模拟退火算法

一、模拟退火算法概念

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，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。根据 Metropolis 准则，粒子在温度 T 时趋于平衡的概率为 e-ΔE/(kT)，其中 E 为温度 T 时的内能，ΔE 为其改变量，k 为 Boltzmann 常数。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 f，温度 T 演化成控制参数 t，即得到解组合优化问题的模拟退火算法：由初始解 i 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，并逐步衰减 t 值，算法终止时的当前解即为所得近似最优解，这是基于蒙特卡罗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(Cooling Schedule)控制，包括控制参数的初值 t 及其衰减因子 Δt、每个 t 值时的迭代次数 L 和停止条件 S。

二、模拟退火算法的模型

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。

模拟退火的基本思想：

(1) 初始化：初始温度 T(充分大)，初始解状态 S(是算法迭代的起点)，每个 T 值的迭代次数 L

(2) 对 k=1，……，L 做第(3)至第 6 步：

(3) 产生新解 S'

(4) 计算增量 Δt'=C(S')-C(S)，其中 C(S)为评价函数

(5) 若 Δt'<0 则接受 S'作为新的当前解，否则以概率 exp(-Δt'/T)接受 S'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。

终止条件通常取为连续若干个新解都没有被接受时终止算法。

(7) T 逐渐减少，且 T->0，然后转第 2 步。

算法对应动态演示图：

模拟退火算法新解的产生和接受可分为如下四个步骤：

第一步是由一个产生函数从当前解产生一个位于解空间的新解；为便于后续的计算和接受，减少算法耗时，通常选择由当前新解经过简单地变换即可产生新解的方法，如对构成新解的全部或部分元素进行置换、互换等，注意到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。

第二步是计算与新解所对应的目标函数差。因为目标函数差仅由变换部分产生，所以目标函数差的计算最好按增量计算。事实表明，对大多数应用而言，这是计算目标函数差的最快方法。

第三步是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 Δt'<0 则接受 S' 作为新的当前解 S，否则以概率 exp(-Δt'/T) 接受 S' 作为新的当前解 S。

第四步是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。而当新解被判定为舍弃时，则在原当前解的基础上继续下一轮试验。

模拟退火算法与初始值无关，算法求得的解与初始解状态 S(是算法迭代的起点)无关；模拟退火算法具有渐近收敛性，已在理论上被证明是一种以概率 1 收敛于全局最优解的全局 优化算法；模拟退火算法具有并行性。.

三、模拟退火算法的应用领域

模拟退火算法是解 NP 完全组合优化问题的有效近似算法，将该算法应用于路径优化问题，利用该算法对类似货郎担问题的路径问题进行求解；针对城市道路行走不同的目标条件

(路径最短、时间最短)进行优化,选择最佳行走路径;并将用该算法优化得到的计算结果与树形算法进行比较,显示该算法能够克服传统优化算法易陷入局部极值的缺点,同时表明该算法在解类似货郎担交通路径方面的问题时有较高的精确性。因而该算法在解决城市道路交通问题方面具有一定的实用价值。

在企业运营与管理中,管理者总是希望把人员最佳分派以发挥其优势,从而降低成本,提高效益.例如,某公司需完成 m 项任务,恰好有 n 名员工可承担这些任务 $(n\geq m)$;每项任务只能由一名员工来做,每名员工也只能做一项任务;不同的员工完成各项任务的成本不同.这样就可采用模拟退火算法将企业的人员分配做到最佳的分配。

四、具体案例

C#数值计算之模拟退火法简介

摘要

本文简介了模拟退火的基本思想，以于模拟时的主要参数的选择根据，然后给出一个求二维函数极值的具体问题和解法,并给出 C# 源代码。

l 概述

在管理科学、计算机科学、分子物理学和生物学以及超大规模集成电路设计、代码设计、图像处理和电子工程等科技领域中，存在大量组合优化瓶。其中许多问题如货郎担问题、图着色问题、设备布局问题以及布线问题等，至今没有找到有效的多项式时间算法。这些问题已被证明是 NP 完全问题。

1982 年,KirkPatrick 将退火思想引入组合优化领域，提出一种解大规模组合优化问题的算法，对 NP 完全组合优化问题尤其有效。这源于固体的退火过程，即先将温度加到很

