

全局优化的了望算法

蔡延光¹, 钱积新², 孙优贤²

(1. 广东工业大学 自动化学院, 广东 广州 510090;

2. 浙江大学 工业控制技术国家重点实验室, 浙江 杭州 310027)

摘要: 提出求解全局优化问题的了望算法. 了望算法利用了望技术确定群山最高点的常识, 通过望管理机制、了望点产生策略、局部问题构造与求解机制, 能在较短的时间内求解全局优化问题. 大量的测试表明, 了望算法具有较高的收敛率和较强的获得问题全部解的能力, 对初始点几乎没有依赖, 参数选择简单. 了望算法能够保证在迭代过程中迭代点的质量逐步变好, 所提出的三层次记忆机制极大地提高了望算法的收敛速度. 大量的对比测试也表明, 在收敛率和全局搜索能力等方面了望算法较遗传算法有一定的优势, 且在大多数情况下了望算法耗时较少. 由于了望算法是根据人类的高级行为智能和推理智能提出的一种智能算法, 它为解决全局优化问题开辟了一条新的途径.

关键词: 了望算法; 全局优化; 智能算法

中图分类号: O224

文献标识码: A

文章编号: 1007-7162(2006)02-0004-11

引言

最近二十多年来, 全局优化技术得到了长足的发展, 模拟退火算法、遗传算法、神经网络算法、禁忌搜索算法(Tabu Search)、蚁群算法(Ant Colony Algorithm)等智能算法得到了广泛而深入的研究, 在解决全局优化问题方面获得了很大的成功. 模拟退火算法^[1-4]是基于物理学中固体物质退火过程所表现出的特性而发展起来的一种智能算法, 属于一种低级的智能, 它的计算量虽然较 Monte Carlo 方法显著减少, 但其全局收敛性能仍然很差. 遗传算法^[5-7]是基于生物本能的智能算法, 是利用生物进化论中优胜劣汰、适者生存的原理及遗传机理而提出的一种智能算法, 其主要优点是并行性和全局空间搜索; 但其计算时间长, 常常出现早熟收敛. 神经网络算法^[8-9]是利用基于联结的生物神经系统协同工作原理而设计的一种智能算法, 其计算简单、快速; 但它的参数鲁棒性差, 容易陷于局部最优, 有时可能收敛于问题的不可行解. 禁忌搜索算法^[10-13]是对人类智能的一种模拟, 具有较强的爬山能力和较好的局部改进能力; 但它对初始解有较强的依赖性, 迭代过程是串行的, 参数的选择不当可能造成早熟收敛. 蚁群算法^[14-15]是模拟蚂蚁群体行为特征而设计的一种基于动物智能的智能算法, 利用正反馈原理, 蚁群算法具有很强的鲁棒性和发现较好解的能力, 是一种基于合作的无监督机制的并行算法; 但是蚁群算法计算时间较长, 参数选择更多地依靠实验和经验、没有一个程序化的方法来确定, 且容易出现迭代停止现象. 以上提及的五大智能算法在解决实际问题时都容易陷于局部最优, 且在迭代过程中不能保证迭代点

收稿日期: 2005-07-11

基金项目: 国家自然科学基金资助项目(60374062); 广东省自然科学基金项目(04009488); 广东省科技计划项目(2004B10101038)

作者简介: 蔡延光(1963-), 男, 博士, 教授, 主要研究方向为组合优化、智能优化、智能决策支持系统.

© 1994-2012 China Academic Journal Electronic Publishing House. All rights reserved. <http://www.cnki.net>

的质量逐步变好.

本文基于了望技术确定群山最高点的常识, 提出解决全局优化问题的了望算法. 了望算法本质上是一种基于监督的并行算法, 在了望管理机制的协调下, 把了望点产生策略和局部寻优算法有机地结合起来, 能在较短的时间内求解全局优化问题. 大量的测试表明, 了望算法具有较高的收敛率和较强的获得问题全部解的能力, 对初始点几乎没有依赖, 参数选择简单, 能够保证在迭代过程中迭代点的质量逐步变好. 大量的对比测试也表明, 在收敛率和全局搜索能力等方面了望算法较遗传算法有一定的优势, 且在大多数情况下了望算法耗时较少. 为了加快了望算法的收敛速度, 引入了望算法的三层次记忆机制, 即基点记忆、了望点记忆、局部寻优记忆.

