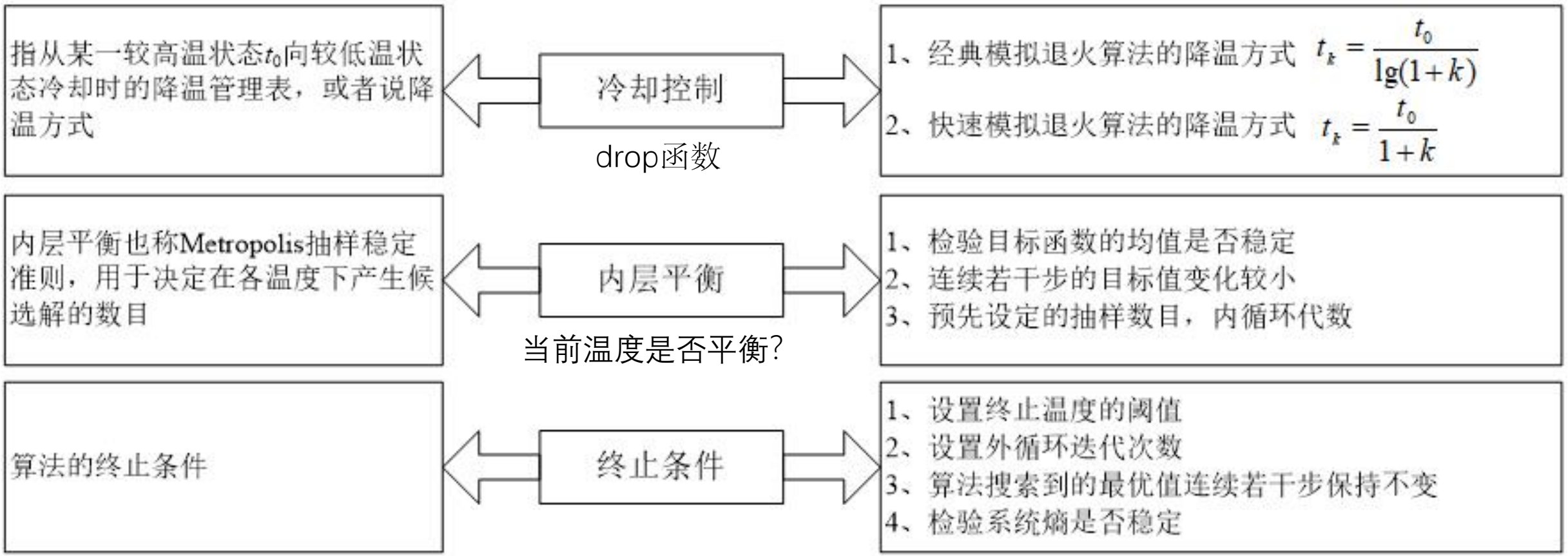
① 初始温度T_0；

② 退火的程度（每次降低多少温度），假设由函数drop(x)决定；

③ 退到什么程度结束，即设定跳出循环的条件（先简单地取一个终止温度T_1）。


开始
初始化：随机选择一个解i，计算适应值$f(i)$。
设置代数k=0，初始温度t_0=T，令s=i
满足终止条件？
是
否
在t_k温度下达到平衡？
是
否
从i的邻域中随机选择一个解j，
计算j的适应值$f(j)$
$f(j)$>$f(i)$
是
否
计算接受概率$P(t_k)$=$e^{-[(f(j)-f(i))/tk]}$
否
random(0,1)<$P(t_k)$
是
i=j; $f(i)$=$f(j)$; 更新s
降温冷却：t_{k+1}=drop(t_k); 令k=k+1
结束

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 $P_{ij}^{T}=\begin{cases}1, & \text{if } E(j)\le E(i)\\ e^{-(\frac{E(j)-E(i)}{KT})}=e^{-(\frac{\Delta E}{KT})}, & \text{otherwise}\end{cases}$


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
drop函数
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
当前温度是否平衡?
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

