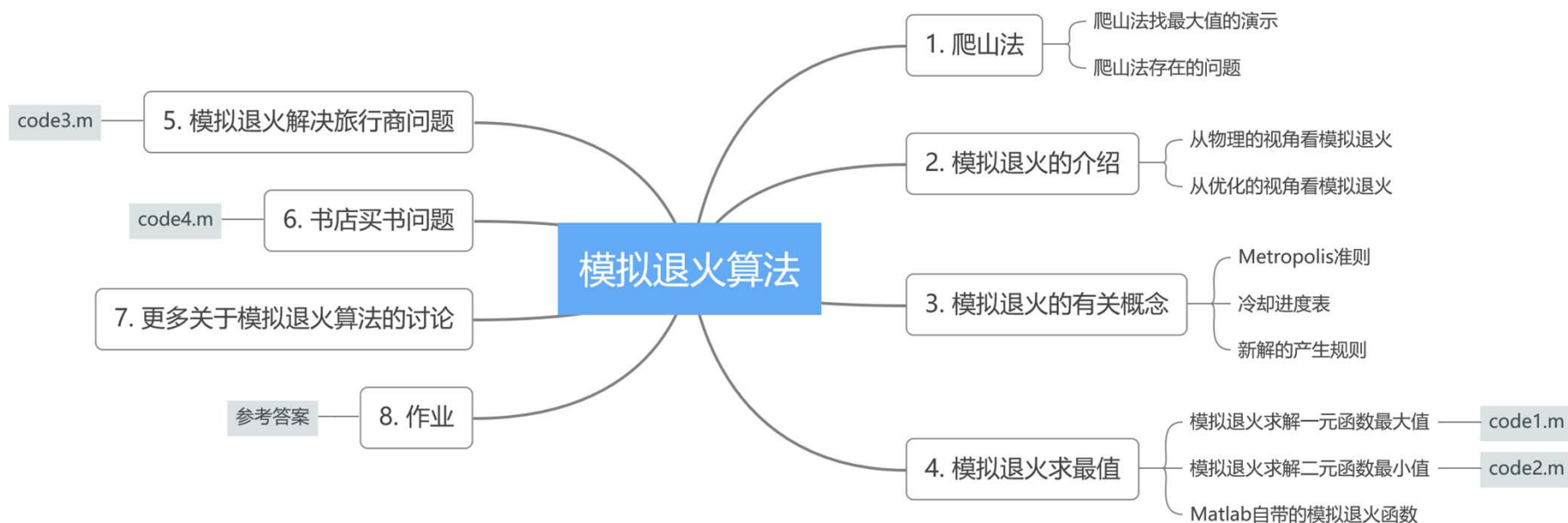
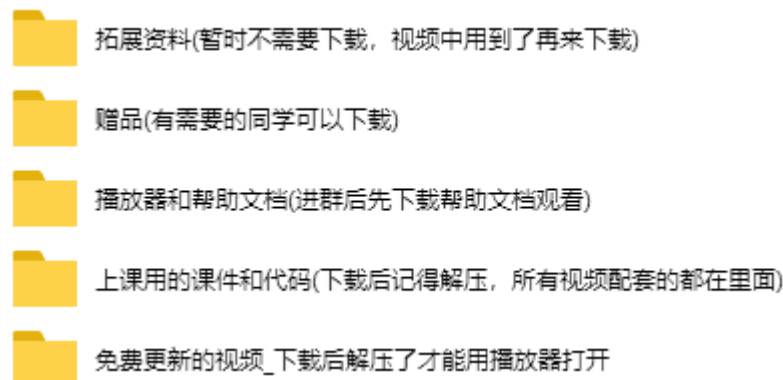


模拟退火算法



温馨提示

- (1) 视频中提到的附件可在**售后群的群文件**中下载。
包括**讲义、代码、我视频中推荐的资料**等。



(2) 关注我的**微信公众号《数学建模学习交流》**，后台发送**“软件”**两个字，可获得常见的建模软件下载方法；发送**“数据”**两个字，可获得建模数据的获取方法；发送**“画图”**两个字，可获得数学建模中常见的画图方法。另外，也可以看看公众号的历史文章，里面发布的都是对大家有帮助的技巧。

(3) **购买更多优质精选的数学建模资料**，可关注我的微信公众号《数学建模学习交流》，在后台发送**“买”**这个字即可进入店铺进行购买。

(4) 视频价格不贵，但价值很高。单人购买观看只需要**58元**，和另外两名队友一起购买人均仅需**46元**，视频本身也是下载到本地观看的，所以请大家**不要侵犯知识产权**，对视频或者资料进行二次销售。

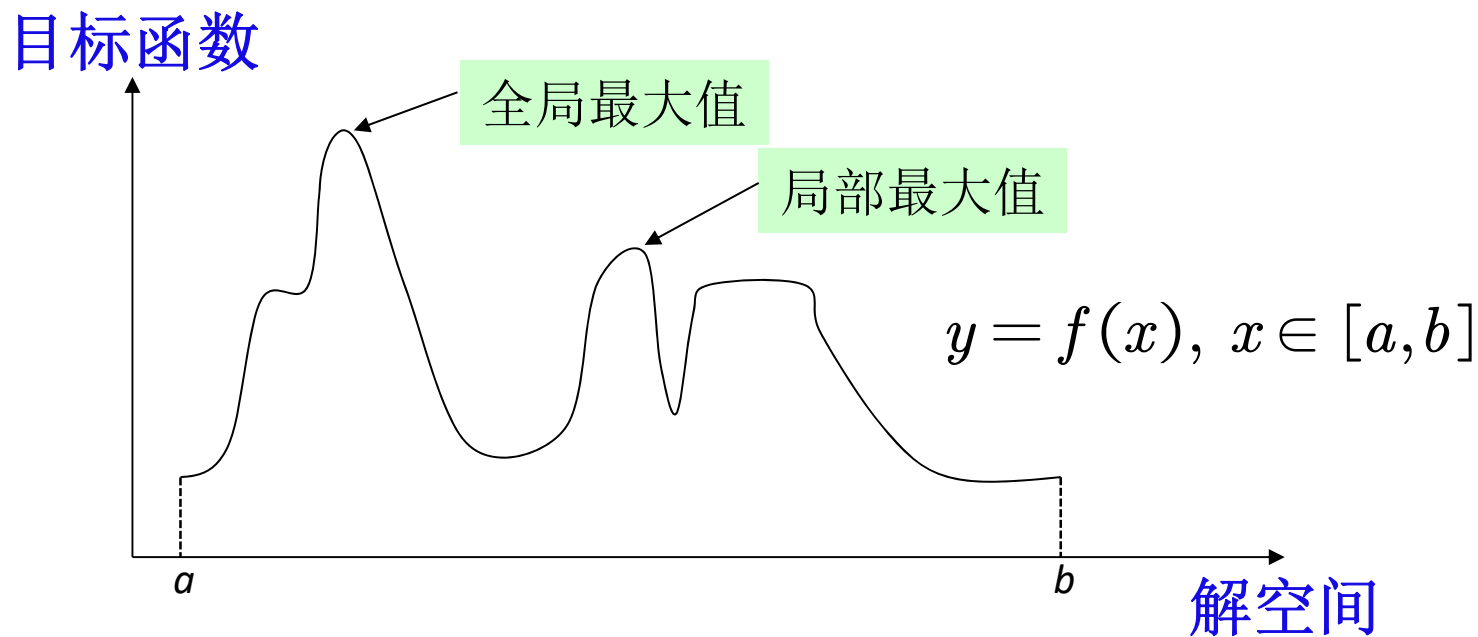
模拟退火算法

本节我们将学习第二个智能算法：模拟退火算法(Simulated Annealing Algorithm, 简记为SAA或SA), 它是基于金属退火机理而建立起来的一种最优化方法, 它能够以随机搜索技术从概率意义上找出目标函数的全局最值点。

与上一节学的粒子群算法相比, 模拟退火有如下特点:

- (1) 不仅能处理连续优化问题, 还能很方便的处理组合优化问题, 且编程简单易于实现, 目标函数的收敛速度较快。
- (2) 参数的选择至关重要, 初始参数的合理选取是保证算法的全局收敛性和效率的关键, 选择不当得到的结果可能会很差。
- (3) 面对不同的问题, 可以设计不同的退火过程, 不同过程得到的准确度和效率可能有较大差别, 因此需要一定的经验 (我们可以学习相关领域的学者提出的论文)。

最简单的例子



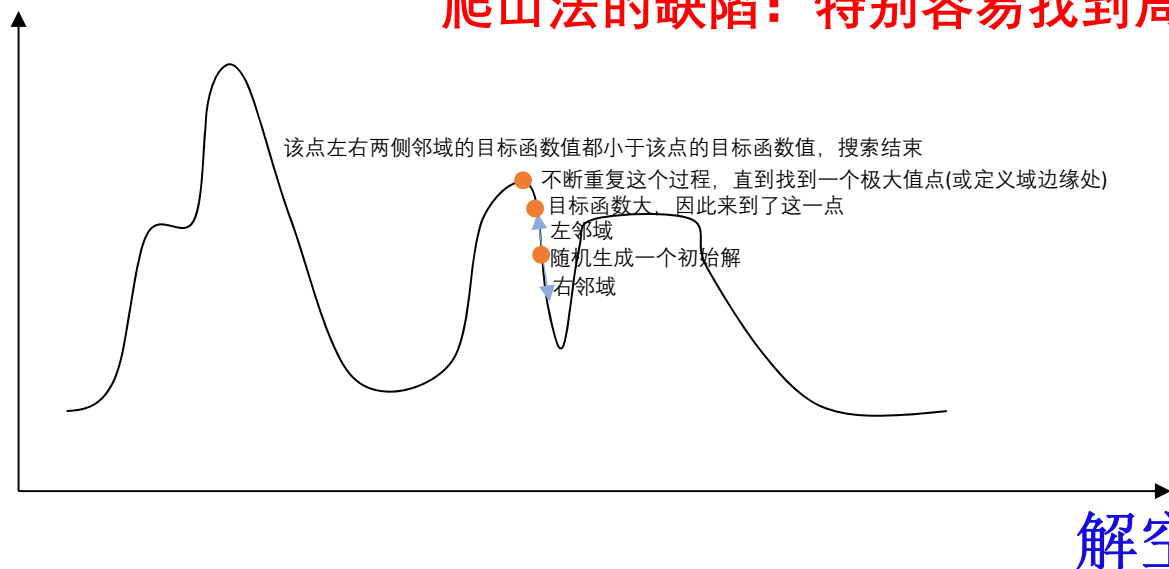
怎么找到这个一元函数的最大值?
(只有一个上下界约束, 即函数的定义域)