高，再缓慢降温(即退火)，使达到能量最低点。如果急速降温(即为淬火)则不能达到最低点。

即：模拟退火算法是一种能应用到求最小值问题或基本先前的更新的学习过程（随机或决定性的）。在此过程中，每一步更新过程的长度都与相应的参数成正比，这些参数扮演着温度的角色。然后，与金属退火原理相类似，在开始阶段为了更快地最小化或学习，温度被升得很高，然后才（慢慢）降温以求稳定。

模拟退火算法的主要思想

就函数最小值问题来说，模拟退火的主要思想是：在搜索区间（二维平面中）随机游走（即随机选择点），再以 Metropolis 抽样准则，使随机游走逐渐收敛于局部最优解。而温度即是 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时过程向局部或全局最优解移动的快慢。

算法）一起 Metropolis 冷却参数表、领域结构和新解产生器、接受准则和随机数产生器（即.

算法设计与分析 计算机 101-04 顾鑫

构成算法的三大支柱。

重点抽样与 Metroplis 算法：

Metropolis 是一种有效的重点抽样法，其算法为：系统从能量一个状态变化到另一个状态时，相应的能量从 E1 变化到 E2，概率为 p = exp[- (E2- E1)/kT]° 如果 E2 < E1，系

统接收此状态，否则，以一个随机的概率接收此或丢弃此状态。这种经常一定次数的迭代，系统会逐渐趋于一引稳定的分布状态。

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），以一定机率接受。模拟退火方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。

其中温度是一个 $Metropolis$ 的重要控制参数，模拟退火可视为递减控制参数什时 $Metroplis$ 算法的迭代。开始 T 值大，可能接受较差的恶化解，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；但可使马可夫链越小，到达准平衡分布的时间越短，

参数的选择：

我们称调整模拟退火法的一系列重要参数为冷却进度表。它控制参数 T 的初值及其衰减函数，对应的 $MARKOV$ 链长度和停止条件，非常重要。

一个冷却进度表应当规定下述参数：

1．控制参数 t 的初值 $t0$;

2．控制参数 t 的衰减函数；

3．马尔可夫链的长度 Lk。（即每一次随机游走过程，要迭代多少次，才能趋于一个准平

衡分布，即一个局部收敛解位置）

4．结束条件的选择

有效的冷却进度表判据：

一．算法的收敛：主要取决于衰减函数和马可夫链的长度及停止准则的选择

二．算法的实验性能：最终解的质量和 CPU 的时间

3

参数的选择：

一）控制参数初值 $T0$ 的选取

一般要求初始值 $t0$ 的值要充分大，即一开始即处于高温状态，且 $Metropolis$ 的接收率约为 1。

二）衰减函数的选取

衰减函数用于控制温度的退火速度，一个常用的函数为：$T(n+1)=K*T(n)$，其中 K 是一个非常接近于 1 的常数。

三）马可夫链长度 L 的选取

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。

四）终止条件

有很多种终止条件的选择，各种不同的条件对算法的性能和解的质量有很大影响，本文只介绍一个常用的终止条件。即上一个最优解与最新的一个最优解的之差小于某个容差，即可停止此次马尔可夫链的迭代。