3. 应用举例

例3.1 已知背包的装载量为c=10，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。

假设问题的一个可行解用0和1的序列表示，例如i=(10100)表示选择第1和第3个物品，而不选择第2、第4和第5个物品。

```python
weights = [2, 3, 5, 1, 4]
values = [2, 5, 8, 3, 6]
capacity = 10
initial_temperature = 100
drop = 0.99
stopping_temperature = 1e-8
K=1
```

```python
# 计算背包中物品的总价值
2 usages
def value_calculate(state, values):
    total_value = 0
    for i in range(len(state)):
        if state[i] == 1:
            total_value += values[i]
    return total_value
```

```python
# 计算背包中物品的总重量
2 usages
def weight_calculate(state, weights):
    total_weight = 0
    for i in range(len(state)):
        if state[i] == 1:
            total_weight += weights[i]
    return total_weight
```

```python
def simulated_annealing(weights, values, capacity, initial_temperature, drop, stopping_temperature,K):
    num_items = len(weights)
    current_state = [random.randint( a: 0,  b: 1) for _ in range(num_items)]
    current_weight = weight_calculate(current_state, weights)
    current_value = value_calculate(current_state, values)
    best_state = current_state.copy()
    best_weight = current_weight
    best_value = current_value
    temperature = initial_temperature
    while temperature > stopping_temperature: #终止条件
        k=0
        while k<3: #平衡条件
            new_state = current_state.copy()
            index = random.randint( a: 0, num_items - 1)  # 用随机函数寻找改变邻域
            new_state[index] = 1 - new_state[index]  # 改变状态，不是1就是0
            new_weight = weight_calculate(new_state, weights)
            new_value = value_calculate(new_state, values)
            if new_weight <= capacity:
                delta_f = -abs(new_value - current_value)
                if new_value > current_value or math.exp(delta_f / (temperature * K) > random.random()):
                    current_state = new_state
                    current_weight = new_weight
                    current_value = new_value
                    if current_value > best_value:
                        best_state = current_state.copy()
                        best_weight = current_weight
                        best_value = current_value
            k+=1
        temperature *= drop
    return best_state, best_value,best_weight
```

```
24  def simulated_annealing(weights, values, capacity, initial_temperature, drop, stopping_temperature,K):
25      num_items = len(weights)
26      current_state = [random.randint( a: 0,  b: 1) for _ in range(num_items)]
27      current_weight = weight_calculate(current_state, weights)
28      current_value = value_calculate(current_state, values)
29      best_state = current_state.copy()
30      best_weight = current_weight
31      best_value = current_value
32      temperature = initial_temperature
```

- randint将返回一个在[a, b]范围内的随机整数，包括a和b本身。
- 列表表达式[expression for item in iterable if condition]

  例：squares = [x*x for x in range(1, 11)]#1~10的幂
- for _ in range(num_items)里的_是一个合法的变量名， 表示这个变量在循环体内部不需要被使用。当我们只需要执行一定次数的循环而不需要使用循环变量的值时，可以使用_来代替一个有实际意义的变量名

```
while temperature > stopping_temperature: #终止条件
    k=0
    while k<3: #平衡条件
        new_state = current_state.copy()
        index = random.randint( a: 0, num_items - 1)  # 用随机函数寻找改变邻域
        new_state[index] = 1 - new_state[index]  # 改变状态，不是1就是0
        new_weight = weight_calculate(new_state, weights)
        new_value = value_calculate(new_state, values)
        if new_weight <= capacity:
            delta_f = -abs(new_value - current_value)
            if new_value > current_value or math.exp(delta_f / (temperature * K) > random.random()):
                current_state = new_state
                current_weight = new_weight
                current_value = new_value
                if current_value > best_value:
                    best_state = current_state.copy()
                    best_weight = current_weight
                    best_value = current_value
        k+=1
    temperature *= drop
```

选择的物品（0表示不选，1表示选）: [0, 0, 1, 1, 1]
总价值: 17
当前重量: 10

基于 D-Wave Advantage 的量子退火公钥密码攻击算法研究

王潮　王启迪　洪春雷　胡巧云　裴植

(上海大学 特种光纤与光接入网重点实验室　上海 200444)

摘　要　D-Wave 专用量子计算机的原理量子退火凭借独特的量子隧穿效应可跳出传统智能算法极易陷入的局部极值，可视为一类具有全局寻优能力的人工智能算法．本文研究了两类基于量子退火的 RSA 公钥密码攻击算法(分解大整数 $N=pq$)，一是将密码攻击数学方法转为组合优化问题或指数级空间搜索问题，通过 Ising 模型或 QUBO 模型求解，提出了乘法表的高位优化模型，建立新的降维公式，使用 D-Wave Advantage 分解了 200 万整数 2269753．大幅度超过普渡大学、Lockheed Martin 和富士通等实验指标，且 Ising 模型系数 h 范围缩小了 84%，系数 J 范围缩小了 80%，极大的提高了分解成功率，这是一类完全基于 D-Wave 量子计算机的攻击算法．二是基于量子退火算法融合密码攻击数学方法优化密码部件的攻击，采用量子退火优化 CVP 问题求解，通过量子隧穿效应获得比 Babai 算法更近的向量，提高了 CVP 问题中光滑对的搜索效率，在 D-Wave Advantage 上实现首次 50 比特 RSA 整数分解．实验表明，在通用量子计算机器件进展缓慢情况下，D-Wave 表现出更好的现实攻击能力，且量子退火不存在 NISQ 量子计算机 VQA 算法的致命缺陷贫瘠高原问题：算法会无法收敛且无法扩展到大规模攻击.

关键词　RSA；D-Wave；量子退火；CVP

中图法分类号　TP309

感谢观看