爬山法

爬山法找函数最大值

目标函数

爬山法的缺陷：特别容易找到局部最优解

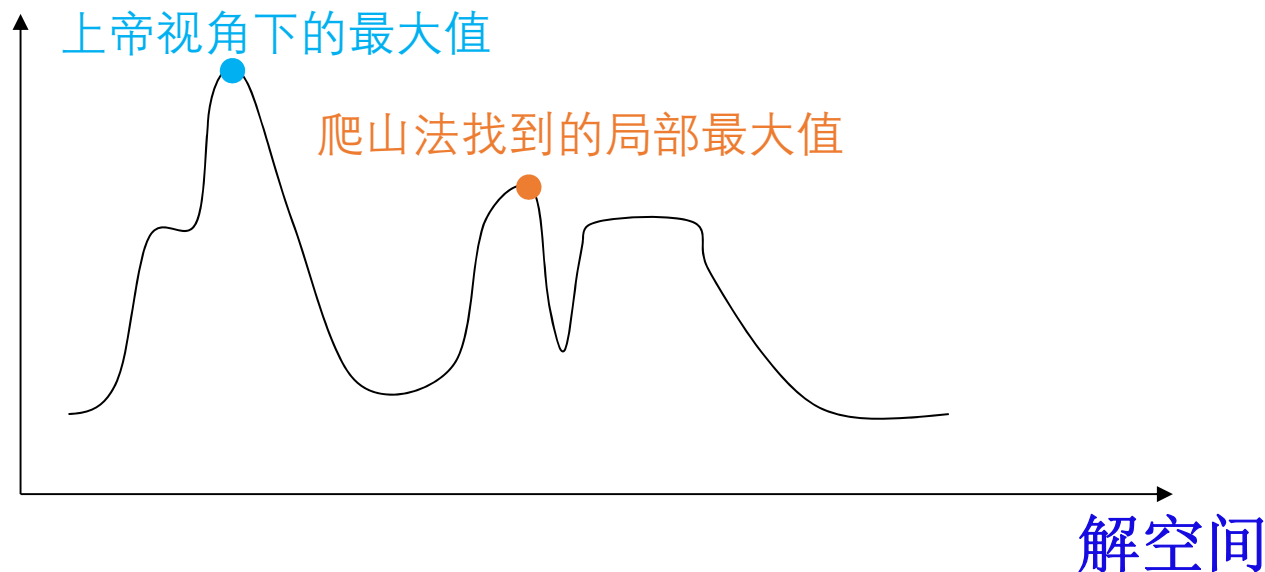


以最简单的连续一元函数找最大值为例, 我们来讲解爬山法:

- (1) 在解空间中随机生成一个初始解;
- (2) 根据初始解的位置, 我们在下一步有两种走法: 向左邻域走一步或者向右邻域走一步 (走的步长越小越好, 但会加大计算量);
- (3) 比较不同走法下目标函数的大小, 并决定下一步往哪个方向走。上图中向左走目标函数较大, 因此我们应该往左走一小步;
- (4) 不断重复这个步骤, 直到找到一个极大值点(或定义域边缘处), 此时我们就结束搜索。

爬山法的问题出在哪?

目标函数



假如我们现在站在上帝视角来看这个图, 很明显我们要找的最大值在左侧的蓝色点处, 但是我们刚刚找到的是位于黄色点处的局部最大值点。

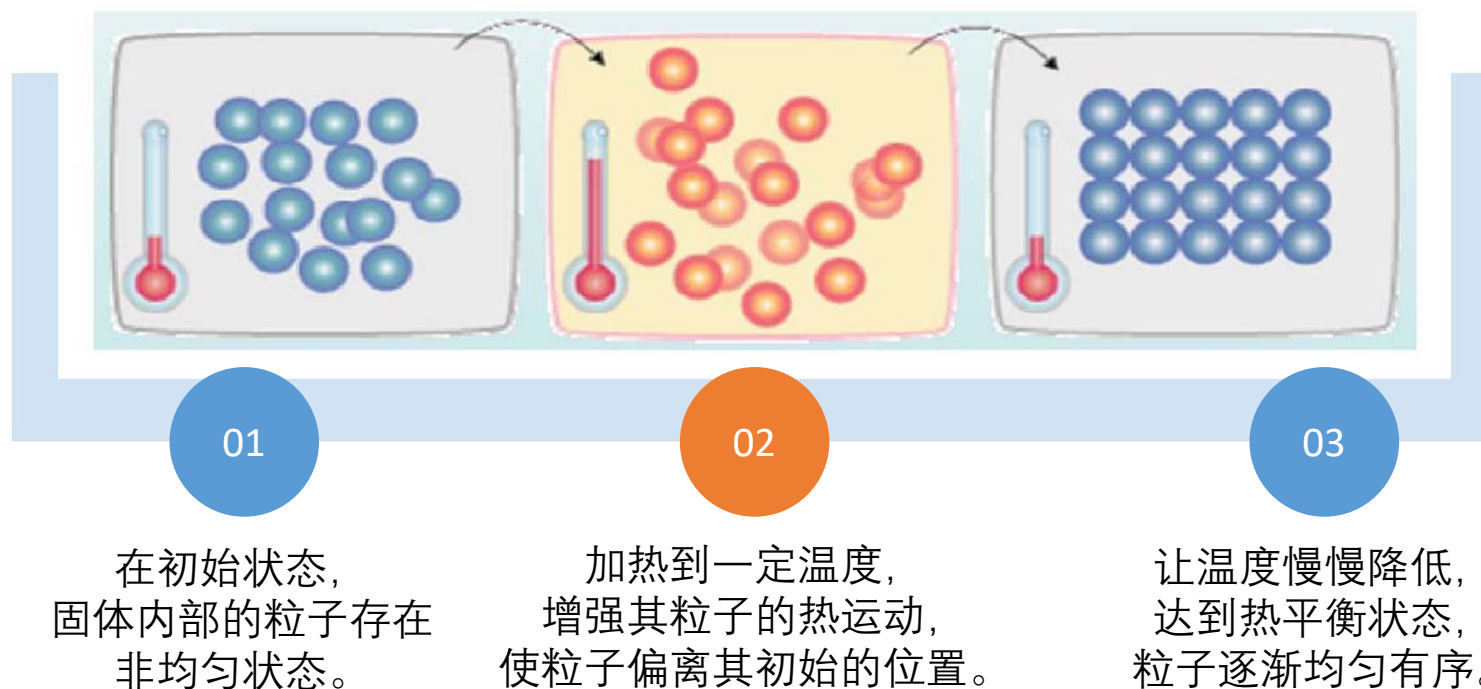
为什么爬山法的搜索在黄色点处就停止了昵? 是因为我们观察到该点左右两侧邻域的目标函数值都小于该点的目标函数值。若我们想要搜索到最大值处, 就还需要往左走一段, 因此爬山法实际上是一种“眼光狭隘”的搜索算法(贪心)。

我们马上要介绍的模拟退火算法就能克服这个问题。它在到达黄色点处后, 会以一定的概率接受一个比当前解要差的解 (即黄色点处左右两侧的邻域), 此时就有几率跳出黄色点, 从而我们就有了找到蓝色点的可能性。

模拟退火的介绍

模拟退火算法的思想最初是由美国物理学家Metropolis(梅特罗波利斯)在1953年提出的, 1983年, Kirkpatrick等将其应用于求解组合优化问题。

其出发点是基于物理中固体物质的退火过程与一般的优化问题之间的相似性。



- [1] Metropolis N. Equation of state calculations by fast computing machines[J]. Journal of chemical physics, 1953.
[2] Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by Simulated Annealing[J]. Science, 1983, 220.
[3] 图片来源: 大连大学模拟退火课件 <https://www.bilibili.com/video/av90608542>

从物理的视角看模拟退火

物理退火过程由以下三部分组成:

(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,固体将熔为液体,从而消除系统原先存在的非均匀状态。

(2) 等温过程。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的,当自由能达到最小时,系统达到平衡状态。

(3) 冷却过程。使粒子热运动减弱,系统能量下降,得到晶体结构。

其中,加温过程对应算法的设定初温,等温过程对应算法的 Metropolis 抽样过程,冷却过程对应控制参数的下降。这里能量的变化就是目标函数,要得到的最优解就是能量最低态。Metropolis 准则是 SA 算法收敛于全局最优解的关键所在, Metropolis 准则以一定的概率接受恶化解,这样就使算法跳离局部最优的陷阱。

Metropolis抽样过程以及Metropolis准则我们在后面的内容中会重点介绍。

注意: 没学过物理的同学如果看不懂这个物理过程也没关系, 我们的重点在于如何把这个模拟退火的思想应用在我们求解优化问题中。

从优化的视角看模拟退火

(复习的时候重点理解这一页)

模拟退火算法用于求解优化问题的步骤(以最小化为例):

- (1) 给定一个初始温度 T_0 , 并随机生成一个初始解 x_0 , 并计算相应的目标函数值 $f(x_0)$;
- (2) 令当前温度 T 等于冷却进度表中的下一个值 T_i ; (第一次迭代时 $T = T_0$)
- (3) 在当前解 x_i 的附近随机产生一个新解 x_j , 计算新解的目标函数值 $f(x_j)$; (第一次迭代时 $x_i = x_0$)
- (4) 如果 $f(x_j) < f(x_i)$, 则接受新解 x_j ; 如果 $f(x_j) > f(x_i)$, 则计算 $\Delta f = f(x_j) - f(x_i)$, 并计算 $p = e^{-\Delta f/T_i}$, 然后随机生成一个在区间 $[0,1]$ 上服从均匀分布的随机数 r , 如果 $r < p$, 则接受新解 x_j ;
- (5) 在温度 T_i 下, 将步骤(3)和(4)重复 L_i 次;
- (6) 判断是否满足退出条件, 如果满足则退出迭代, 否则回到步骤(2)继续迭代。

冷却进度表: 我们在后面介绍

Metropolis抽样过程: 我们上面步骤的(3)至(5)步

Metropolis准则: 我们上面步骤的第(4)步, 后面我们详细介绍

上面的步骤包含了两个循环, 你看得出来吗?

 数学建模学习交流

Metropolis 准则

[了解即可](#)

(4) 如果 $f(x_j) < f(x_i)$, 则接受新解 x_j ; 如果 $f(x_j) > f(x_i)$, 则计算 $\Delta f = f(x_j) - f(x_i)$, 并计算 $p = e^{-\Delta f/T_i}$, 然后随机生成一个在区间 $[0,1]$ 上服从均匀分布的随机数 r , 如果 $r < p$, 则接受新解 x_j ;


问题: 这个 p 是什么? 为什么要这么构造?