了望算法模拟了人类的高级行为智能——视觉智能、以及根据视觉信息分析问题的推理智能, 是一种智能算法, 与已知的五大智能算法在原理上有明显的不同, 是解决全局优化问题的一个新尝试.

为了实现方便, 用 C 语言风格书写算法.

1 了望算法

研究全局优化问题

$$\begin{cases} \min F(X), X \in R \subset E^n \\ R = \{X \mid g_i(X) \geq 0, i = 1, 2, \dots, m\}, \end{cases} \quad (1)$$

记 $G(X) = (g_1(X), g_2(X), \dots, g_m(X))^T$.

1.1 基本原理

为了表述方便, 引入几个基本概念.

定义 1 对于问题(1), 设 $R' \subseteq R, X^0 \in R'$,

1) 把 $\min F(X) (X \in R')$ 称为 X^0 处的局部优化问题, 在不引起混淆时简称为局部问题; 求局部问题的解称为局部寻优;

2) 如果在 R' 上 $F(X)$ 是(下)单峰函数, 则称 R' 为 $F(X)$ 的(下)单峰区域, 在不引起混淆时简称为单峰区域.

定义 2 对于问题(1),

1) 了望基准点(简称基点)是可行域 R 中的一个点;

2) 设 $X^b \in R$ 是基点, 基于基点 X^b 的了望点是可行域 R 中的一个点, 在不引起混淆时把基于基点的了望点简称为基点的了望点或了望点; 把 X^b 的全部了望点按照某种标准(例如欧氏距离、Hamming 距离)分类, 称 X^b 的第 k 类了望点为 X^b 的第 k 阶了望点.

定义 3 局部寻优算法是一个基于迭代的优化问题的求解方法, 它具有固有特性: 寻优过程所产生的迭代点必须在初始点所在的单峰区域内.

很早以前, 人类就知道通过了望确定群山最高点的常识. 为了寻找(多峰)群山的最高点, 人们一般在群山中任意选取某座山的一个点作为出发点, 进行局部爬高(即向该座山的最高点攀登), 到达该座山的最高点(相对于群山而言是局部最高点)后, 以此局部最高点作为基点进行了望, 以寻找比基点“更高的点”; 再以此“更高的点”所在的山某个点作为出发点(例如, 直接以此“更高的点”作为出发点), 开始下一轮的局部爬高, ..., 直至找到群山的最高点. 那么, 如何通过了望获得比基点“更高的点”呢? 可以利用下面的常识解决这个问题: 站得越高, 望得越远; 观测目标离得越近, 看得越清楚, 于是可对它的局部进行精细的观察; 观测目标离得越远, 可分辨

的尺寸越大,只能对其进行粗略的观察.选择基点的若干个了望点作为“更高的点”的候选点:在基点附近可稠密地取一些了望点,而在离基点远的地方了望点可取得稀疏一些.人们站在基点对了望点进行目测,当了望点在水平视线的上方,则为了望点高于基点(如果忽略观测者自身的高度),“更高的点”就是该了望点.

了望算法是利用以上常识而设计的一种求解全局优化问题的智能算法.了望算法由了望管理机制(主要负责算法进程控制与协调)、了望点产生策略(负责产生了望点)、局部问题构造与求解机制3部分组成,其求解全局优化问题的基本思路是:1)由了望管理机制确定基点;2)由了望点产生策略产生基点的了望点;3)由了望管理机制按照一定的标准选择了望点;4)构造所有被选定的了望点的局部问题并用局部寻优算法求解这些局部问题;5)在获得所有被选定的局部问题的解后,由了望管理机制确定下一个基点,进行新一轮的迭代,直至算法终止条件出现,把当前最好可行解作为问题的解.

针对以上基本思路,1.2节将讨论其中第1)、3)、5)项的实现问题(隐含在算法1中),第2节讨论其中第2)项的实现问题,第3节讨论其中第4)项的实现问题.

1.2 算法

算法1是了望算法的一种实现形式,用于求解问题(1),其核心部分是:

①外层循环:控制了望算法的进程,选择基点 X^B .当基点数超过预先设定的最大值或者允许了望标志等于False时,算法终止运行,输出最优解.

②第2层循环:通过调用算法2,产生基于基点 X^B 的各阶了望点.

