“智能优化原理与算法”课程
研究报告

报告名称：模拟退火算法及其改进和应用

姓名：尉前进

学号：1170400423

日期：2020.6.8

哈尔滨工业大学

1. 算法发展历程及基本原理

1.1 算法发展历程

著名的模拟退火算法,它是一种基于蒙特卡洛思想设计的近似求解最优化问题的方法。

美国物理学家 N.Metropolis 和同仁在 1953 年发表研究复杂系统、计算其中能量分布的文章，他们使用蒙特卡罗模拟法计算多分子系统中分子的能量分布。

美国 IBM 公司物理学家 S.Kirkpatrick、C. D. Gelatt 和 M. P. Vecchi 于 1983 年在《Science》上发表了一篇颇具影响力的文章：《以模拟退火法进行最优化（Optimization by Simulated Annealing）》。他们借用了 Metropolis 等人的方法探讨一种旋转玻璃态系统（spin glass system）时，发觉其物理系统的能量和一些组合最优（combinatorial optimization）问题（著名的旅行推销员问题 TSP 即是一个代表例子）的成本函数相当类似：寻求最低成本即似寻求最低能量[1]。由此，他们发展出以 Metropolis 方法为本的一套算法，并用其来解决组合问题等的寻求最优解[2]。

几乎同时，欧洲物理学家 V.Carny 也发表了几乎相同的成果，但两者是各自独立发现的；只是 Carny“运气不佳”，当时没什么人注意到他的大作；或许可以说，《Science》杂志行销全球，“曝光度”很高，素负盛名，而 Carny 却在另外一本发行量很小的专门学术期刊《J.Opt.Theory Appl.》发表其成果因而并未引起应有的关注。

Kirkpatrick 等人受到 Metropolis 等人用蒙特卡罗模拟的启发而发明了“模拟退火”这个名词，因为它和物体退火过程相类似。寻找问题的最优解（最值）即类似寻找系统的最低能量。因此系统降温时，能量也逐渐下降，而同样意义地，问题的解也“下降”到最值。

目前，模拟退火算法迎来了鼎盛时期，无论是理论研究还是应用研究都成了十分热门的课题[3-7]。尤其是他的应用研究显得格外活跃，已在工程中得到了广泛应用，诸如生产调度、控制工程、机器学习、神经网络、模式识别、图像处理、离散/连续变量的结构优化问题等领域。它能有效地求解常规优化方法难以解决的组合优化问题和复杂函数优化问题，实用范围极广。

1.2 算法基本原理

模拟退火算法来源于固体退火原理,将固体加温至充分高,再让其徐徐冷却。加温时，固体内部粒子随温升变为无序状，内能增大；而徐徐冷却时粒子渐趋有序，在每个温度上都达到平衡态，最后在常温时达到基态，内能减为最小。模拟退火算法与金属退火过程的相似关系如表 1.1 所示。根据 Metropolis 准则，粒子在温度 T 时趋于平衡的概率为 $\exp(-\Delta E/T)$，其中 E 为温度 T 时的内能，ΔE 为

其改变量。用固体退火模拟组合优化问题，将内能 E 模拟为目标函数值，温度T演化成控制参数，即得到解组合优化问题的模拟退火算法：由初始解X_0和控制参数初值T开始，对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步减小T值，算法终止时的当前解即为所得近似最优解，这是基于 Monte Carlo 迭代求解法的一种启发式随机搜索过程。退火过程由冷却进度表控制，包括控制参数的初值T_0及其衰减因子K、每个T值时的迭代次数L和停止条件。

表 1.1 模拟退火算法与金属退火过程的相似关系

物理退火	模拟退火
粒子状态	解
能量最低态	最优解
溶解过程	设定初温
等温过程	Metropolis 采样过程
冷却	控制参数的下降
能量	目标函数

1.3 模拟退火算法思想

模拟退火的主要思想是：在搜索区间随机游走（即随机选择点），再利用 Metropolis 抽样准则，使随机游走逐渐收敛于局部最优解。而温度是 Metropolis 算法中的一个重要控制参数，可以认为这个参数的大小控制了随机过程向局部或全局最优解移动的快慢。

Metropolis 是一种有效的重点抽样法，其算法为：系统从一个能量状态变化到另一个状态时，相应的能量从E_1变化到E_2，其概率为

$$p=\exp(-\frac{E_2-E_1}{T}) \tag{1.1}$$

如果$E_2<E_1$，系统接受此状态；否则，以一个随机的概率接受或丢弃此状态。状态 2 被接受的概率为

$$p(1\to 2)=\begin{cases}1 & E_2<E_1\\ \exp(-\frac{E_2-E_1}{T}) & E_2\geq E_1\end{cases} \tag{1.2}$$

这样经过一定次数的迭代，系统会逐渐趋于一个稳定的分布状态。

重点抽样时，新状态下如果向下，则接受（局部最优）；若向上（全局搜索），则以一定概率接受。模拟退火算法从某个初始解出发，经过大量解的变换后，可以求得给定控制参数值时组合优化问题的相对最优解。然后减小控制参数T的值，重复执行 Metropolis 算法，就可以在控制参数T趋于零时，最终求得组合

优化问题的整体最优解。控制参数的值必须缓慢衰减。

温度是一个Metropolis算法的重要控制参数，模拟退火可视为递减控制参数T时Metroplis算法的迭代。开始时T值大，可以接受较差的恶化解；随着T的减小，只能接受较好的恶化解；最后在T趋于0时，就不再接受任何恶化解了。

