实验12 多元宇宙-模拟退火算法

(全局搜索MVO，局部搜索SA)

学号：1131190111 名字：唐川淇

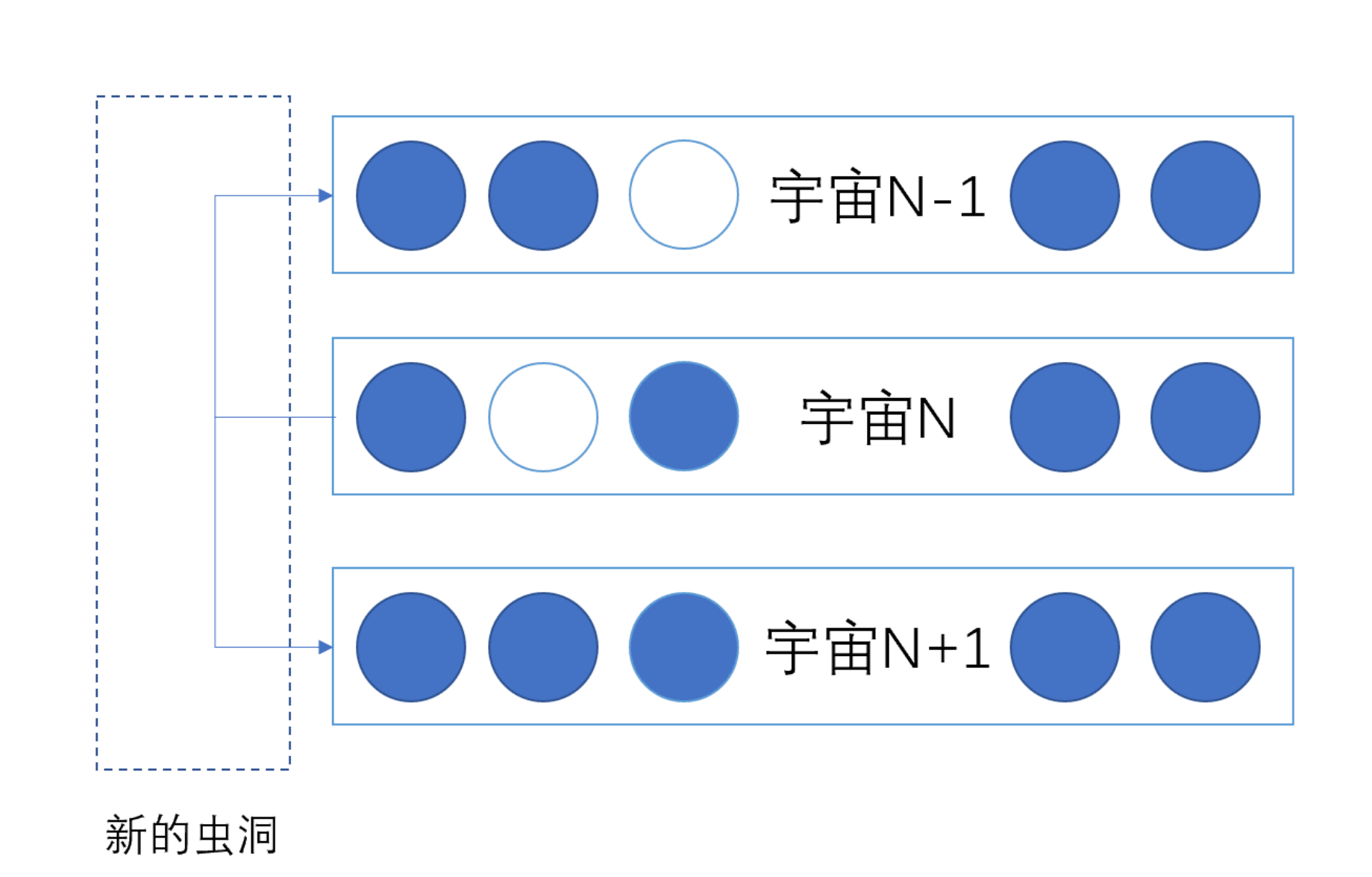
# 问题分析

由于多元宇宙优化算法中虫洞只存在与最优宇宙与所有宇宙之间，这导致的所有宇宙都会趋于最优宇宙而变化而缺少对于空间的探索能力。而模拟退火算法中对于每一个温度态都会对当前解产生随机扰动，并且以一定的概率随即变换，这样就增强了算法的全局搜索能力。由于模拟退火算法具有的良好特性，可以考虑在多元宇宙算法中加入模拟退火算法来增强其全局搜索能力而避免陷入局部最优解。

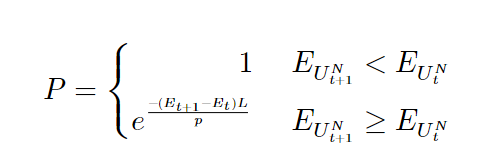