现代的模拟退火算法形成于 20 世纪 80 年代初, 其思想源于固体的退火过程, 即将固体加热至足够高的温度, 再缓慢冷却; 升温时, 固体内部粒子随温度升高变为无序状, 内能增大, 而缓慢冷却时粒子又逐渐趋于有序, 从理论上讲, 如果冷却过程足够缓慢, 那么冷却中任一温度时固体都能达到热平衡, 而冷却到低温时将达到这一低温下的内能最小状态。

在这一过程中, 任一恒定温度都能达到热平衡是个重要步骤, 这一点可以用 Monte Carlo 算法模拟, 不过其需要大量采样, 工作量很大。但因为物理系统总是趋向于能量最低, 而分子热运动则趋向于破坏这种低能量的状态, 故而只需着重取贡献比较大的状态即可达到比较好的效果, 因而 1953 年 Metropolis 提出了这样一个重要性采样的方法, 即设从当前状态 i 生成新状态 j , 若新状态的内能小于状态 i 的内能 (即 $E_j < E_i$), 则接受新状态 j 作为新的当前状态; 否则, 以概率 $\exp\left[\frac{-(E_j - E_i)}{kT}\right]$ 接受状态 j , 其中 k 为 Boltzmann 常数, 这就是通常所说的 Metropolis 准则。

模拟退火算法借用了 Metropolis 这篇论文的思想, 将内能改变为目标函数值, 同时去掉了分母上的 k , 最终构成了我们现在看到的公式。

参考: 卓金武等 《MATLAB在数学建模中的应用》, 北京航空航天大学出版社, 2011, P83

 数学建模学习交流

更多关于 $p = e^{-\Delta f/T_i}$ 的介绍

了解即可

将模拟退火搜索最优解的过程视为随机过程中的马尔科夫过程, 这里的 p 就视为状态转移概率, 即由一个状态(一个候选解)向另一个状态(另一个候选解)的转移概率, 那么在数学上能够证明: 在一定条件下, 当温度降为0时, 最后的状态一定是全局最优解。这样就能够为模拟退火算法在求解最优化问题提供理论支持。

当温度 T 降为 0 时, x_i 的分布为

$$P_i^* = \begin{cases} \frac{1}{|S_{\min}|}, & x_i \in S_{\min}, \\ 0, & \text{其他}, \end{cases}$$

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。

注意:

- (1) 通俗的理解这里的概率为: 接受一个新解为当前解的概率;
- (2) 这里的概率只有这样取才能够收敛到最优解吗? 不一定, 感兴趣的同学可以看康立山等1994年编写的《非数值并行算法 (第一册) 模拟退火算法》。

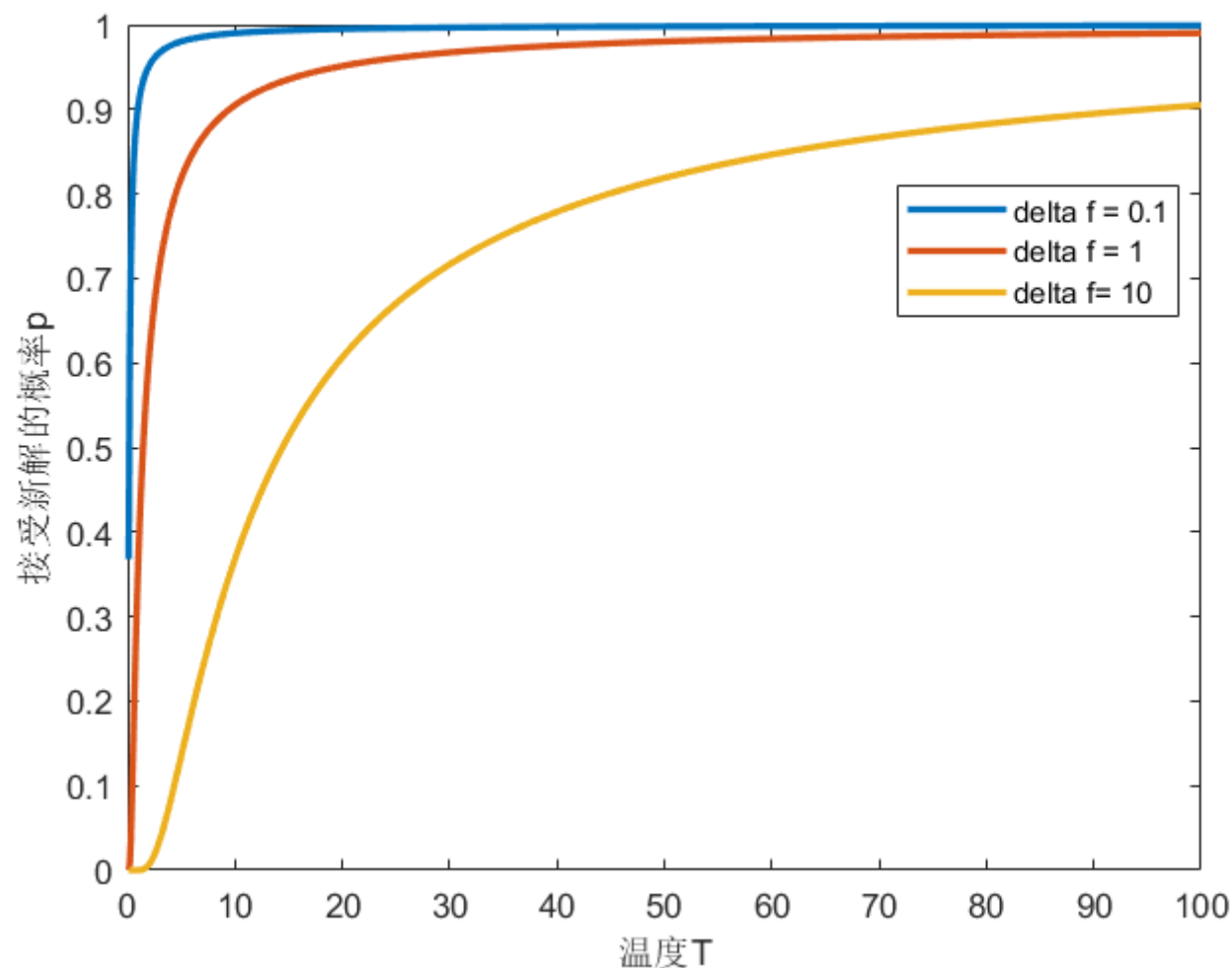
参考: 司守奎等《数学建模算法与应用 (第2版) 》, 国防工业出版社, 2015, P325

 数学建模学习交流

接受新解的概率 p 到底有多大?

当 $\Delta f = f(x_j) - f(x_i) > 0$ 时,
接受新解的概率 $p = e^{-\Delta f/T_i}$

code0.m



- (1) 温度一定时, Δf 越小, 概率越大, 即目标函数相差越小接受的可能性越大。
- (2) Δf 一定时, 温度越高, 概率越大, 即搜索前期温度较高时更有可能接受新解。

冷却进度表

退火过程由一组初始参数,即冷却进度表(cooling schedule)控制,它的核心是尽量使系统达到准平衡,以使算法在有限的时间内逼近最优解。冷却进度表包括:

- (1) 控制参数的初值 T_0 :冷却开始的温度。
- (2) 控制参数 T 的衰减函数:因计算机能够处理的都是离散数据,因此需要把连续的降温过程离散化成降温过程中的一系列温度点,衰减函数即计算这一系列温度的表达式。
- (3) 控制参数 T 的终值 T_f (停止准则)。
- (4) Markov 链的长度 L_k :任一温度 T 的迭代次数。

注意: (参数的设置很关键, 模拟退火难就难在参数的选取上)

- (1) 初始温度 T_0 的选择我们后面再来详细介绍, 最常取100。
- (2) 衰减函数最常用的是 $T_{k+1} = \alpha \times T_k$, α 一般是0.5-0.99间的常数。
- (3) 这里说的停止准则是指当温度下降到一个特别小的数时就退出搜索, 即退出外循环。在后面我们会介绍其他退出外循环的规则。
- (4) 这里提到了Markov链的长度 L_k , 这个是随机过程中马尔科夫链里面的概念, 它和我们前面步骤(5): “在温度 T_i 下, 将步骤(3)和(4)重复 L_i 次;” 是一个意思, 即在当前温度下需要内循环的次数。从经验来说, 对于简单的情况可以令 $L_k = 100n$, 其中 n 为自变量的个数。

参考: 卓金武等 《MATLAB在数学建模中的应用》, 北京航空航天大学出版社, 2011, P84

如何设置模拟退火的温度

(1) 设置模拟退火温度的初值 T_0

求解全局优化问题的随机搜索算法一般都采用大范围的粗略搜索与局部的精细搜索相结合的搜索策略。只有在初始的大范围搜索阶段找到全局最优解所在的区域,才能逐渐缩小搜索的范围,最终求出全局最优解。模拟退火算法是通过控制退火温度的初值 T_0 和其衰减变化过程来实现大范围的粗略搜索与局部的精细搜索。一般来说,只有足够大的 T_0 才能满足算法要求(但对不同的问题“足够大”的含义也不同,有的可能 $T_0 = 100$ 就可以,有的则要1000。在问题规模较大时,过小的 T_0 往往导致算法难以跳出局部陷阱而达不到全局最优。但为了减少计算量, T_0 不宜取得过大,而应与其他参数折中选取。

(2)控制温度的衰减函数

衰减函数可以有多种形式,一个常用的衰减函数是

$$T_{k+1} = \alpha T_k, k = 0, 1, 2, \dots$$

其中, α 是一个常数,可以取为0.5-0.99,它的取值决定了降温的过程。 α 较大会放慢温度衰减的速度,可能导致算法进程迭代次数的增加,从而使算法进程接受更多的变换,访问更多的邻域,搜索更大范围的解空间,返回更好的最终解,但求解的时间会有很大的增加,目前使用较多的 α 一般为0.95。

可以多次尝试不同的参数,观察求解时间和求解结果。

新解的产生规则

(3) 在当前解 x_i 的附近随机产生一个新解 x_j , 计算新解的目标函数值 $f(x_j)$;
(第一次迭代时 $x_i = x_0$)

我们以 n 元函数求最值为例, 来介绍新解的产生规则:

假设当前解为: (x_1, x_2, \dots, x_n)

我们首先生成一组随机数 (y_1, y_2, \dots, y_n) , 其中 y_i 服从 $N(0, 1)$

接下来我们再计算 (z_1, z_2, \dots, z_n) , 其中 $z_i = y_i / \sqrt{y_1^2 + y_2^2 + \dots + y_n^2}$,

对于每一个 $i \in [1, n]$, 进行下面的操作:

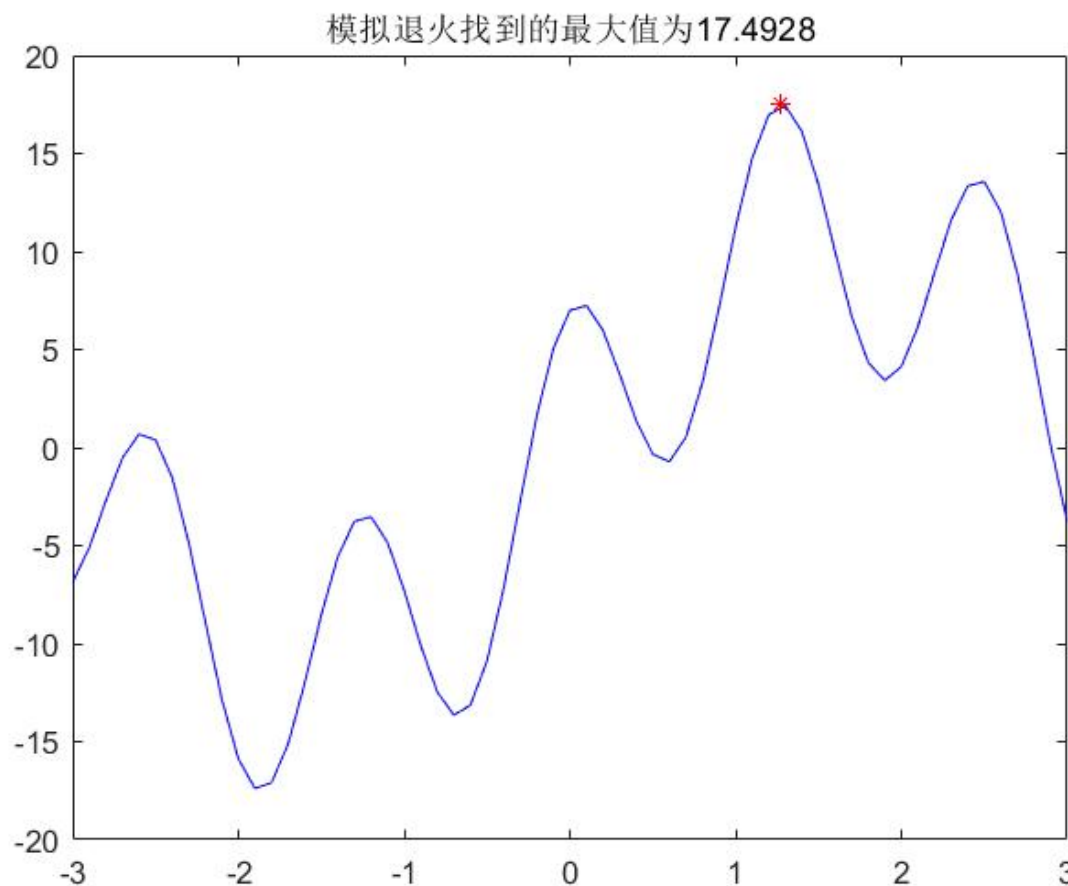
- (1) 计算: $x_i^{new} = x_i + T \times z_i$, T 是当前的温度(也可以使用 $x_i^{new} = x_i + \sqrt{T} \times z_i$ 这个公式)
- (2) 接下来检查 x_i^{new} 是否位于上下界内, 即是否满足 $lb_i \leq x_i^{new} \leq ub_i$ 这个约束
 - (a) 如果 $lb_i \leq x_i^{new} \leq ub_i$ 满足, 则直接令 $x_j = x_i^{new}$ 即可
 - (b) 如果 $x_i^{new} < lb_i$, 则 $x_j = r \times lb_i + (1 - r) \times x_i$, 这里 r 是区间 $[0, 1]$ 上均匀分布的随机数
 - (c) 如果 $x_i^{new} > ub_i$, 则 $x_j = r \times ub_i + (1 - r) \times x_i$, 这里 r 是区间 $[0, 1]$ 上均匀分布的随机数

注意: 新解的产生规则在学术上没有统一的规定, 我这里参考的是Matlab内置函数的那种定义方法。

模拟退火求解一元函数最大值

code1.m

求函数 $y = 11\sin x + 7\cos(5x)$ 在 $[-3, 3]$ 内的最大值

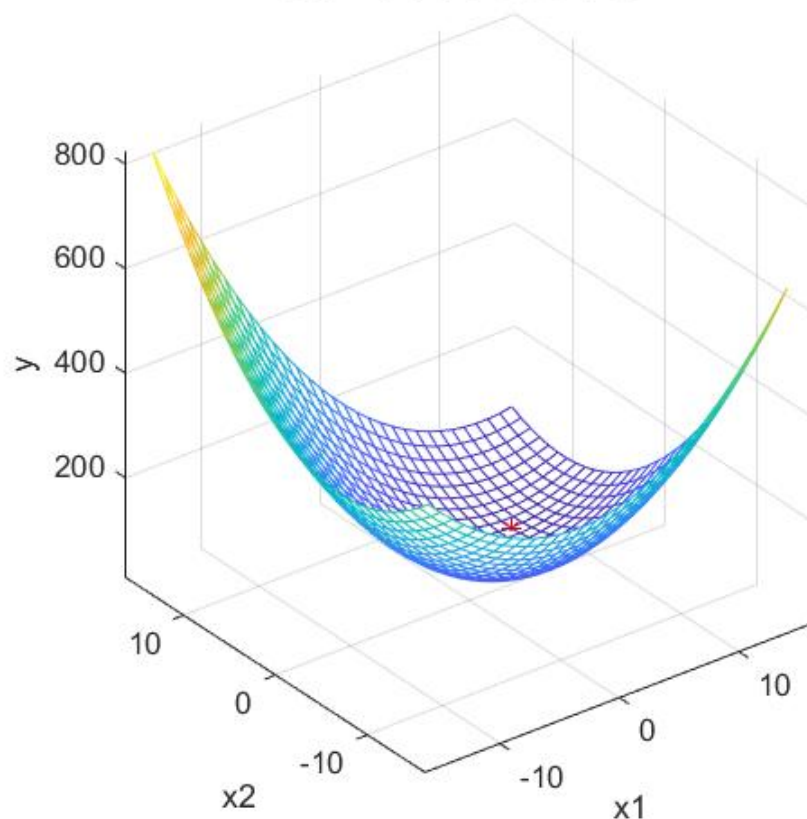


模拟退火求解二元函数最小值

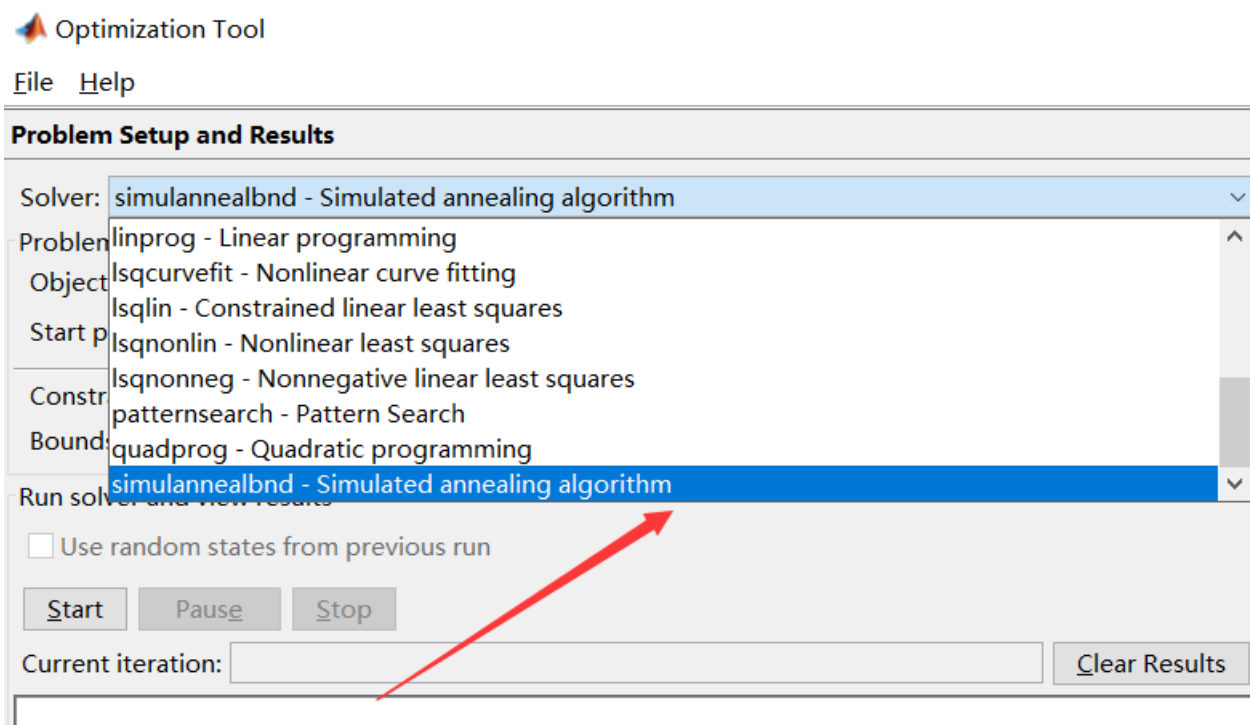
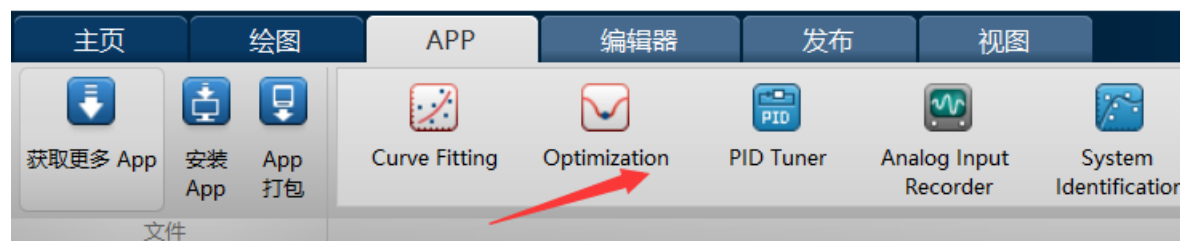
code2.m

求函数 $y = x_1^2 + x_2^2 - x_1x_2 - 10x_1 - 4x_2 + 60$ 在 $x_1, x_2 \in [-15, 15]$ 内的最小值