在无限高温时，系统立即均匀分布，接受所有提出的变换。T的衰减越小，T到达终点的时间越长；但可使Markov链减小，以使到达准平衡分布的时间变短。

2. 模拟退火算法流程及说明

2.1 模拟退火算法流程

模拟退火算法新解的产生和接受可分为如下三个步骤：

（1） 由一个产生函数从当前解X产生一个位于解空间的新解X'（其中$X'=X+Z$，Z为一个在$(0,1)$上均匀分布的随机数）；为便于后续的计算和接受，减少算法耗时，通常选择由当前解经过简单变换即可产生新解的方法。注意，产生新解的变换方法决定了当前新解的邻域结构，因而对冷却进度表的选取有一定的影响。

（2） 判断新解是否被接受，判断的依据是一个接受准则，最常用的接受准则是Metropolis准则：若$\Delta E<0$，则接受X'作为新的当前解X；否则，以概率$\exp(-\Delta E/T)$接受X'作为新的当前解X。

（3） 当新解被确定接受时，用新解代替当前解，这只需将当前解中对应于产生新解时的变换部分予以实现，同时修正目标函数值即可。此时，当前解实现了一次迭代，可在此基础上开始下一轮试验。若当新解被判定为舍弃，则在原当前解的基础上继续下一轮试验。

模拟退火算法求得的解与初始解状态（算法迭代的起点）无关，具有渐近收敛性，已在理论上被证明是一种以概率1收敛于全局最优解的优化算法。模拟退火算法可以分解为解空间、目标函数和初始解三部分。该算法具体流程如下[8]：

（1） 初始化：设置初始温度T_0（充分大）、初始解状态X_0（是算法迭代的起点）、每个T值的迭代次数L；

（2） 对$k=1,\cdots,L$做第（3）至第（6）步；

（3） 产生新解X'；

（4） 计算增量$\Delta E=E(X')-E(X)$，其中$E(X)$为评价函数；

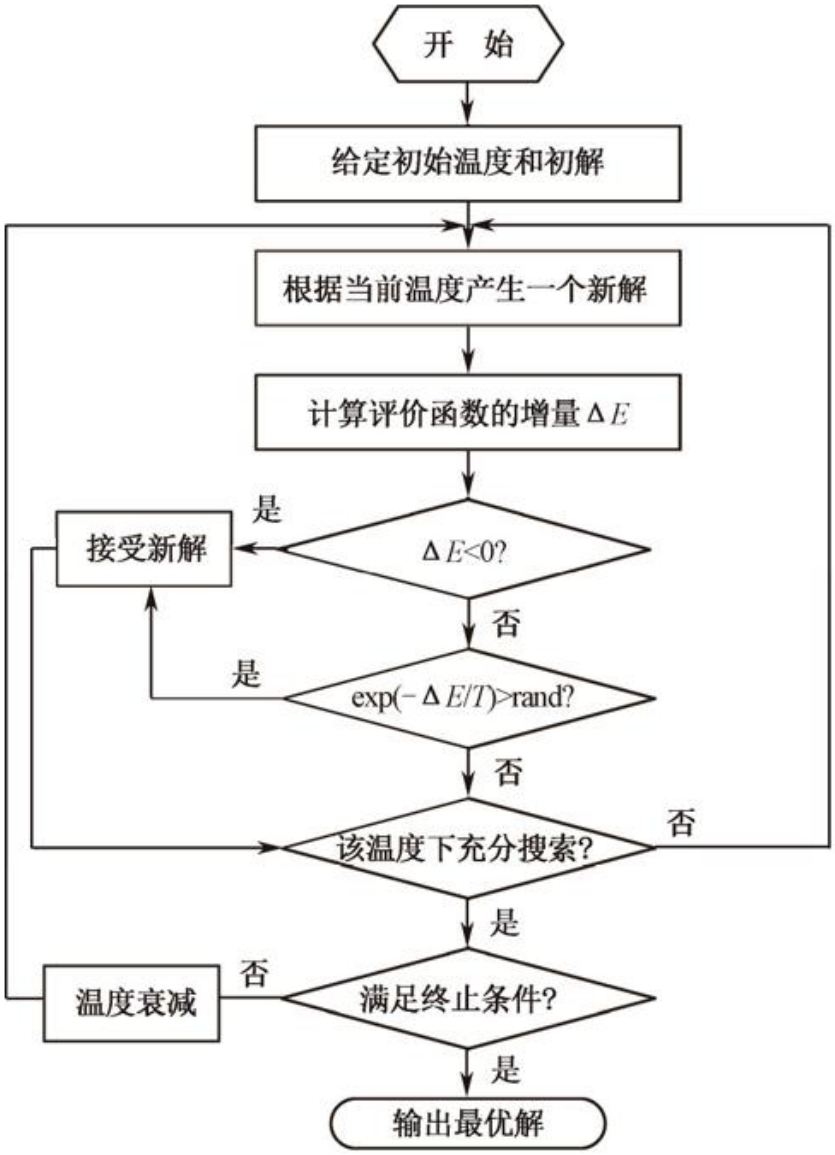
（5） 若$\Delta E<0$，则接受X'作为新的当前解，否则以概率$\exp(-\Delta E/T)$接受X'作为新的当前解；