③内层循环:选择了望点进行局部寻优、置允许了望标志的值.

算法1 求解问题(1)的了望算法

输入:目标函数 $F(X)$ 及约束条件 $G(X) \geq 0$;

控制参数设置:

Max_Global_Outlook= 正整数; //(允许考虑的基点个数,即最大基点数,该参数为预先设定.)

Max_Local_Outlook= 正整数; //(基点的了望点的最大阶数,Max_Local_Outlook应该取得足够大,使得对于该基点,在 R 中不存在(或者可以忽略)基于它的大于Max_Local_Outlook阶的了望点.)

初始:给定初始可行解 $X^0 \in R$;

$X^B = X^0$; //基点

$X^{\min} = X^B$; //到目前为止的最好可行解

Old $X^{\min} = X^{\min}$; //临时变量

Local $X^{\min} = X^B$; //局部问题的解

Outlook_Flg= True; //允许了望标志, = True 允许了望, = False 不允许了望.

Global_Outlook= 0; //基点计数

Local_Outlook= 0; //了望阶数

While Outlook_Flg= True 且 Global_Outlook < Max_Global_Outlook //外层循环开始

{

Old $X^{\min} = X^{\min}$;

Outlook_Flg= False;

© 1994-2012 China Academic Journal Electronic Publishing House. All rights reserved. <http://www.>

如果 $\text{Global_Outlook} = 0$ 则 $\text{Local_Outlook} = 0$, 否则 $\text{Local_Outlook} = 1$; //(初始时因为 $X^B = X^0$, 故需要求解 X^B 处的局部问题; 但以后的 X^B 本身是前面某个局部问题的解, 因此不必求解 X^B 处的局部问题.)

While $\text{Local_Outlook} \leq \text{Max_Local_Outlook}$ //第 2 层循环开始

{

Set_Outlook_Points = ϕ ;

Generate_Outlook_Point(X^B , Local_Outlook, Set_Outlook_Points); //(产生了望点, 见第 2 节.)

While Set_Outlook_Points $\neq \phi$ //内层循环开始

{

取集合 Set_Outlook_Points 中的第一个元素 X^0 ;

如果 $F(X^0) \leq F(X^B)$, 则 //片段 1, 选择了望点进行局部寻优

{

$\text{Local}X^{\min} = \text{Local_Search}(X^0)$; //(片段 2. Local_Search(X): 局部寻优算法. 其功能是寻找 X 处的局部问题的解(通常把 X 作为初始点), 返回值为局部问题的解. 见第 3 节的讨论.)

如果 $F(\text{Local}X^{\min}) \leq F(X^{\min})$ 且 $\text{Local}X^{\min} \neq \text{Old}X^{\min}$ //片段 3

{

$X^{\min} = \text{Local}X^{\min}$;

Outlook_Flg = True;

};

};

在集合 Set_Outlook_Points 中删除 X^0 ;

}; //内层循环结束

Local_Outlook = Local_Outlook + 1; //处理更高阶的了望点

}; //第 2 层循环结束

Global_Outlook = Global_Outlook + 1;

$X^B = X^{\min}$; // $X^{\min}$ 作为新的基点

}; //外层循环结束

输出最优解 $X^{\min}$;

结束.

2 了望点产生策略

1.1 节所描述的关于获得了望点的常识(即在基点附近可稠密地取一些了望点, 而在离基点远的地方了望点可取得稀疏一些)可以有多种实现形式, 即我们可以设计多种了望点产生策略. 本节仅讨论产生了望点的两个策略: 方体了望点产生策略和球面了望点产生策略.

设 $X^B = (x_1^B, x_2^B, \dots, x_n^B)^T$ 是基点, 下面讨论在 X^B 处产生第 k 阶了望点的方体了望点产生策略和球面了望点产生策略.

2.1 方体了望点产生策略

方体了望点产生策略是在以 X^B 为中心的棱长为 $2r_k$ 的 n 维立方体表面和可行域 R 的交集中选取 X^B 的第 k ($k = 0, 1, 2, \dots$) 阶了望点, 其中 $r_k = H(h, k, X^B, F, G)$, 它是 h, k, X^B, F, G 的函数, $h(>0)$ 为预先设定的一个常数(称为基本了望步长), 且 $r_0 = 0 < r_1 < r_2 < \dots$; 当 $k=0$ 时, X^B 是唯一的了望点.