以上说明可能太过于抽象，下一节将以一个实际的例子来说明，其中所有的源码已贴出，可以从中了解到很多细节。

使用模拟退火法求函数 $f(x,y)=5sin(xy)+x2+y2$ 的最小值

解：根据题意，我们设计冷却表进度表为：

即初始温度为 100

衰减参数为 0.95

马可夫链长度为 10000

Metropolis 的步长为 0.02

结束条件为根据上一个最优解与最新的一个最优解的之差小于某个容差。

使用 METROPOLIS 接受准则进行模拟，程序如下

```
/*

 的最小值 f(x,y) = 5sin(xy) + x^2 + y^2 模拟退火法求函数。

算法设计与分析 计算机 101-04 顾鑫

。结束条件为两次最优解之差小于某小量

*/

using System;

namespace SimulateAnnealing

{

class Class1

{
```

```
// 要求最优值的目标函数

static double ObjectFunction( double x, double y )

{

double z = 0.0;

z = 5.0 * Math.Sin(x*y) + x*x + y*y;

return z;

}

[STAThread]

static void Main(string[] args)

{

// 搜索的最大区间

const double XMAX = 4;

const double YMAX = 4;

// 冷却表参数
```

```
int MarkovLength = 10000; // 马可夫链长度
5

double DecayScale = 0.95; // 衰减参数

double  StepFactor = 0.02; // 步长因子

double Temperature = 100; // 初始温度

double  Tolerance = 1e-8; // 容差

double PreX,NextX; // prior and next value of x

double PreY,NextY; // prior and next value of y

double  PreBestX, PreBestY; // 上一个最优解

double BestX,BestY; // 最终解

double AcceptPoints = 0.0; // Metropolis 过程中总接受点

Random rnd = new Random();

// 随机选点

PreX = -XMAX *rnd.NextDouble() ;
```

```
PreY = -YMAX * rnd.NextDouble();

PreBestX = BestX = PreX;

PreBestY = BestY = PreY;

// 每迭代一次退火一次(降温), 直到满足迭代条件为止

do

{

Temperature *=DecayScale;

AcceptPoints = 0.0;

// 在当前温度 T 下迭代 loop(即 MARKOV 链长度)次

for (int i=0;i<MarkovLength;i++)
```

```
{

// 1) 在此点附近随机选下一点
```

```
do

{

NextX = PreX + StepFactor*XMAX*(rnd.NextDouble()-0.5);

NextY = PreY + StepFactor*YMAX*(rnd.NextDouble()-0.5);

}

while ( !(NextX >= -XMAX && NextX <= XMAX && NextY >= -YMAX && NextY <=
YMAX) );

// 2) 是否全局最优解

if (ObjectFunction(BestX,BestY) > ObjectFunction(NextX,NextY))

{

// 保留上一个最优解

PreBestX =BestX;

PreBestY = BestY;

// 此为新的最优解

BestX=NextX;
```

```
BestY=NextY;

}

// 3) Metropolis 过程

if( ObjectFunction(PreX,PreY) - ObjectFunction(NextX,NextY) > 0 )

{
```



```
, // 接受此处 lastPoint 即下一个迭代的点以新接受的点开始
PreX=NextX;

PreY=NextY;

 AcceptPoints++;

}

else

{

/ ObjectFunction(PreX,PreY) ) ( double change = -1 * ObjectFunction(NextX,NextY) -
Temperature ;

if( Math.Exp(change) > rnd.NextDouble() )
```

{

PreX=NextX;

PreY=NextY;

AcceptPoints++;

}

// 不接受，保存原解

}

}

, ObjectFunction (PreX, PreY), Temperature); ,PreX, PreY

> (PreBestX, PreBestY)) –Math.Abs(ObjectFunction(BestX,BestY) ObjectFunction }

while(

Tolerance);

,BestX, BestY); 滿獺汯？牖瑩？湩？最小值在点

滿獺汯？牖瑩？湩？鄹最小值为:{0},ObjectFunction(BestX, BestY));

}

}

}

结果：

最小值在点:-1.956,1.618

最小值为:-2.686

后记：

一开始在网上搜索模拟退火的资料并想作为C#数值计算的一个例子，找不到现成的源码。后来自己实验了很久，终于将此程序写出。

本文尽量避免太过学术化，如数学和物理名称和公式，仓促下笔，有很多地方可能讲得不是很清楚，希望老师谅解。

模拟退火还可以应用到其它更多更复杂的问题，如“推销员问题”等组合优化问题。本例只是求一个二维函数的最小值问题，而且其冷却表参数的选择也过于简单，只能作用一个初步的入门简介。

五、总结

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，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。根据Metropolis准则，粒子在温度T时趋于平衡的概率为e-ΔE/(kT)，其中E为温度T时的内能，ΔE为其改变量，k为Boltzmann常数。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f，温度T演化成控制参数t，即得到解组合优化问题的模拟退火算法：由初始解i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，并逐步衰减t值，算法终止时的当前解即为所得近似最优解，这是基于蒙特卡罗迭代求解法

的一种启发式随机搜索过程。退火过程由冷却进度表(Cooling Schedule)控制，包括控制参数的初值t及其衰减因子Δt、每个t值时的迭代次数L和停止条件S。