（6） 如果满足终止条件，则输出当前解作为最优解，结束程序；

（7） T逐渐减小，且$T\to 0$，然后转第（2）步。

模拟退火算法流程如图 2.1 所示。

图 2.1



2.2 关键参数说明

模拟退火算法的性能质量高，它比较通用，而且容易实现。不过，为了得到最优解，该算法通常要求较高的初温以及足够多次的抽样，这使算法的优化时间往往过长。从算法结构知，新的状态产生函数、初温、退温函数、Markov 链长度和算法停止准则，是直接影响算法优化结果的主要环节。

状态产生函数

设计状态产生函数应该考虑到尽可能地保证所产生的候选解遍布全部解空间。一般情况下状态产生函数由两部分组成，即产生候选解的方式和产生候选解的概率分布。候选解的产生方式由问题的性质决定，通常在当前状态的邻域结构内以一定概率产生。

初温

温度T在算法中具有决定性的作用，它直接控制着退火的走向。由随机移动的接受准则可知：初温越大，获得高质量解的几率就越大，且Metropolis的接收率约为1。然而，初温过高会使计算时间增加。为此，可以均匀抽样一组状态，以各状态目标值的方差为初温。

退温函数

退温函数即温度更新函数，用于在外循环中修改温度值。目前，最常用的温度更新函数为指数退温，即$T(n+1)=K\times T(n)$，其中$0<K<1$是一个非常接近于1的常数。

Markov链长度L的选取

Markov链长度是在等温条件下进行迭代优化的次数，其选取原则是在衰减参数T的衰减函数已选定的前提下，L应选得在控制参数的每一取值上都能恢复准平衡，一般L取100～1000。

算法停止准则

算法停止准则用于决定算法何时结束。可以简单地设置温度终值T_f，当$T=T_f$时算法终止。然而，模拟火算法的收敛性理论中要求T_f趋向于零，这其实是不实际的。常用的停止准则包括：设置终止温度的阈值，设置迭代次数阈值，或者当搜索到的最优值连续保持不变时停止搜索。

3. 一种改进的模拟退火算法

3.1 改进思想

基于梯度的各种优化算法易于陷入局部最优解的主要原因是因为它们只在当前迭代点的领域内利用梯度信息进行局部搜索,没有在大范围内进行搜索的策略。由于单独进行大范围搜索将会降低计算效率和解的精度,所以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。在随机搜索算法的迭代过程中,首先按照一定的概率分布在较大的范围内随机地产生试探点,以实现大范围的粗略搜索,然后逐步缩小随机产生试探点的范围,使搜索过程逐渐变为局部的精细搜索。这种搜索策略的基本思想是,在搜索过程的初始阶段找到全局最优解所在的一个局部区域,然后逐渐缩小搜索范围,最终求出全局最优解。如果在初始的大范围搜索阶段没有找到全局最优解所在的区域,在后面的局部搜索阶段将可能陷入局部最优解,因为在局部搜索阶段随机产生的所有试探点将会逐渐集中在当前迭代点的局部范围内。

模拟退火算法是通过适当地控制温度的变化过程实现大范围的粗略搜索与局部的精细搜索相结合的搜索策略。由于产生随机向量的概率密度函数以及接受试探点为新的当前迭代点的接受概率都与温度有关,所以,当温度较高时,随机产生的试探点的散布范围较大,并且能够以较大的概率接受使目标函数值增加的试

探点,从而可实现大范围搜索。随着温度逐渐下降,随机产生的试探点的散布范围逐渐减小,接受使目标函数值增加的试探点的概率也逐渐减小,从而使搜索过程逐渐变为局部探索。如果随着温度逐渐下降,随机产生的试探点越来越集中在当前迭代点的局部范围内,那么,在温度较低时,模拟退火算法将近似于传统的随机搜索算法,因而一旦陷入局部最优解将很难逃脱出来。因此,为了提高模拟退火算法求得全局最优解的可靠性和计算效率,一方面要保持适当的温度下降速度,另一方面要使产生的随机向量保持一定的散布程度,使随机产生的试探点不能都集中在当前迭代点的局部范围内。当产生随机向量的概率密度函数与温度有关时,温度更新函数不仅决定了温度的下降速度,而且决定了在整个退火过程中所产生的随机向量的散布程度。

因此,我们可得出如下确定温度更新函数的一种启发式准则:对于任意给定的包含坐标原点的有界区域 $A \subset R^n$ 以及任意给定的 $k_0 > 0$,温度更新函数应使在所有退火时间 $k \geq k_0$ 所产生的随机向量全部落在区域 A 内的概率等于零。

3.2 随机向量的产生

假定模拟退火算法在任意退火时间 k 所产生的随机向量 Z^k 的各个分量相互独立,为了产生随机向量 Z^k,我们定义如下的一维条件概率密度函数:

$$g(v \mid T) = \begin{cases} \dfrac{T^{1/m}}{m(v+T)^{(m+1)/m}} & v \geq 0 \\ 0 & v < 0 \end{cases} \qquad (3.1)$$

式中 $v \in R, T > 0$ 是给定的参数, $m \geq 1$ 是给定的常数。设 $V_1^k, V_2^k, \cdots, V_n^k$ 是一组具有概率密度函数 $g(\cdot \mid T_k)$ 的随机变量, $W_1, W_2, \cdots, W_n$ 是一组在 $[-1,1]$ 上均匀分布的随机变量, $V_1^k, V_2^k, \cdots, V_n^k$, $W_1, W_2, \cdots, W_n$ 两两相互独立,则模拟退火算法在任意退火时间 k 所产生的随机向量 Z^k 可由如下一组随机变量的函数定义:

$$Z_i^k = \frac{W_i V_i^k}{\sqrt{\sum_{j=1}^{N} W_j^2}}, i = 1, 2, \cdots, n \qquad (3.2)$$