以 $n=2$ 为例说明由方体了望点产生策略所产生的第 k 阶了望点的几何意义. 例如, 第 k 阶了望点取自于 $\{(x_1^B \pm r_i, x_2^B \pm r_k)^T, (x_1^B \pm r_k, x_2^B \pm r_j)^T \mid i = 0, 1, 2, \dots, k; j = 0, 1, 2, \dots, k-1\} \cap R$ (见图1).

下面讨论 r_k 的构造. 主要考虑 r_k 是 h, k 的函数的情形, 可以用等距法、算术增距法、Fibonacci 增距法、几何增距法确定 r_k .

1) 等距法

当可行域较小时, 取

$$r_k = kh, k = 0, 1, 2, \dots \quad (2)$$

此时 $r_{k+1} - r_k = h$ ($k \geq 0$), 即 $r_{k+1} - r_k$ ($k = 0, 1, 2, \dots$) 是相等的.

2) 算术增距法

当可行域较大时, 取

$$r_k = \frac{k(k+1)}{2}h, k = 0, 1, 2, \dots \quad (3)$$

此时 $r_{k+1} - r_k = (k+1)h$ ($k \geq 0$), 即 $\{r_{k+1} - r_k\}$ 是一个公差为 h 的算术级数.

3) Fibonacci 增距法

当可行域很大时, 取

$$r_0 = 0, r_k = F_{k-1}h + r_{k-1}, k = 1, 2, \dots \quad (4)$$

其中 F_k 是 Fibonacci 数, $F_0 = 1, F_1 = 1, F_k = F_{k-1} + F_{k-2}$. 易见, $r_{k+1} - r_k = \frac{F_k}{F_{k-1}}(r_k - r_{k-1}) = F_k h$ ($k \geq 1$), 即 $\{r_{k+1} - r_k\}$ 是一个 Fibonacci 级数.

4) 几何增距法

当可行域很大时, 取

$$r_k = \frac{(1+\delta)^k - 1}{\delta}h, k = 0, 1, 2, \dots \quad (5)$$

其中 $\delta > 0$ 是预设的常数. 易见 $r_{k+1} - r_k = (1+\delta)(r_k - r_{k-1}) = h(1+\delta)^k$ ($k \geq 1$), 即 $\{r_{k+1} - r_k\}$ 是一个公比为 $1+\delta$ 的几何级数.

周知 $\frac{F_k}{F_{k-1}} \rightarrow \frac{1+\sqrt{5}}{2}$ ($k \rightarrow \infty$) $\approx (1+0.618)$, 如果取 $\delta = 0.618$, Fibonacci 法就可近似地归结为几何增距法的一种特殊形式.

5) r_k 与 X^B 有关

例如, 当 $\|X^B\| \neq 0$ 时, 取

$$r_0 = 0, r_k = \|X^B\|(1+\delta)^k, k = 1, 2, \dots \quad (6)$$

其中 $\delta > 0$ 是预设的常数, $\|X^B\|$ 是 X^B 的范数(例如取 $\|X^B\| = \sqrt{\sum_{i=1}^n X_i^{B^2}}$, 或者取 $\|X^B\| =$

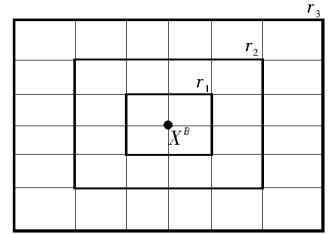


图1 用方体了望点产生策略产生了望点

$\sum_{i=1}^n \|X_i^B\|$). 易见, $r_{k+1} - r_k = \delta \|X^B\| (1 + \delta)^k = (1 + \delta)(r_k - r_{k-1}) (k \geq 2)$.

如果 $\|X^B\| = 0$, 则式(6)中的 $\|X^B\|$ 用一个较小的正数代替.

根据以上讨论, 可获得方体了望点产生策略产生了望点的算法.