模拟退火找到的最小值为8

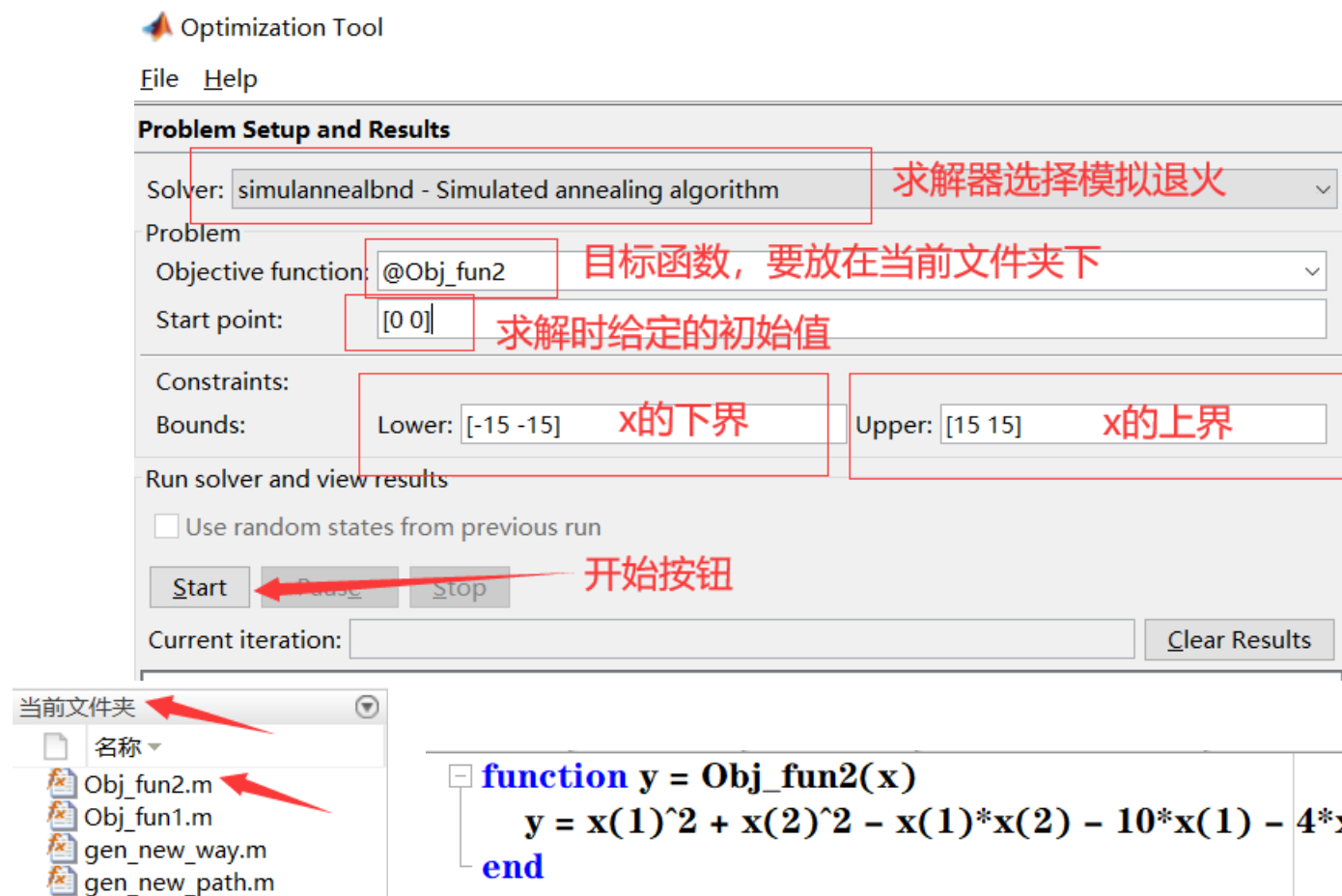


Matlab自带的模拟退火函数



这个工具箱是最开始安装Matlab可选的, 如果没有安装的话就需要重新下载Matlab

设置目标函数



Optimization Tool

File Help

Problem Setup and Results

Solver: simulannealbnd - Simulated annealing algorithm 求解器选择模拟退火

Problem

Objective function: @Obj_fun2 目标函数, 要放在当前文件夹下

Start point: [0 0] 求解时给定的初始值

Constraints:

Bounds: Lower: [-15 -15] x的下界 Upper: [15 15] x的上界

Run solver and view results

☐ Use random states from previous run

Start Pause Stop 开始按钮

Current iteration: Clear Results

当前文件夹

- 名称
- Obj_fun2.m
- Obj_fun1.m
- gen_new_way.m
- gen_new_path.m

```
function y = Obj_fun2(x)
    y = x(1)^2 + x(2)^2 - x(1)*x(2) - 10*x(1) - 4*x(2) + 60;
end
```

注意: 这里的目标函数是求最小值, 如果求最大值则需要添加负号转换为求最小值。
另外, 这个目标函数要保存在m文件中, 且一定要放在Matlab当前文件夹下。

停止准则

Options

☒ Stopping criteria ← 模拟退火停止搜索的一些准则

Max iterations: ☒ Use default: Inf
☐ Specify:

Max function evaluations: ☒ Use default: 3000*numberOfVariables
☐ Specify:

Time limit: ☒ Use default: Inf
☐ Specify:

Function tolerance: ☒ Use default: 1e-6
☐ Specify:

Objective limit: ☒ Use default: -Inf
☐ Specify:

Stall iterations: ☒ Use default: 500*numberOfVariables
☐ Specify:

- **Max iterations** bounds the number of iterations the algorithm takes.
- **Max function evaluations** bounds the number of function evaluations the algorithm performs.
- **Time limit** bounds the number of seconds the algorithm runs.
- **Function tolerance** The algorithm stops if the average change in the objective function after **Stall iterations** iterations is below **Function tolerance**.
- **Objective limit** The algorithm stops if the objective function goes below **Objective limit**.
- **Stall iterations** The algorithm stops if the average change in the objective function after **Stall iterations** iterations is below **Function tolerance**.

优化工具箱中的function evaluation是怎么计算的?

在进行优化的时候, 用户可以指定目标函数和/或约束函数。每当这些函数被调用一次, 就算一个function evaluation。在一次iteration过程中, 往往会有若干中间步骤, 所以一次迭代会有多次function evaluation。所以这个参数不等同于迭代次数, 而往往大于迭代次数。

停止准则

Function tolerance: ☒ Use default: 1e-6 ☐ Specify:

Objective limit: ☒ Use default: -Inf ☐ Specify:

Stall iterations: ☒ Use default: 500*numberOfVariables ☐ Specify:

```
% Compute change in bestfval and individuals in last options.StallIterLimit
```

```
% iterations
```

```
funChange = Inf;
```

```
if solverData.iteration >= options.StallIterLimit
```

```
    bestfvals = solverData.bestfvals;
```

```
    funChange = sum(abs(diff(bestfvals)));
```

```
end
```

```
if funChange <= options.TolFun
```

```
    solverData.running = false;
```

```
    if isTrialFeasible(solverData.bestx(:), [], [], [], [], problem.lb(:), problem.ub(:), 0)
```

```
        solverData.message = sprintf('Optimization terminated: change in best function value less than options.FunctionTolerance.')
```

```
        solverData.exitflag = 1;
```

```
else
```

源代码: sacheckexit.m

从1000次后开始看这次迭代前1000次的最佳函数改变值的和。

Obj_fun2中有两个自变量, 因此Stall iterations=500*2=1000; 从第1000次迭代开始, 以后的每一次迭代前都要计算这次迭代的前1000次迭代时的最佳函数值的一阶差分的和(加绝对值, 因为后面的最佳值小于或等于前面的最佳值), 如果这个和小于Function tolerance, 就可以退出搜索了。

退火设置

☐ Annealing parameters

Annealing function: Fast annealing

Reannealing interval: ☒ Use default: 100
☐ Specify:

Temperature update function: Exponential temperature update

Initial temperature: ☒ Use default: 100
☐ Specify:

Annealing function(退火函数): 设置新解的产生规则, 前面课件我们实际上介绍过了

Annealing function:

Fast annealing
Fast annealing
Boltzmann annealing
Custom

Reannealing interval:

Fast annealing: $x_i^{new} = x_i + T \times z_i$

Boltzmann annealing: $x_i^{new} = x_i + \sqrt{T} \times z_i$

Custom: 用户自己来设置新解的产生规则的函数

退火设置

Ingber, L. Adaptive simulated annealing (ASA): Lessons learned. Invited paper to a special issue of the Polish Journal Control and Cybernetics on "Simulated Annealing Applied to Combinatorial Optimization." 1995.

Annealing parameters

Annealing function:

Fast annealing

Reannealing interval:

☒ Use default: 100
☐ Specify:

Temperature update function:

Exponential temperature update

Initial temperature:

☒ Use default: 100
☐ Specify:

`simulannealbnd` reanneals after it accepts `ReannealInterval` points. Reannealing sets the annealing parameters to lower values than the iteration number, thus raising the temperature in each dimension. The annealing parameters depend on the values of estimated gradients of the objective function in each dimension. The basic formula is

$$k_i = \log \left(\frac{T_0 \max_j(s_j)}{T_i s_i} \right),$$

where

k_i = annealing parameter for component i .

T_0 = initial temperature of component i .

T_i = current temperature of component i .

s_i = gradient of objective in direction i times difference of bounds in direction i .

`simulannealbnd` safeguards the annealing parameter values against Inf and other improper values.

步骤参考: Matlab官网关于模拟退火算法的介绍

Matlab模拟退火的函数和我们之前自己写的函数不同, 它使用的是自适应模拟退火算法, 详情可见上面的参考文献, 主要有以下几点区别:

(1) 我们写的模拟退火函数中, 温度会随着迭代次数增加一直下降, 而自适应模拟退火算法会在搜索过程中根据解的状态再升温, 来提高搜索能力, 以免陷入局部最优解。

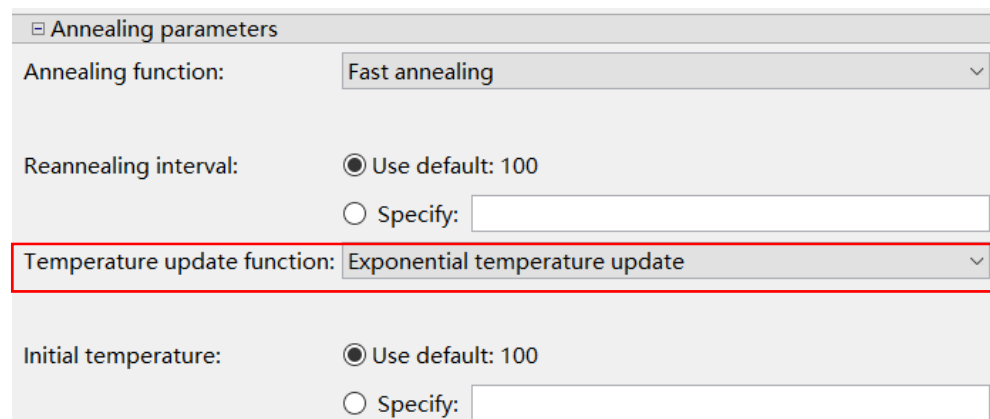
(2) 在每次迭代过程中, Matlab模拟退火的函数都只找一个新解, 即设置Markov链的长度等于1, 这样能节省搜索的时间。

(3) Matlab模拟退火的函数使用的Metropolis准则和我们使用的不同, 即接受新解的概率 p 不相同, 后面我们会再来介绍。

(4) 左边公式有一个退火参数 k , 这个参数在更新温度时有用, 后面我们会讲。

退火设置

步骤参考: Matlab官网关于模拟退火算法的介绍



Annealing parameters

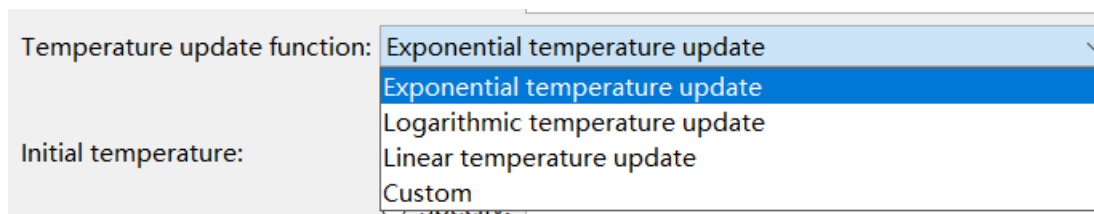
Annealing function: Fast annealing

Reannealing interval: ☒ Use default: 100
☐ Specify:

Temperature update function: Exponential temperature update

Initial temperature: ☒ Use default: 100
☐ Specify:

Temperature update function(温度更新函数): 设置温度衰减函数



Temperature update function: Exponential temperature update

Initial temperature:

前面三个温度更新函数的Matlab源码分别是:

`temperature = options.InitialTemperature.*.95.^optimValues.k;`

`temperature = options.InitialTemperature./log(optimValues.k);`

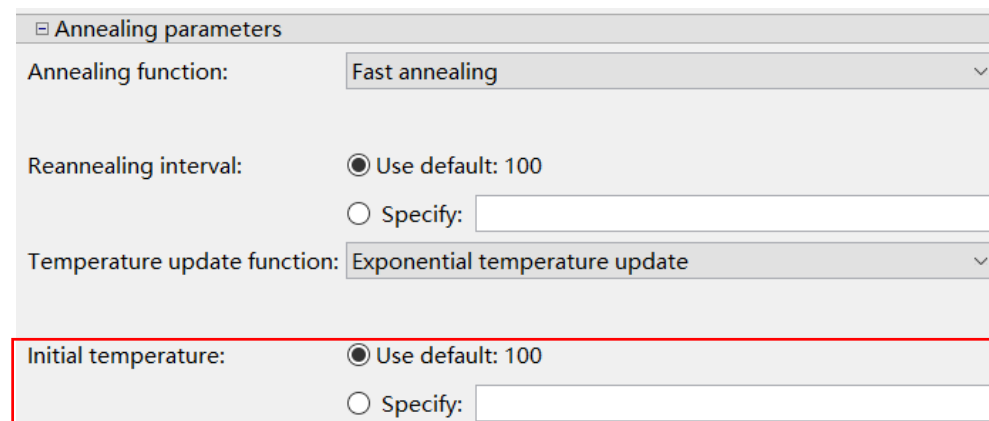
`temperature = options.InitialTemperature./optimValues.k;`

注: 这个`optimValues.k`就是上一页中的退火参数`k`, 直到“再退火”前, `k`就是迭代次数

The TemperatureFcn option specifies the function the algorithm uses to update the temperature. Let k denote the annealing parameter. (The annealing parameter is the same as the iteration number until reannealing.)