式中 Z_i^k 是随机向量 Z^k 的第 i 个分量。假定对所有的 $i \in \{1, 2, \cdots, n\}, Z_i^k \neq 0$,那么,根据随机变量函数的概率分布,可得随机向量 Z^k 在给定温度 T_k 下的条件概率密度函数 $p(\cdot \mid T_k)$ 为

$$p(z^k \mid T_k) = \frac{1}{2^n} \int_0^1 \cdots \int_n^1 \prod_{i=1}^{n} g\left[\frac{|z_i^k|}{s_i} \sqrt{\sum_{j=1}^{n} s_j^2} \mid T_k \right] \frac{1}{s_i} \sqrt{\sum_{j=1}^{n} s_j^2} ds_1 \cdots ds_n, z_i^k \neq 0, i = 1, 2, \cdots, n \qquad (3.3)$$

式中 $z^k = (z_1^k, z_2^k, \cdots, z_n^k)$。由于具有概率密度函数 $g(\cdot \mid T_k)$ 的随机变量 V_i^k 可通过下式产生:

$$V_i^k = T_k(\frac{1}{U_m} - 1), i = 1,2,\cdots,n \qquad (3.4)$$

式中$U_1, U_2, \cdots, U_n$是一组两两相互独立的在$[0,1]$上均匀分布的随机变量，所以，由随机变量函数(3.2)所定义的随机向量Z^k可通过下式产生：

$$Z_i^k = \frac{W_i T_k}{\sqrt{\sum_{j=1}^{N} W_j^2}}\left(\frac{1}{U_i^m} - 1\right), i = 1,2,\cdots,n \qquad (3.5)$$

根据上述定义的产生随机向量的概率密度函数，利用上述启发式准则，可确定相应的温度更新函数。

3.3 温度更新函数

下面定理给出了在上述启发式准则的意义下概率密度函数(3.1)所对应的温度更新函数。

定理 1　模拟退火算法中产生随机向量的概率密度函数由(3.1)式定义，假定在不同退火时间所产生的随机向量相互独立，那么，如果温度更新函数为

$$T_k = \frac{T_0}{k^m}, k = 1,2,\cdots \qquad (3.6)$$

式中$T_0 > 0$是初始温度，$m \geq 1$与(3.1)式定义的一维概率密度函数$g(\cdot | T_k)$中的m相同，则对于任意给定的包含坐标原点的有界区域$A \subset R^n$以及任意给定的$k_0 > 0$，温度更新函数应使在所有退火时间$k \geq k_0$所产生的随机向量全部落在区域A内的概率等于零。证明略。

通常的模拟退火算法所采用的温度更新函数为$T_k = T_k \cdot T_{k-1}, 0 < T_k < 1, k \geq 1$，根据与物理系统退火过程的类比关系可得，在温度较高时，T_k可以取较小的值，使温度有较快的下降速度，随着温度逐渐减少，T_k的值应逐渐增大，使温度的下降速度逐渐减慢，当T_k接近于零时，T_k应接近于 1。定理 1 给出的温度更新函数可以表示为$T_1 = T_0, T_k = \left(\frac{k-1}{k}\right)^m T_{k-1}, k \geq 2$，相当于$T_1 = 1, T_k = (\frac{k-1}{k})^m, k \geq 2$。可以看出，当$k \geq 2$时，随着退火时间(即迭代次数)$k$的增加，$T_k$的值逐渐增大，当$k \to \infty$时，$T_k \to 1$，并且$m$的值越小，$T_k$趋近于 1 的速度越快，温度的下降速度越慢。由此可见，定理 1 给出的温度更新函数符合与物理系统退火过程的类比关系，而且通过适当地选择 m 的值可以控制温度的下降速度。

在前面给出的模拟退火算法步骤中，采用定理 1 给出的温度更新函数及相应的产生随机向量的概率密度函数可得到一种高效的拟退火全局优化算法。

3.4 边界约束的处理方法

对于具有边界约束的全局优化问题：

$$\min_x f(x)$$
$$s.t. a_i \le x_i \le b_i, i = 1,2,\cdots,n \tag{3.7}$$

式中 $x=(x_1,x_2,\cdots,x_n)$，a_i 和 b_i 分别是 x_i 的下界和上界。为了使模拟退火算法随机产生的试探点满足给定的边界约束条件,可采用如下方法对上述模拟退火算法的 X'^k 做适当的修正。第一种方法是对每个新产生的试探点先检查是否满足给定的边界约束条件,如果满足,则计算该试探点的目标函数值;否则重新产生试探点。第二种方法是先对随机产生的试探点进行适当的处理,保证其满足给定的边界约束条件,然后再计算相应的目标函数值。假定第 k 次迭代所产生的新的试探点记为 $\hat{X}'^k$,它可能不满足给定的边界约束条件,则满足边界约束条件的试探点 X'^k 可通过下式得到:

$$X_i'^k = \begin{cases} b_i - (\hat{X}_i'^k - b_i)\mathrm{MOD}(b_i - a_i) & \hat{X}_i'^k > b_i \\ \hat{X}_i'^k & a_i < \hat{X}_i'^k < b_i \\ a_i - (a_i - \hat{X}_i'^k)\mathrm{MOD}(b_i - a_i) & \hat{X}_i'^k < a_i \end{cases} \tag{3.8}$$

式中 $\hat{X}_i'^k$ 和 $X_i'^k$ 分别是试探点 $\hat{X}'^k$ 和 X'^k 的第 i 个分量，$(a)\mathrm{MOD}(b)$ 表示 a/b 的余数。采用此方法处理边界约束,每次迭代只需产生一个试探点。

4. 数值仿真分析

4.1 测试函数

算例 1.