算法 2 用方体了望点产生策略产生了望点

Function Generate_Outlook_Point(X^B , Local_Outlook, Set_Outlook_Points)

/* 功能与参数说明: 以 X^B 为基点产生其第 Local_Outlook 阶的全部了望点, 了望点保存在集合

Set_Outlook_Points 中. */

{for ($i_1 = -\text{Local_Outlook}; i_1 \leq \text{Local_Outlook}; i_1++$)

{ $x_1 = x_1^B + \text{sign}(i_1) r_{|i_1|}$; //sign(x) 为符号函数

for ($i_2 = -\text{Local_Outlook}; i_2 \leq \text{Local_Outlook}; i_2++$)

{ $x_2 = x_2^B + \text{sign}(i_2) r_{|i_2|}$;

for ($i_3 = -\text{Local_Outlook}; i_3 \leq \text{Local_Outlook}; i_3++$)

{ $x_3 = x_3^B + \text{sign}(i_3) r_{|i_3|}$;

... ..

for ($i_n = -\text{Local_Outlook}; i_n \leq \text{Local_Outlook}; i_n++$)

{ $x_n = x_n^B + \text{sign}(i_n) r_{|i_n|}$;

如果 $|i_1|, |i_2|, \dots, |i_n|$ 中至少有一个等于 Local_Outlook, 且 $(x_1, x_2, \dots, x_n)^T \in$

R , 则 Set_Outlook_Points = Set_Outlook_Points $\cup \{(x_1, x_2, \dots, x_n)^T\}$;

};

};

};

};

};

2.2 球面了望点产生策略

球面了望点产生策略是在 n 维球面 $(x_1 - x_1^B)^2 + (x_2 - x_2^B)^2 + \dots + (x_n - x_n^B)^2 = r_k^2$ 和可行域 R 的交集中选取 X^B 的第 k ($k = 0, 1, 2, \dots$) 阶了望点, 其中 r_k 与方体了望点产生策略的意义及确定方法完全相同. 因为 $r_0 = 0 < r_1 < r_2 < \dots$, 故在球面了望点产生策略下, 同阶了望点与 X^B 的距离相等, 较高阶了望点离 X^B 较远.

以 $n = 2$ 为例说明由球面了望点产生策略所产生的第 k 阶了望点的几何意义. 如果希望至多取 q 个第 k 阶了望点, 则不妨在 X^B 出发的 q 条射线 (q 条射线一般是均匀分布的) 与圆周 $(x_1 - x_1^B)^2 + (x_2 - x_2^B)^2 = r_k^2$ 的交点中选取可行点作为了望点 (见图 2), 这里, $q = q(h, k, X^B, F, G)$.

类似于算法 2, 可获得球面了望点产生策略产生了望点的算法. 限于篇幅, 不再写出.

3 局部问题构造与局部寻优算法

了望算法的提出也受到下面设想的推动: 为了求解目标函数是多峰函数的全局优化问题,

先把 R 划分为数量尽量少的两两互不相交的单峰区域(或者,把 R 划分为两两互不相交的单峰区域,并使每个单峰区域尽量大),每个单峰区域对应一个局部问题,这样把全局优化问题分解为一系列目标函数是单峰函数的子问题(即局部问题);然后用目标函数是单峰函数的优化问题的求解方法获得局部问题的解;最后通过简单地比较各局部问题的解就可获得全局优化问题的解.

但是,以上设想直接实现起来非常困难.了望算法充分地利用局部寻优算法的固有特性构造并求解局部问题,既不需要明确地界定单峰区域也不需要准确地划分 R ,结合了望点产生策略,解决全局优化问题,是以上设想的一个改良.根据局部寻优算法的固有特性,显然只要给出局部寻优的初始点(为了节约计算时间,通常把了望点作为初始点),就可以获得局部问题的解;只要给出局部寻优的初始点,局部问题随之确定.

局部寻优算法可以通过对求解目标函数是单峰函数的优化问题的方法(如一维无约束优化的黄金分割法、成功-失败法、牛顿法,多维无约束优化的拟牛顿法、共轭梯度法、DFP法,以及有约束优化问题罚函数法、SWIFT法、可行方向法、线性逼近法等)作一些必要的改造得到,使其最大限度满足局部寻优算法的固有特性.按照这个思路,我们对以上提及的一些算法进行了改造.例如局部牛顿法是在牛顿法的基础上增加了对边界和拐点等的特别处理而获得的局部寻优算法,局部DFP法是在DFP法的基础上增加了对边界等的特别处理而获得的局部寻优算法.限于篇幅,改造过程从略.

4 了望算法的记忆机制

为了避免在搜索过的区域进行重复搜索,提高算法的收敛速度,现引入了望算法的三层次记忆机制,即基点记忆(第一层)、了望点记忆(第二层)、局部寻优记忆(第三层).