# 模型的建立与求解

## 算法改进

考虑在某一次迭代中，宇宙会向其附近的空间建立虫洞，示意图如图。

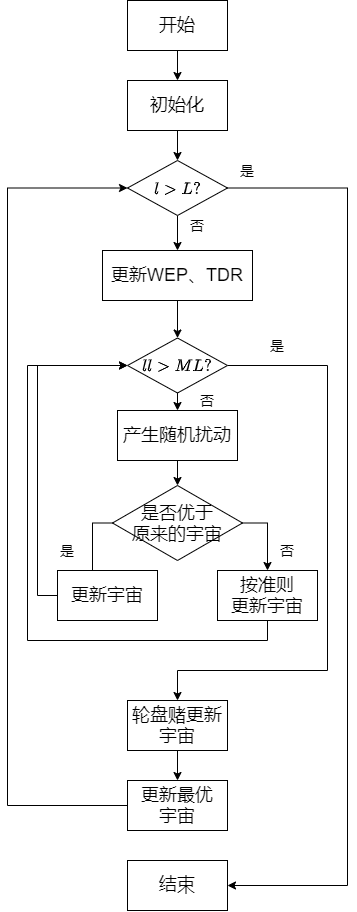


宇宙N-1和宇宙并不是真正存在的宇宙，而是对于宇宙N产生随机扰动之后产生的新宇宙，宇宙N会以类似模拟退火中的Metropolis准则向新宇宙转移。准则定义如下：

公式中E代表宇宙N在第t次迭代中的膨胀率，如果新产生的宇宙的膨胀率低于原先宇宙的膨胀率，那么原来的宇宙就会通过虫洞将新宇宙的粒子转移给自己。反之，如果新产生的宇宙的膨胀率高于原来的宇宙并不会直接将概率定为0，为了增加全局搜索能力，宇宙会按照概率建立虫洞。