计算函数 $\sum_{i=1}^{n}[x_i^2 - A\cos(2\pi x_i)], 5.12 \le x_i \le 5.12$，的最小值，其中个体 x 的维数 $n=10$。这是一个具有多个局部极值的函数，其函数值图形如图 4.1 所示。只有一个极小点 $x=(0,0)$，理论最小值 $f(0,0)=0$。

仿真过程如下

（1）初始化 Markov 链长度为 $L=200$，衰减参数为 $K=0.998$，步长因子为 $S=0.01$，初始温度为 $T=100$，容差为 $YZ=1\times10^{-8}$；随机产生初始解，并计算其目标函数值。

（2）在变量的取值范围内，按步长因子随机产生新解，并计算新目标函数值；以 Metropolis 算法确定是否替代旧解，在一种温度下，迭代 L 次。

（3）判断是否满足终止条件：若满足，则结束搜索过程，输出优化值；若不满足，则减小温度，进行迭代优化。优化结束后，其适应度进化曲线如图 4.1 所示。优化后的结果为：$x=[0.0002,-0.0026]$，函数 $f(x)$ 的最小值为 0.0013

采用改进方法的参数与原始方法一样，优化结束后，其适应度进化曲线如图 4.1 所示。优化后的结果为：$x=[-0.0001119,-0.0002934]$，函数 $f(x)$ 的最小值

为 0.000019563

算例 2.

计算函数 $f(x,y)=-20\exp[-0.2\sqrt{0.5(x^2+y^2)}]-\exp[0.5(\cos 2\pi x+\cos 2\pi y)]+e+20$，$-5<x,y<5$ 的最小值。这是一个具有多个局部极值的函数，其函数值图形如图 4.1 所示。只有一个极小点 $x=(0,0)$，理论最小值 $f(0,0)=0$。

仿真过程如下

（1）初始化 Markov 链长度为 $L=200$，衰减参数为 $K=0.998$，步长因子为 $S=0.01$，初始温度为 $T=100$，容差为 $YZ=1\times10^{-8}$；随机产生初始解，并计算其目标函数值。

（2）在变量的取值范围内，按步长因子随机产生新解，并计算新目标函数值；以 Metropolis 算法确定是否替代旧解，在一种温度下，迭代 L 次。

（3）判断是否满足终止条件：若满足，则结束搜索过程，输出优化值；若不满足，则减小温度，进行迭代优化。优化结束后，其适应度进化曲线如图 4.1 所示。优化后的结果为：$x=[-0.000004205,0.00005981]$，函数 $f(x)$ 的最小值为 0.00020693

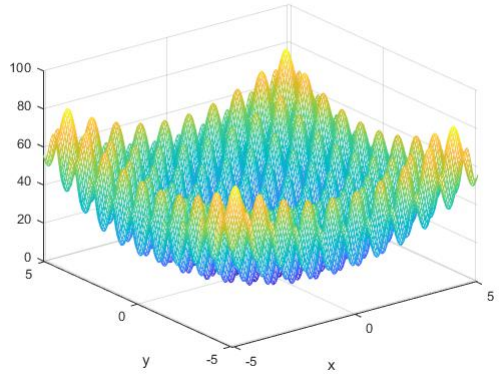
采用改进方法的参数与原始方法一样，优化结束后，其适应度进化曲线如图 4.1 所示。优化后的结果为：$x=[-0.0001262,0.0002503]$，函数 $f(x)$ 的最小值为 0.00079500