1) 基点记忆

基点记忆(MB)记忆全部基点.初始, $MB = \emptyset$; 如果 X^B 为基点,则 $MB = MB \cup \{X^B\}$. 如果实行基点记忆,则将算法1的片段3所对应的条件改为: 如果 $F(\text{Local}X^{\min}) \leq F(X^{\min})$ 且 $\text{Local}X^{\min} \notin MB$. 这样,可以避免在迭代过程中重复地把一个点作为基点.

2) 了望点记忆

了望点记忆(MO)记忆全体构造过局部问题并实施过局部寻优的了望点.初始, $MO = \emptyset$; 如果 X^0 是某个基点的了望点,且求解过 X^0 处的局部问题,则 $MO = MO \cup \{X^0\}$. 如果实施了望点记忆,则将算法1的片段1所对应的条件改为: 如果 $F(X^0) \leq F(X^B)$ 且不存在 $X' \in MO$ 使 $\|X' - X^0\| \leq \varepsilon$ ($\varepsilon \geq 0$ 为预设的精度). 这样,对任意点及其附近的点总共至多只能构造一次局部问题.

3) 局部寻优记忆

局部寻优记忆(ML)记忆局部寻优过程所经历的部分迭代点.初始, $ML = \emptyset$; 如果 X^l 是局部寻优过程所获得的一个迭代点,且不存在 $X' \in ML$ 满足 $\|X' - X^l\| \leq \varepsilon$ ($\varepsilon \geq 0$ 为预设的精度),则 $ML = ML \cup \{X^l\}$; 否则终止此局部寻优,因为如果 $F(X)$ 是平滑的、不具有突变性,继续进行局部寻优的话,则所得到的迭代点的质量一般会比以前有实质性更好.此外,当有局部寻优记忆时,算法1的片段1所对应的条件也可改为: 如果 $F(X^0) \leq F(X^B)$ 且不存在 $X' \in ML$ 使 $\|X' - X^0\| \leq \varepsilon$ ($\varepsilon \geq 0$ 为预设的精度).

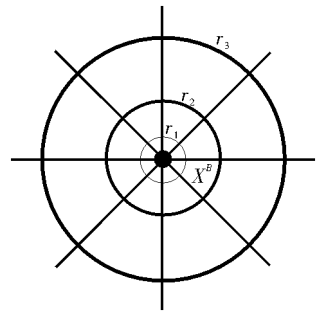


图2 用球面了望点产生策略产生了望点

MB、MO、ML 体现了了望算法 3 个层次的记忆, 显然有 $MB \subseteq ML, MO \subseteq ML$, 根据需要选择其中一个或多个记忆. 为了实现对 MB、MO、ML 中的元素进行快速查找(如二分查找), 在迭代过程中应该同时对它们的元素进行排序(如二分插入排序).

5 测试结果与评述

用了望算法对一些著名的基准问题(Benchmark Problem, 如见文献[16]) 进行大量的反复测试, 结果令人满意. 为了进一步了解了望算法的性能, 我们用遗传算法做对比测试. 限于篇幅, 这里仅列出对 4 个著名的基准问题所作的对比测试结果, 并进行了简要的评述.

所有算法使用 C 语言编程实现, 并在赛扬 2. 4G 台式机上执行.

问题 1 $F_1(x) = x^3 + 3x^2 - 9x, x \in [- 5, 5]$.

该问题的解有两个: $x = 1, x = - 5. \min F_1(x) = - 5$. 周知, 非智能算法如成功一失败法、0. 618 法很难获得该问题的解.

问题 2 Rosenbrock 函数: $F_2(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2, x_i \in [- 10, 10], i = 1, 2$.

最优解 $(1, 1)^T, \min F_2(x_1, x_2) = 0$.

问题 3 六驼峰一驼背(Six-Hump Camel-Back) 函数: $F_3(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + x_1^6/3 + x_1x_2 - 4x_2^2 + 4x_2^4, x_i \in [- 5, 5], i = 1, 2$.

该问题的解有两个: $(0. 089842, - 0. 712656)^T, (- 0. 089842, 0. 712656)^T, \min F_3(x_1, x_2) = - 1. 0316285$.