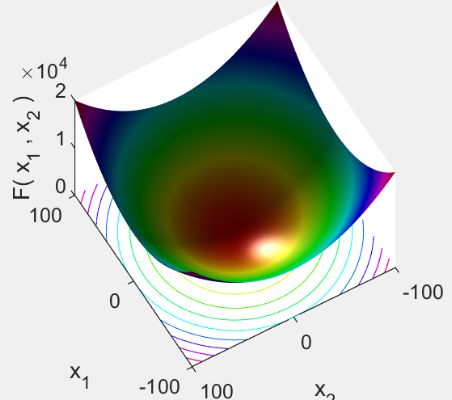
由此MVO-SA算法建立完毕，得到新的算法的算法流程图如图所示。



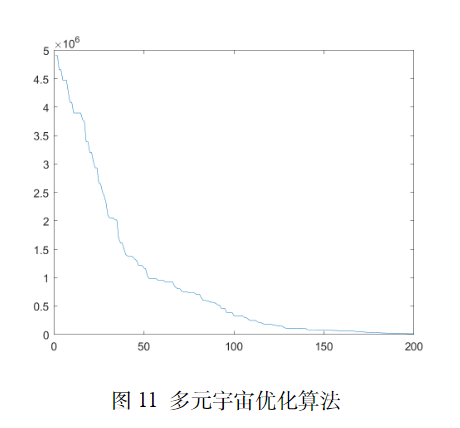
## 实验测试

首先设定函数如，函数是对每个分量的平方求和。

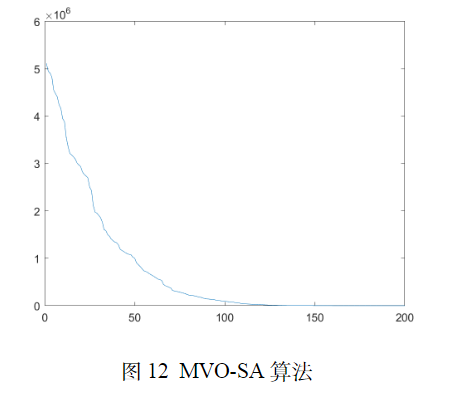




函数维度为50，对于每一维的范围均为[-600,600]，函数的第一维与第二维在三维空间中可以展示如图。有图像可以轻易发现，该函数在选定空间的极值为0，当x取为[0,0,0,… ,0]时可以达到极值，下面分别使用传统的多元宇宙优化算法和MVO-SA算法求解该函数的极值。



首先使用原始多元宇宙优化算法，设定宇宙数目为60，最大迭代次数为500，使用传统的多元宇宙优化算法对函数求极值，得到的膨胀率变化曲线如图。膨胀率随着迭代次数的增加逐渐下降，最终趋于稳定。



同样设定宇宙数目为60，最大迭代次数为500，模拟退火的循环次数为10，p为10，使用MVO-SA算法计算上述函数的极值，得到的膨胀率变化曲线如图所示。

## 结论

发现MVO-SA算法相较于传统的多元宇宙优化算法收敛速度更快，由于MVO-SA算法增加了对于宇宙周边空间的搜索能力，所以其膨胀率曲线更加平滑。MVO-SA算法在保留多元宇宙优化算法的优良性的同时，增强了其全局搜索能力，同时具有更好的收敛速度，在实际应用中也非常有价值。

附录

|  |
| --- |
| 代码 |
| 介绍： main.m |
| clear;  close all;  clc;  % 多元宇宙优化  Universes\_no=10; % 宇宙数量  Max\_iteration=200; % 最大迭代数  [lb,ub,dim,fobj]=mvo\_int(); % 初始化参数  [Best\_score,Best\_pos,cg\_curve]=...  MVO(Universes\_no,Max\_iteration,lb,ub,dim,fobj); %MVO优化算法  plot(cg\_curve); |

|  |
| --- |
| 代码 |
| 介绍： mvo\_int.m |
| function [lb,ub,dim,fobj]=mvo\_int()  % 设置参数  lb=zeros(1,50)-600; % 下界  ub=zeros(1,50)+600; % 上界  dim=50; % 维数  fobj=@F; % 函数  function O=F(x)  O=0;  for i=1:dim  O=O+x(i)^2;  end  end  end |