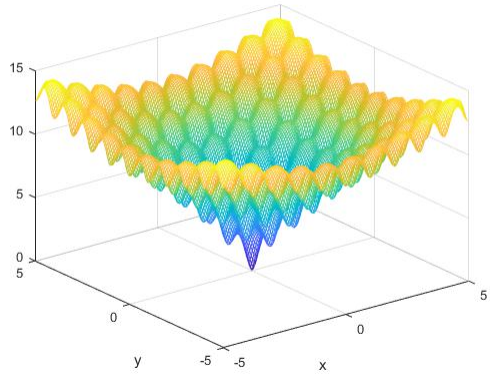
图 4.1

算例 1

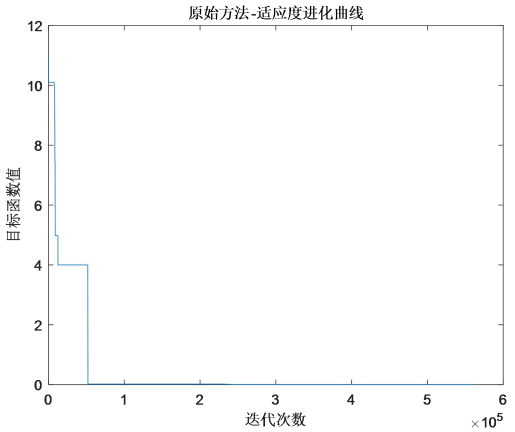
算例 2

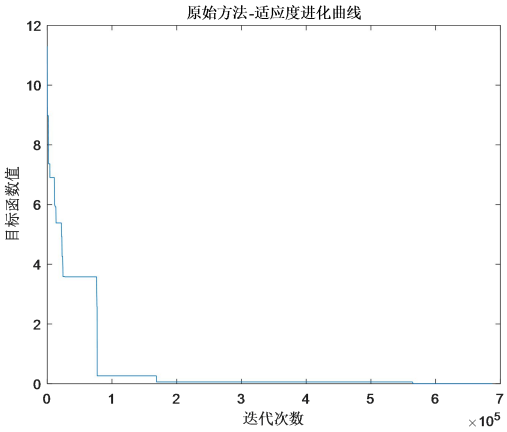
原始图像



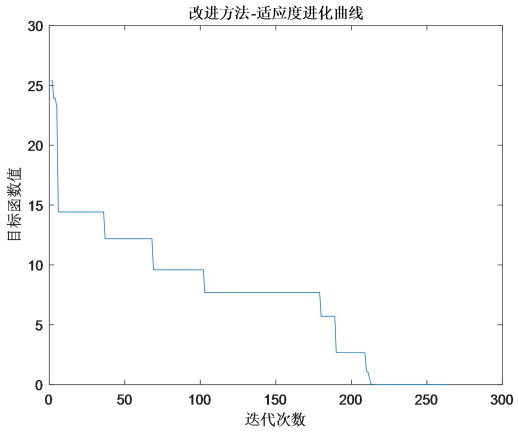


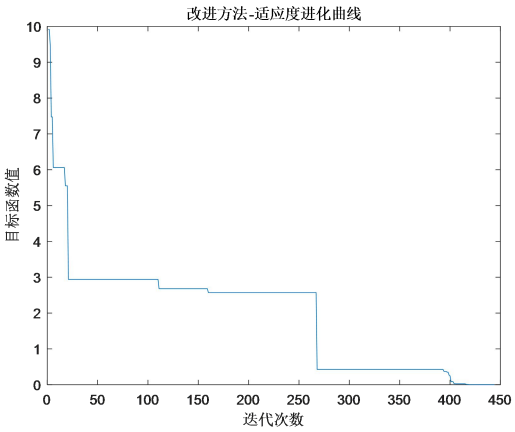
原始方法优化





改进方法优化





4.2 整数规划问题之 0-1 背包问题

问题描述：

一个 0-1 背包问题即：给定一个可装重量 M 的背包及 n 件物品，物品 i 的重量和价值分别为W_i、C_i，i=1，2，...，n。要选若干件物品装入背包，使其价值之和为最大。

求解的模拟退火算法描述为：

1. 数学模型

$$\max z = C_1X_1 + ... + C_nX_n$$
$$m = W_1X_1 + ... + W_nX_n \le M$$
$$X_i \in \{0,1\}, i = 1,2,...n$$

2. 初始解

可任选解空间中的一向量为初始解，不妨选择零向量为初始解。

3. 新解的产生

随机选取物品 i：若 i 不在背包中，则将其直接装入背包，或者同时从背包中随机取出另一物品 j；若 i 在背包中，则将其取出，并同时随机装入另一物品 j。

4. 目标函数差及背包重量差

设当前解以及新解对应的目标函数值和背包重量分别为 z，z1 和 m,m1,则关

于新解的目标函数差以及背包重量差分别为

$$\Delta z = z_1 - z$$

$$\Delta m = m_1 - m$$

5. 新解的接收概率

新解的接收概率是扩充了可行性测定的 Metropolis 准则

$$\begin{cases} 0 & \mathrm{m} + \Delta \mathrm{m} > \mathrm{M} \\ 1 & \mathrm{m} + \Delta \mathrm{m} \le \mathrm{M} \text{且} \Delta + \succ 0 \\ \exp(\Delta z / t) \end{cases}$$

其中 t 为温度参数。

6. 冷却计划表

包括如下四个参数：

t0: 控制温度 t 的初值

α：控制温度 t 的衰减函数

L： Markov 链的长度

S： 停止准则，常用的停止准则是，在相继的 K 个 Markov 链中在解无任何改变时即停止算法运行

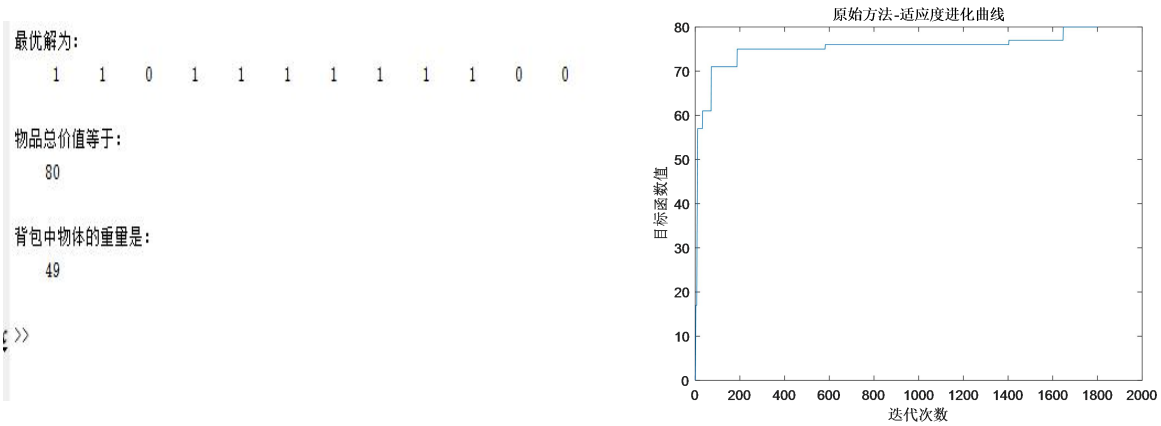
仿真过程如下：

（1）背包中有 12 个物体，物体重量分别为[5 10 13 4 3 11 13 10 8 16 7 4]对应每个物体的价值为[2 5 18 3 2 5 10 4 11 7 14 6]