退火设置

步骤参考: Matlab官网关于模拟退火算法的介绍



The image shows the 'Annealing parameters' dialog box in Matlab. It contains the following settings:

- Annealing function: Fast annealing
- Reannealing interval: ☒ Use default: 100, ☐ Specify: []
- Temperature update function: Exponential temperature update
- Initial temperature: ☒ Use default: 100, ☐ Specify: []

The 'Initial temperature' section is highlighted with a red rectangle.

Initial temperature(初始温度): 默认每个维度均为100

Specifying initial temperature

The default initial temperature is set to 100 for each dimension. If you want the initial temperature to be different in different dimensions then you must specify a vector of temperatures. This may be necessary in cases when problem is scaled differently in each dimensions.

我们自己写的模拟退火中, 只设置了一个温度, 即目标函数中每一维自变量的温度都是相同的。

Matlab模拟退火的函数中, 我们可以为每一维自变量分别设置初始温度, 并且在“再退火”后, 每一维度的温度会自适应变化。

思考: 什么情况下设置不同的温度比较合适?

提示: 看看新解的产生规则, 如果上下界差异特别大, 就设置较高温。

接受准则

步骤参考: Matlab官网关于模拟退火算法的介绍

Acceptance criteria

Acceptance probability function: Simulated annealing acceptance

The algorithm determines whether the new point is better or worse than the current point. If the new point is better than the current point, it becomes the next point. If the new point is worse than the current point, the algorithm can still make it the next point. The algorithm accepts a worse point based on an acceptance function. Choose the acceptance function with the `AcceptanceFcn` option. Choices:

- `@acceptancesa` (default) — Simulated annealing acceptance function. The probability of acceptance is

$$\frac{1}{1 + \exp\left(\frac{\Delta}{\max(T)}\right)},$$

where

Δ = new objective – old objective.

T_0 = initial temperature of component i

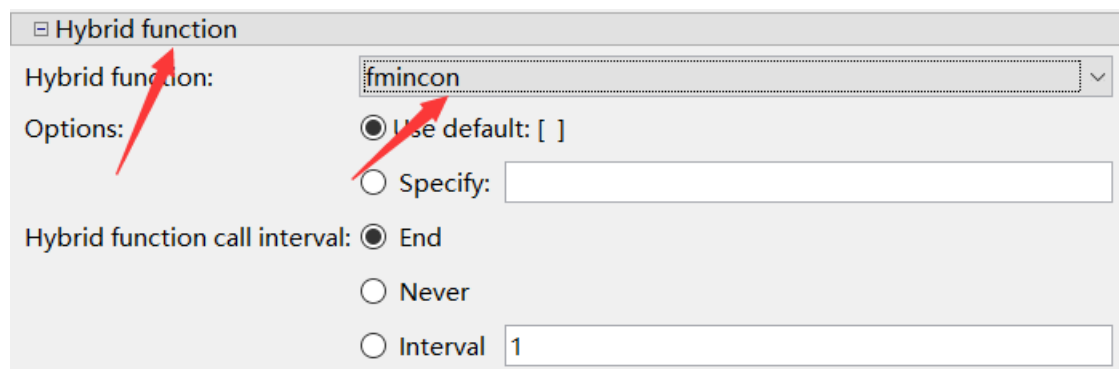
T = the current temperature.

Since both Δ and T are positive, the probability of acceptance is between 0 and 1/2. Smaller temperature leads to smaller acceptance probability. Also, larger Δ leads to smaller acceptance probability.

$$\text{原来的: } p = e^{-\Delta f/T} = \frac{1}{\exp(\frac{\Delta f}{T})}$$

后续使用其他函数求解

步骤参考: Matlab官网关于模拟退火算法的介绍



Hybrid function

Hybrid function: fmincon

Options: ☒ Use default: [] ☐ Specify:

Hybrid function call interval: ☒ End ☐ Never ☐ Interval 1

Hybrid function(混合函数): 将模拟退火的结果作为初始值, 调用其他函数求解

注: 这里一共提供了四种函数, 我们之前学过fmincon函数, 因此这里我们使用fmincon函数即可。

Hybrid function call interval:

- **End** — Run the hybrid function once, after simulannealbnd finishes.
(模拟退火全部运行完成后再使用混合函数)
- **Never** — Do not run the hybrid function. (不使用混合函数)
- **Interval** — Run the hybrid function after every **Interval** iterations of simulannealbnd. (可以指定在迭代多少次后使用混合函数)

模拟退火过程可视化

Plot functions

Plot interval:

☒ Best function value ☒ Best point ☒ Stopping criteria

☒ Temperature plot ☒ Current point ☒ Current function value

☐ Custom function:

Plot interval: 每迭代多少次更新一次图形。

Best function value: 每次迭代结束后找到的目标函数最小值。

Best point plots: 每次迭代结束后目标函数最小值对应的x的值。

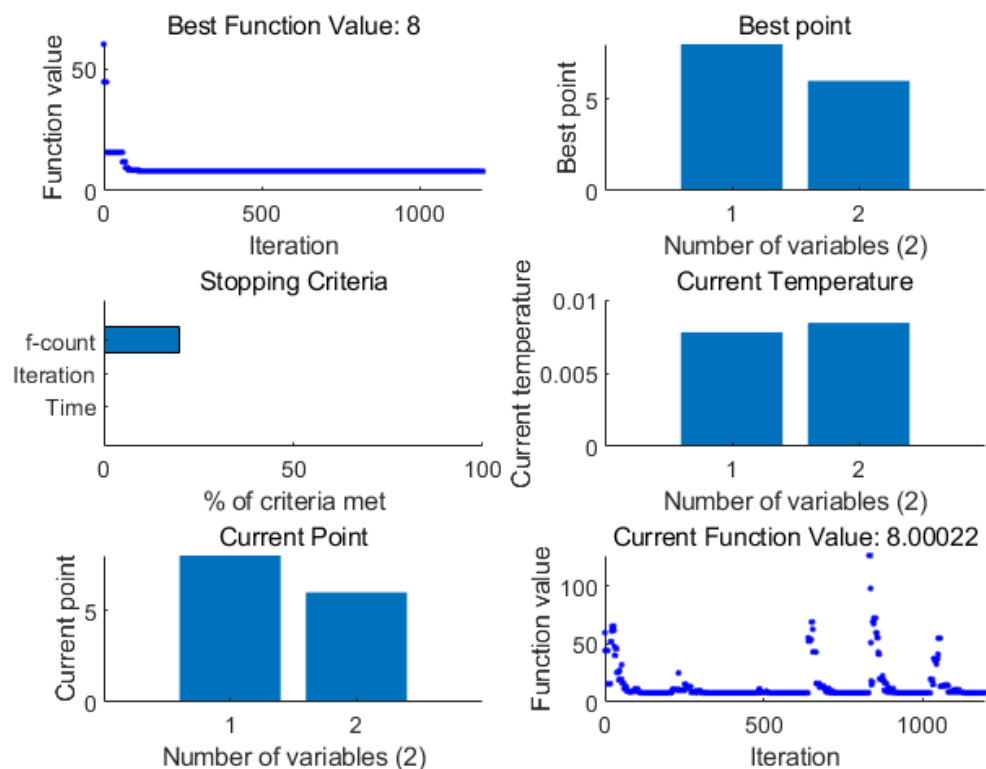
Stopping criteria: 绘制函数的终止条件的变化。

Temperature plot: 绘制温度的变化。

Current point: 每次迭代结束后对应的x的值。

Current function value: 每次迭代结束后对应的目标函数。

Custom function: 绘制自己定义的函数图形。



展示迭代的过程

☐ Display to command window

Level of display: iterative

Display interval: ☒ Use default: 10

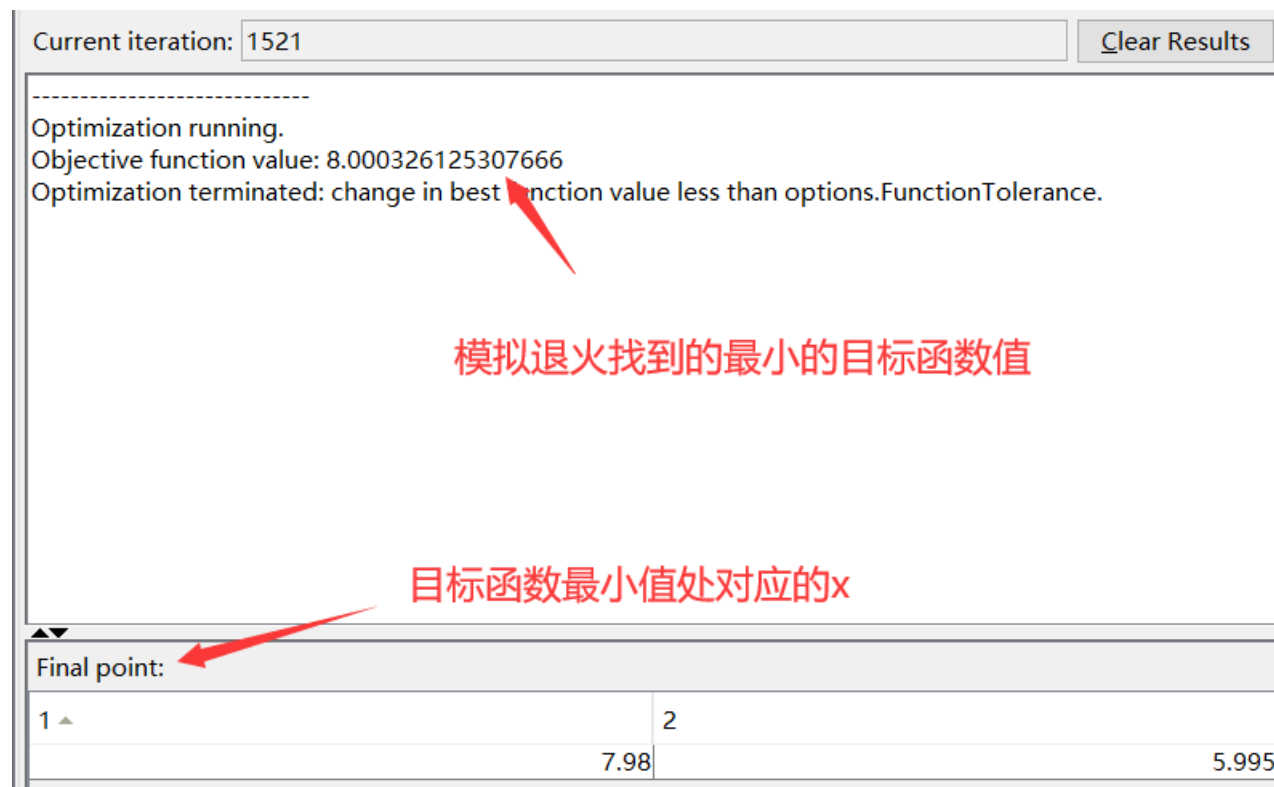
☐ Specify:

		Best	Current	Mean
Iteration	f-count	f(x)	f(x)	temperature
0	1	60	60	100
10	11	30.7205	30.7205	56.88
20	21	8.17976	8.17976	34.0562
30	31	8.17976	25.6691	20.3907
40	41	8.17976	25.6691	12.2087
50	51	8.17976	13.9364	7.30977
60	61	8.17976	8.63374	4.37663
70	71	8.17976	8.63374	2.62045
80	81	8.16835	8.16835	1.56896
90	91	8.16835	10.8991	0.939395
100	101	8.16835	8.65444	0.56245
110	111	8.16835	8.72807	0.33676
120	121	8.16835	9.08261	0.201631

Display interval: 每迭代多少次输出一次迭代的中间结果

注意：我发现了一个问题，不知道是不是版本原因，当我将展示迭代的过程和混合函数求解一起用时，工具箱会提示错误：“此类型的变量不支持使用点进行索引。”，因此这两个别同时使用。

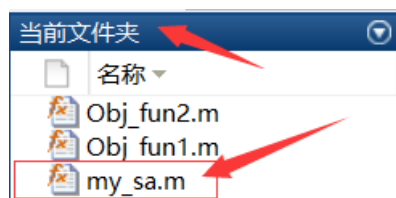
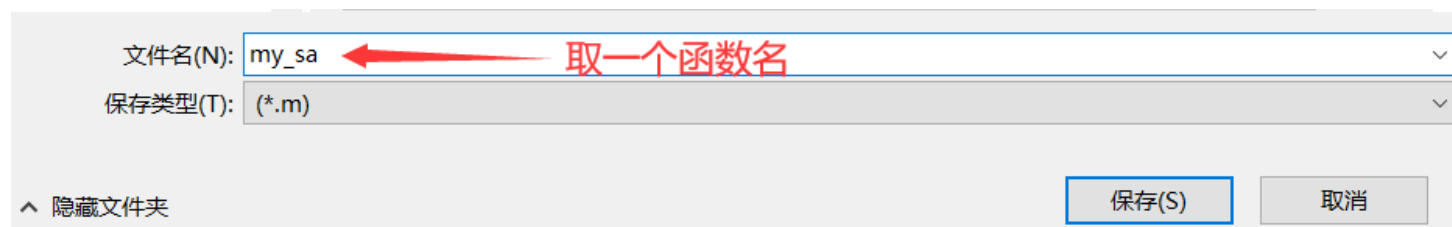
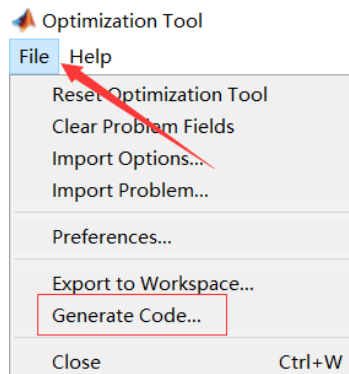
计算的结果



思考: 同一组设置, 多次运行得到的结果不同正常吗?

答案: 当然正常, 模拟退火里面搜索的时候用到了随机数, 例如在新解的产生规则中, 如果运气好我们生成的新解刚好在最优解附近。因此每次运行的结果可能会有差异。

生成可以调用的函数代码



```
function [x,fval,exitflag,output] = my_sa(x0,lb,ub)
%% This is an auto generated MATLAB file from Optimization Tool.

%% Start with the default options
options = optimoptions('simulannealbnd');
%% Modify options setting
options = optimoptions(options,'Display', 'off');
options = optimoptions(options,'HybridInterval', 'end');
[x,fval,exitflag,output] = ...
simulannealbnd(@Obj_fun2,x0,lb,ub,options);
```

命令行窗口

```
fx >> [x,fval,exitflag,output] = my_sa([0 0],[-15 -15],[15 15])
```

查看详细结果

x =

7.9993 5.9998

fval =

8.0000

exitflag =

1

Exit Flag	Meaning
1	Average change in the value of the objective function over options.MaxStallIterations iterations is less than options.FunctionTolerance.
5	Objective function value is less than options.ObjectiveLimit.
0	Maximum number of function evaluations or iterations reached.
-1	Optimization terminated by an output function or plot function.
-2	No feasible point found.
-5	Time limit exceeded.

output =

包含以下字段的 **struct**:

iterations: 1187

funccount: 1198

message: 'Optimization terminated: change in best function value less than options.FunctionTolerance.'

rngstate: [1 × 1 struct]

problemtype: 'boundconstraints'

temperature: [2 × 1 double]

totaltime: 0.1394

模拟退火解决旅行商问题

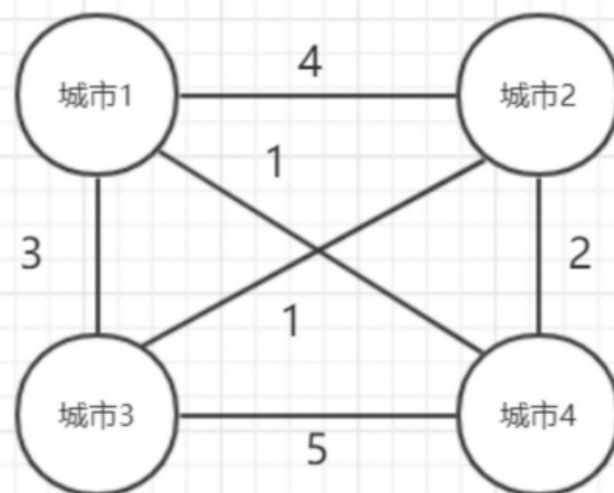
更新10: 蒙特卡罗模拟

(6) 旅行商问题 (Traveling Salesman Problem, TSP)

一个售货员必须访问 n 个城市, 这 n 个城市是一个完全图, 售货员需要恰好访问所有城市的一次, 并且回到最终的城市。

城市于城市之间有一个旅行费用, 售货员希望旅行费用之和最少。

完全图: 完全图是一个简单的无向图, 其中每对不同的顶点之间都恰连有一条边相连。



最小费用 城市1->
城市3->城市2->城
市4->城市1

假设有 n 个城市, 则运算的开支为 $n!$

$$n=10 \rightarrow 10^6$$

$$n=15 \rightarrow 10^{12}$$

$$n=20 \rightarrow 10^{18}$$


$$(112 = 10^8)$$

有 $(n-1)!$ 种路径

每个路径要做
 n 个求和



未来学习智能算法时会解决 n 很大的情形。

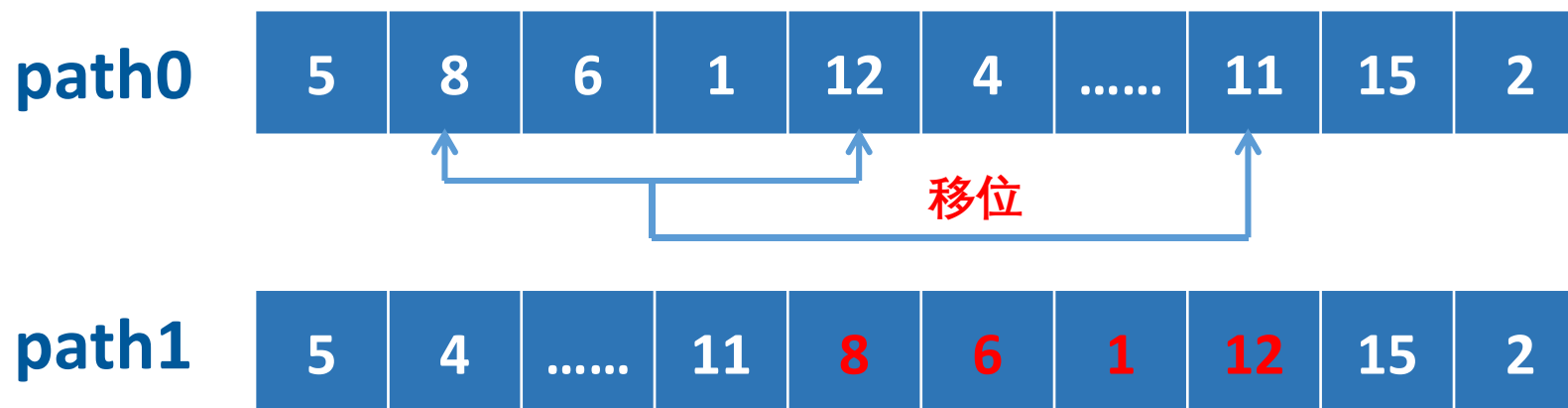
 数学建模学习交流

产生新解的方法

(1) 交换法：随机选择两个点，交换这两个点的位置



(2) 移位法：随机选择三个点，将前两个点之间的点移位到第三个点后



产生新解的方法

苗卉, 杨韬. 旅行商问题(TSP)的改进模拟退火算法[J]. 微计算机信息, 2007, 023(033):241-242,236.