|  |
| --- |
| 代码 |
| 介绍： mvo.m |
| function [Best\_universe\_Inflation\_rate,Best\_universe,Convergence\_curve]=  MVO(N,Max\_time,lb,ub,dim,fobj)  % 虫洞存在的最大与最小值  WEP\_Max=1;  WEP\_Min=0.2;  % 模拟退火循环次数  ML=10;  Best\_universe=zeros(1,dim); % 最佳宇宙位置  Best\_universe\_Inflation\_rate=inf; % 最佳宇宙适应值  Universes=zeros(N,dim); % 初始化宇宙  for i=1:dim  ub\_i=ub(i);  lb\_i=lb(i);  Universes(:,i)=rand(N,1).\*(ub\_i-lb\_i)+lb\_i;  end  Convergence\_curve=zeros(1,Max\_time); % 收敛曲线  Time=1; % 迭代器  while Time<Max\_time+1  WEP=WEP\_Min+Time\*((WEP\_Max-WEP\_Min)/Max\_time); % 更新WEP  TDR=1-((Time)^(1/6)/(Max\_time)^(1/6)); % 计算TDR  Inflation\_rates=zeros(1,size(Universes,1)); % 膨胀率初始化    for i=1:size(Universes,1)  % 边界处理  Flag4ub=Universes(i,:)>ub; % 超出上界  Flag4lb=Universes(i,:)<lb; % 超出下界  Universes(i,:)=(Universes(i,:).\*(~(Flag4ub+Flag4lb)))+  ub.\*Flag4ub+lb.\*Flag4lb;  % 计算膨胀率  Inflation\_rates(1,i)=fobj(Universes(i,:));  % 更行最优宇宙  if Inflation\_rates(1,i)<Best\_universe\_Inflation\_rate  Best\_universe\_Inflation\_rate=Inflation\_rates(1,i);  Best\_universe=Universes(i,:);  end  end  % 模拟退火  for kk=1:ML  for pp=1:N  rr=rand(1,50)\*10-5;  newUiverse=Universes(pp,:)+rr;    Flag4ub=newUiverse>ub; % 超出上界  Flag4lb=newUiverse<lb; % 超出下界  newUiverse=(newUiverse.\*(~(Flag4ub+Flag4lb)))+  ub.\*Flag4ub+lb.\*Flag4lb;    new\_rate=fobj(newUiverse);  if new\_rate < Inflation\_rates(pp)  Universes(pp,:)=newUiverse;  Inflation\_rates(pp)=new\_rate;  else  r2=rand();  Met=exp(-(new\_rate - Inflation\_rates(pp))\*Time/10);  if r2<Met  Universes(pp,:)=newUiverse;  Inflation\_rates(pp)=new\_rate;  end  end    end  end    % 宇宙按膨胀率排序（升序）  [sorted\_Inflation\_rates,sorted\_indexes]=sort(Inflation\_rates);  Sorted\_universes=Universes;  for newindex=1:N  Sorted\_universes(newindex,:)=Universes(sorted\_indexes(newindex),:);  end  % 标准化为单位向量或矩阵，规范化矩阵行  normalized\_sorted\_Inflation\_rates=normr(sorted\_Inflation\_rates);  % 这里有问题  Universes(1,:)= Sorted\_universes(1,:);    for i=2:size(Universes,1)  Back\_hole\_index=i;  for j=1:size(Universes,2) % Universes列数  r1=rand();  if r1<normalized\_sorted\_Inflation\_rates(i)  % 轮盘赌，若随机数小于膨胀率，则白变为黑  White\_hole\_index=RouletteWheelSelection(-sorted\_Inflation\_rates);  if White\_hole\_index==-1  White\_hole\_index=1;  end  Universes(Back\_hole\_index,j)=Sorted\_universes(White\_hole\_index,j);  end  r2=rand();  if r2<WEP  r3=rand();  if r3<0.5  Universes(i,j)=Best\_universe(1,j)+  TDR\*((ub(j)-lb(j))\*rand+lb(j));  end  if r3>0.5  Universes(i,j)=Best\_universe(1,j)-  TDR\*((ub(j)-lb(j))\*rand+lb(j));  end  end    end  end  Convergence\_curve(Time)=Best\_universe\_Inflation\_rate;  Time=Time+1;  end  end |

|  |
| --- |
| 代码 |
| 介绍： RouletteWheelSelection.m |
| function choice = RouletteWheelSelection(weights)  % 轮盘赌  accumulation = cumsum(weights);  p = rand() \* accumulation(end);  chosen\_index = -1;  for index = 1 : length(accumulation)  if (accumulation(index) > p)  chosen\_index = index;  break;  end  end  choice = chosen\_index;  end |