（2）初始化参数 X 为 0，以 Metropolis 算法确定是否替代旧路径，在一种温度下，迭代 L 次。

（3）判断是否满足终止条件：若满足，则结束搜索过程，输出优化值 X；若不满足，则衰减温度，进行迭代优化。优化后的结果如图 4.2 所示，适应度进化曲线如图 4.2 所示。

图 4.2



用时 0.129s

4.3 动态规划算法解决 0-1 背包问题

动态规划解决背包问题的过程：

在解决问题之前，为描述方便，首先定义一些变量：Vi 表示第 i 个物品的价值，Wi 表示第 i 个物品的体积，定义 V(i,j)：当前背包容量 j，前 i 个物品最佳组合对应的价值，同时背包问题抽象化（X1，X2，…，Xn，其中 Xi 取 0 或 1，表示第 i 个物品选或不选）。

1、建立模型，即求 max(V1X1+V2X2+…+VnXn)；

2、寻找约束条件，W1X1+W2X2+…+WnXn<capacity；

3、寻找递推关系式，面对当前商品有两种可能性：

包的容量比该商品体积小，装不下，此时的价值与前 i-1 个的价值是一样的，即 V(i,j)=V(i-1,j)；

还有足够的容量可以装该商品，但装了也不一定达到当前最优价值，所以在装与不装之间选择最优的一个，即 V(i,j)=max｛V(i-1,j)，V(i-1,j-w(i))+v(i)｝。

其中 V(i-1,j)表示不装，V(i-1,j-w(i))+v(i) 表示装了第 i 个商品，背包容量减少 w(i)，但价值增加了 v(i)；

由此可以得出递推关系式：

j<w(i)　　V(i,j)=V(i-1,j)

j>=w(i)　　V(i,j)=max｛V(i-1,j)，V(i-1,j-w(i))+v(i)｝

这里需要解释一下，为什么能装的情况下，需要这样求解（这才是本问题的关键所在！）：

可以这么理解，如果要到达 V(i,j)这一个状态有几种方式？

肯定是两种，第一种是第 i 件商品没有装进去，第二种是第 i 件商品装进去了。没有装进去很好理解，就是 V(i-1,j)；装进去了怎么理解呢？如果装进去第 i 件商品，那么装入之前是什么状态，肯定是 V(i-1,j-w(i))。由于最优性原理（上文讲到），V(i-1,j-w(i))就是前面决策造成的一种状态，后面的决策就要构成最优策略。两种情况进行比较，得出最优。

背包问题的最优解回溯：

通过上面的方法可以求出背包问题的最优解，但还不知道这个最优解由哪些商品组成，故要根据最优解回溯找出解的组成，根据填表的原理可以有如下的寻解方式：

V(i,j)=V(i-1,j)时，说明没有选择第 i 个商品，则回到 V(i-1,j)；

V(i,j)=V(i-1,j-w(i))+v(i)时，说明装了第 i 个商品，该商品是最优解组成的一部分，随后我们得回到装该商品之前，即回到 V(i-1,j-w(i))；

一直遍历到 i＝0 或者 j=0 结束为止，所有解的组成都会找到。

解决问题：

仍以上述背包问题为例，编写 MATLAB 程序求解最优值。最终结果为：

```
物品总价值等于：
    80

背包中物体的重量是：
    50

最优解为：
     1     0     0     0     1     1     1     1     1     1     1     1
```

用时 0.006s

4.4 模拟退火算法解决飞行轨迹问题（路径规划问题）

1. 初始化路径：

模拟退火算法路径规划中初始化一条随机路径：

1. 找到一条间断的路径，即从栅格的每一行中随机取出一个不是障碍物的元素。这种操作在每一行中都取了一个元素，并且这些元素都是自由栅格，因此做完第一步后得到的是一条间断路径。

2. 将间断的路径连续化，在这一步中，需要先从路径的第一个栅格节点判断与之相邻的下一个栅格节点是否连续。

若连续：迭代节点，判断下一个与之相邻的栅格节点

若不连续：那么采用中点插值法计算这两个栅格节点的中点。

a. 若中点是自由栅格，就插入到路径中。

b. 若中点不是自由栅格，那么进一步判断中点栅格周围 8 个邻域的栅格状态。找到邻域中不是障碍的栅格，再插入到路径中。

若 8 个邻域都是障碍，那么说明这条路径没法进行连续化处理，也就是这条路行不通，此时删除这条路径，从 1 步骤再生成一条间隔路径，直到连续化完成。

2. 计算适应度值：

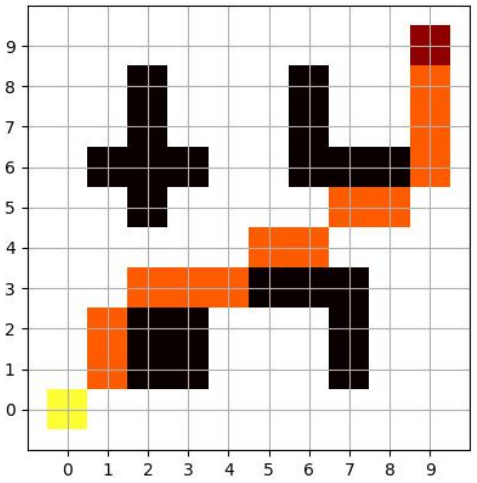
路径规划中适应度函数就是求解路径长度的函数，适应度值就是路径长度。代码中给定相邻的栅格图为 1 个单位长度，则对角的长度为 1.4（根号 2）。具体解释参照遗传算法博客，与遗传算法不同的一点是，模拟退火算法计算出来的适应度值就是一个数值，而遗传算法计算出来的是，路径长度数值组成的列表，也就是列表中的每一个元素就是路径长度。