(3) 倒置法: 随机选择两个点, 将这两个点之间的顺序完全颠倒

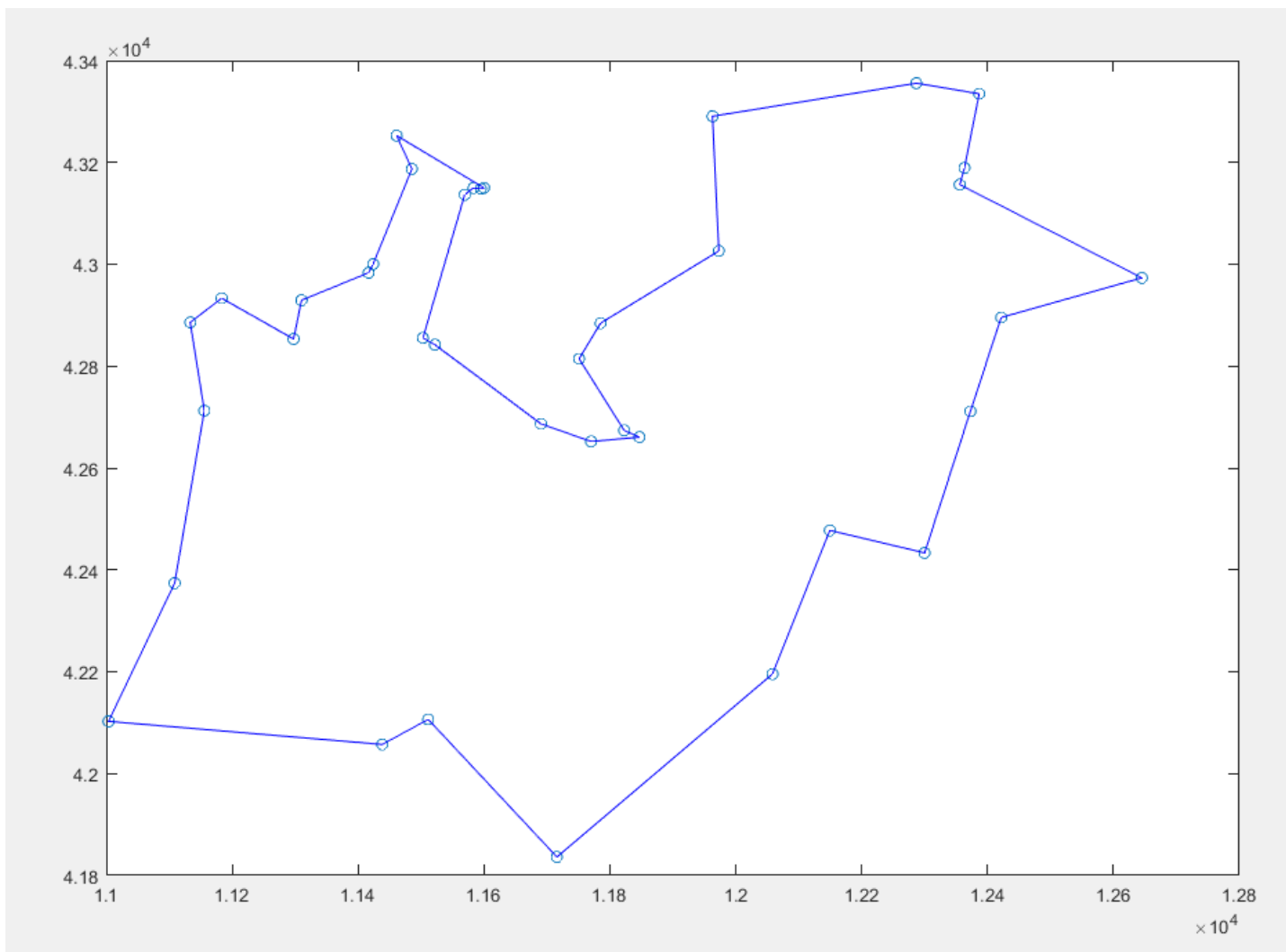


因为这三种产生新解的方法中都需要随机选点, 这样就有可能选择的点都在同一个位置的情况, 那么请问大家: 在编程中, 我们是否有必要对这种情况进行单独处理?

我个人觉得不必要, 如果要处理, 那么每次随机选点后我们都需要使用"if"语句进行判断, 这样会增加计算的时间。就算存在选点在同一个位置的情况, 我们也不用慌张, 一方面: 即使产生的新解和之前的解完全一样, 也不会对我们最终的结果照成干扰; 另一方面, 这种情况出现的概率较低, 特别是当n很大的时候, 且在每个温度下我们可以多次构造新解, 因此这种影响可以忽略。

38个城市TSP求解结果

code3.m



TSP数据集网站: <http://www.tsp.gatech.edu/world/djtour.html>

 数学建模学习交流

书店买书问题

更新10: 蒙特卡罗模拟

(4) 书店买书问题 (0-1规划)

某同学要从六家线上商城选购五本书籍 B1, B2, B3, B4, B5, 每本书籍在不同商家的售价以及每个商家的单次运费如下表所示, 请给该同学制定最省钱的选购方案。

(注: 在同一个店买多本书也只会收取一次运费)

	B1	B2	B3	B4	B5	运费
A 商城	18	39	29	48	59	10
B 商城	24	45	23	54	44	15
C 商城	22	45	23	53	53	15
D 商城	28	47	17	57	47	10
E 商城	24	42	24	47	59	10
F 商城	27	48	20	55	53	15

假设有 m 个书店, n 本书, 那么买书的方案数为 m^n , 下面我们就来用模拟退火来解决下面这个买书问题, 在Excel表格“[书店买书问题.xlsx](#)”中, 有20本书, 15个书店, 请确定一个购书方案可使花费最小。

15家店20本书的例题

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	运费
S1	31	31	41	21	25	28	23	34	38	29	38	33	32	24	23	20	23	26	21	32	10
S2	40	27	38	26	23	29	24	22	37	29	32	34	31	27	31	22	26	27	25	27	10
S3	35	25	41	20	26	21	37	24	34	22	42	31	37	26	28	23	23	21	26	28	14
S4	33	26	22	29	38	25	34	32	34	24	27	25	26	31	39	34	21	21	41	34	7
S5	33	29	36	24	21	24	33	28	25	29	24	26	26	29	37	24	25	25	32	27	12
S6	25	32	20	21	20	32	42	22	33	24	35	28	38	26	34	21	39	25	40	23	5
S7	35	22	35	29	29	26	38	30	27	21	25	30	33	32	30	32	25	23	26	23	10
S8	36	22	39	26	34	25	32	23	35	29	20	32	34	31	25	24	38	25	29	25	8
S9	32	23	22	21	27	22	20	30	27	24	41	27	33	27	29	22	31	26	25	24	14
S10	27	28	36	22	38	27	29	33	29	25	29	33	34	25	24	22	37	27	42	30	9
S11	39	28	26	27	37	28	23	31	35	27	30	28	20	32	31	21	32	31	43	21	12
S12	22	28	38	33	40	23	43	30	35	24	23	26	36	23	34	24	40	24	41	30	6
S13	30	25	27	32	27	30	40	27	36	22	30	29	21	32	41	33	33	29	31	31	11
S14	34	21	27	29	25	21	36	33	21	28	21	30	35	22	22	24	40	27	25	23	5
S15	31	27	24	25	39	23	40	30	22	28	38	31	21	29	21	25	40	22	31	35	9

way是长度为20的向量, 其中每一个位置的元素的取值都位于1-15之间。
含义是: 第i本书是在第way(i)家店购买的; 因此每一个way都是一个可能的解。

way0

8	11	8	10	5	1	1	8	1
---	----	---	----	---	---	-------	---	---	---

随机选择一本书, 改变购买它的书店

way1

8	11	8	10	7	1	1	8	1
---	----	---	----	---	---	-------	---	---	---

求解结果

code4.m

```
function way1 = gen_new_way(way0, s, b)
% way0: 原来的买书方案, 是一个1*b的向量, 每一个元素都位于1-s之间
index = randi([1, b],1); % 看哪一本书要更换书店购买
way1 = way0; % 将原来的方案赋值给way1
way1(index) = randi([1, s],1); % 将way1中的第index本书换一个书店购买
end
```

最佳的方案是:

1 至 19 列

6 14 6 6 6 14 11 6 14 4 14 4 11 14 14 6 4 4 14

20 列

11

此时最优值是:

466

因为没有文献参考, 我只想了这一种产生新解的方法, 要是有的同学使用其他产生新解的方法得到了更小的花费, 欢迎和我交流, (づ[〰]3[〰])づ。

更多关于模拟退火算法的讨论

(1) 模拟退火可以求解带有复杂约束条件的规划问题吗?

可以的, 和粒子群算法一样, 我们可以在目标函数上引入惩罚项来进行求解, 也可以在产生新解后判断新解是否满足约束条件, 若不满足则直接拒绝。

(2) 模拟退火的衍生算法有哪些?

加温退火法、有记忆的模拟退火、带返回搜索的模拟退火、多次寻优法、回火退火法、并行模拟退火算法等, 关于这些算法的具体介绍可参考康立山等1994年编写的《非数值并行算法(第一册)模拟退火算法》, 如果你不是专门研究模拟退火理论的, 这些方法的细节我们可以不需要知道。

(3) 模拟退火现在的研究还多吗, 主要是研究什么?

模拟退火的研究可分为两类, 一类基于马尔可夫链的有关理论, 探究不同条件下模拟退火求解的收敛性(即纯理论研究); 另一类针对具体问题, 给出了模拟退火的若干成功应用(即应用层面)。目前模拟退火相关的论文数量较十多年前有很大的下滑, 知网搜索可以发现, 智能算法中使用最多的是我们下一节要介绍的遗传算法。

(4) 站在数学建模比赛的角度, 模拟退火有什么用?

- (1) 求解无约束(或只有上下界约束)函数最值;
- (2) 求解简单的组合优化问题, 例如旅行商问题和书店买书问题;
- (3) 未来再遇到非线性整数规划问题时, 就可以想想能否用模拟退火求解。

注: 第(1)点粒子群算法也可以求解哦, 第(2)点和第(3)点我们下一节要介绍的遗传算法也可以求解, 另外对于常规的规划问题, 大家一定要优先考虑我们更新12里面介绍的各种求解函数, 智能算法只有在那些函数都无能为力时再想着使用。

课后作业

(1) 自己写一个模拟退火的代码, 求解下面这个函数的最大值; 此外, 再使用Matlab优化工具箱中的模拟退火函数, 来求解它的最大值。

$$y = 21.5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2), x_1 \in [-3, 12.1], x_2 \in [4.1, 5.8]$$

参考的最大值: 38.8503 提示: 这个函数有非常多的局部最大值, 怎么设置合适的退火参数需要不断尝试。

(2) 在书店买书问题中: 加入下面这个规则: “如果在同一个书店的消费额不低于88元, 那么这个书店包邮, 即在这个书店买书不用出运费了”, 请你修改代码对该问题重新求解。

这个问题并不难, 只需要修改计算花费的函数“calculate_money”, 当然你也可以改进产生新解的方式。
我求的最优解为439(●^v^●)
(模拟退火算法结果具有一定的随机性, 可以多次运行取最好的)

参考答案可在本节课件压缩包内找到, 尽量先独立思考后再看答案。

课后作业

(3) 在“更新12规划问题”这一讲中, 我们介绍了经典的背包问题, 请你使用模拟退火算法重新求解那个例子, 并和之前使用的intlinprog函数求解的结果对比, 看两种方法得到的结果是否有差异。本题的最优解是2410。

参考论文: 梁国宏, 张生, 黄辉, et al. 一种改进的模拟退火算法求解0-1背包问题[J]. 广西民族大学学报(自然科学版), 2007(03):97-99.

(4) 在经典的TSP问题中, 假设有 n 个城市, 我们要从某一个城市出发, 依次访问完所有城市后再回到起点处。现在, 我们探究一个类似的问题: 假设起点城市 M 和终点城市 N 是给定的, 要我们求出一个从 M 出发依次经过所有城市并最终到达 N 的最短路径, 即不再要求我们回到起点 M 处。请大家用模拟退火来求解这个问题。

提示: 可以把起点城市放在下标为1的位置, 将终点城市放在下标为 n 的位置。当起点为5且终点为20时, 我得到的最短路径为6636。