3. 产生的新路径是在前一个路径的信息基础上产生的，因此模拟退火算法也算是启发式搜索算法，新的路径从上一个路径的信息中获得。在本文中，采用的是遗传算法中变异的操作来获得新的路径。

4. 具体操作是：在前一条路径中随机找到除了起点和终点以外的另外两个栅格节点，然后将这两个栅格节点中间的路径全部删除，这样子就得到了一条间隔路径，再利用间隔路径连续化操作，使间隔路径连续，随之得到一条新的路径。

5. 与伪代码中的方法一致：如果新路径的长度小于原路径的长度，则用新路径更新原路径的信息；若新路径的长度大于原路径的长度，则计算接受此条新路径

的概率，称为接受概率。在随机产生一个 0-1 的随机数，若接受概率大于随机数，则用新路径更新原路径的信息，否则放弃新的路径，依旧使用原路径进行搜索。
6. 这里就是用到了模拟退火算法的精髓，接受概率，也就是前文提到的 P=e-|f(A)-f(B)|/T P= e-|f(A)-f(B)|/T 直接将新路径长度与原始路径长度的差值代入公式，然后把当前的温度值也代入公式，就可以计算出当前接受新路径的概率，此时随机生成一个 0-1 的概率 R，若 P>R，则接受新路径，否则拒绝新路径，依旧使用原始路径。需要注意，在同一个温度下需进行多次搜寻。其中两个 for 循环，外层循环是温度变化次数，每次迭代会将温度以下降系数步长下降；内层循环是每个温度下迭代的次数。总共迭代次数是两个数值的乘积。
7. 输出结果：




```
开始时间:  1592361067.5596504
初始路径 [0, 11, 2, 3, 14, 15, 16, 7, 8, 18, 28, 39, 38, 47, 46, 45, 56, 55, 54, 53, 42, 41, 51, 60, 71, 81, 92, 83, 73, 74, 75, 85, 96, 87, 77, 78, 89, 99]
初始路径距离 42.999999999999986
最终路径 [0, 11, 21, 32, 33, 34, 45, 46, 57, 58, 69, 79, 89, 99]
最终路径距离 Search workspace (ctrl + click)
结束时间:  1592361068.9449325
共用时 1.385282039642334 秒
```

5. 结论

模拟退火算法具有适用范围广，求得全局最优解的可靠性高，算法简单，便于实现等优点。模拟退火算法在搜索策略上与传统的随机搜索方法不同，它不仅引入了适当的随机因素，而且还引入了物理系统退火过程的自然机理。这种自然机理的引入使模拟退火算法在迭代过程中不仅接受使目标函数值变“好”的试探点，而且还能够以一定的概率接受使目标函数值变“差”的试探点，接受概度随着温度的下降逐渐减小。模拟退火算法的这种搜索策略有利于避免搜索过程因陷入局部最优解而无法自拔的弊端，有利于提高求得全局最优解的可靠性。

从图 4.1 可以看出，与传统的采用指数形式温度更新函数的模拟退火算法相比，本改进的模拟退火算法可以显著地减少计算目标函数值的平均次数，提高求得全局最优解的计算效率，是一种高效的模拟退火全局优化算法。

参考文献

[1] KirkPatrick S,Gelatt C,Vecchi M.Optimization by simulated annealing [J].S-cience,1983,220:671-680.

[2] 杨若黎, 顾基发.一种高效的模拟退火全局优化算法[J]. 系统工程理论与实践，1997,17(5): 30-33.

[3] Cerny V. Thermodynamical Approach to the Traveling Salesman Problem: An Effieient Simulation Algorithm [J]. Journal of Optimization Theory and Applications, 1985(45): 41-45.

[4] Corana A，Marchesi M，Martini C, et al. Minimizing Multimodal Funetion s for Continuous Variables with the “Simulated Annealing” Algorithm [J]. ACM Transaetions on Mathematical software, 1987(13): 262-280.

[5] Yao Xin, Li Guojie. General Simulated Annealing [J]. The Journal of Com puter Science and Technology，1991, 6(4): 329-338.

[6] Laarhoven P, Aarts E. Lenstra Jobshop scheduling by simulated annealing [J]. Ops.Res., 1992, 40(l): 113-125.

[7] Tarek M，Nabhan A，Ibert Y, et al. Parallel simulated annealing algorithm with low communication over head [J], IEEE Transaetionson Parallel and Distributed systems，1995, 6(12): 1226-1233.

[8] 李士勇, 李研. 智能优化算法原理与应用[M]. 哈尔滨: 哈尔滨工业大学出版社，2012: 40-46.

[9] https://blog.csdn.net/qq_34942642/article/details/106332121