问题 4 球体模型(Sphere Model): $F_4(x_1, x_2, \dots, x_n) = x_1^2 + x_2^2 + \dots + x_n^2, x_i \in [- 100, 100], i = 1, 2, \dots, n$.

取 $n = 8$. 最优解 $x_i = 0(i = 1, 2, \dots, n), \min F_4(x_1, x_2) = 0$.

了望算法的参数设置见表 1.

表 1 了望算法的参数设置

问题	参数设置
1	1) 最大基点数: [4, 6] 中的随机整数. 2) 基点的了望点的最大阶数: 11. 3) 初始解: 可行域中的随机数. 4) 了望点产生策略: 方体了望点产生策略, 其中基本了望步长为 [0. 9, 1. 1] 中的随机数, r_k 用等距法确定. 5) 局部寻优算法: 局部牛顿法, 精度为 $[10^{-4}, 10^{-6}]$ 中的随机数. 6) 记忆机制: 基点记忆、了望点记忆.
2	1) 基点的了望点的最大阶数: 22. 2) 局部寻优算法: 局部 DFP 法, 精度为 $[10^{-4}, 10^{-6}]$ 中的随机数. 3) 其余参数同问题 1 的设置.
3	1) 基点的了望点的最大阶数: 11. 2) 其余参数同问题 2 的设置.
4	1) 最大基点数: [2, 4] 中的随机整数. 2) 基点的了望点的最大阶数: 4. 3) 了望点产生策略: 球面了望点产生策略, 其中基本了望步长为 [20, 50] 中的随机数, 每阶了望点的数目为 [4, 6] 中的随机整数(在对应的球面上近似均匀选取), r_k 用算术增距法确定. 4) 局部寻优算法: 局部 DFP 法, 精度为 $[10^{-4}, 10^{-6}]$ 中的随机数. 5) 其余参数同问题 1 的设置.

遗传算法的参数设置: 二进制编码方法, 种群数为 80, 交叉概率为 0.6, 变异概率为 0.001. 收敛准则: 4 个问题的遗传代数分别为 200、1000、300、2000(实际上, 我们对这 4 个问题中的每一个分别做了 100、200、300、500、1000、2000、5000 代的遗传算法运行测试, 从时间耗费和收敛率方面综合考虑, 前述所列遗传代数于相应问题为最有利的配置).

测试结果及评述见表 2, 其中: 收敛率= 收敛次数/ 测试次数.

表 2 了望算法和遗传算法的对比测试结果与评述

问题	算法	测试 次数	平均耗 时/s	收敛 率/%	评述
1	了望算法	300	0.23	100	了望算法平均耗时只有遗传算法的 1/2, 两个算法的收敛率接近. 但是, 了望算法每次测试都能同时获得两个解, 而遗传算法只有 66% 的概率同时获得两个解.
	遗传算法	500	0.44	99	
2	了望算法	300	12.36	100	了望算法平均耗时较多, 大约是遗传算法的 4 倍. 主要原因是作为局部寻优算法的局部 DFP 法在解该问题的局部问题时效率不高. 统计表明, 每次测试执行局部寻优算法的次数是问题 1 的 4.6 倍左右; 这就是说, 若该问题的局部寻优效率达到问题 1 的局部寻效率的 1/3, 则平均耗时应与遗传算法接近.
	遗传算法	300	3.19	40	
遗传算法收敛率太低, 不到了望算法的 1/2.					
3	了望算法	300	0.93	100	了望算法的平均耗时稍少, 两个算法的收敛率接近. 了望算法同时获得两个解的概率为 99.6%; 而遗传算法所有收敛测试都只能获得一个解, 且每个解单独出现的机会相等.
	遗传算法	500	1.05	98	
4	了望算法	300	0.82	100	了望算法平均耗时只有遗传算法的 1/5, 了望算法的收敛率比遗传算法高 17%.
	遗传算法	300	4.15	83	

下面结合测试结果, 对了望算法作一些简要的讨论.

1) 已知的测试表明, 了望算法在取最大基点数 4 时能够求解最优解的个数至多为 2 的所有测试问题(包括本文的 4 个基准问题). 显然, 最大基点数设置得越大, 获得全部解的可能性就越大. 然而, 大量的测试表明, 最大基点数增大到一定程度后, 其继续增大并不会导致算法的平均耗时的明显增大. 例如, 问题 1 取最大基点数 4 和最大基点数 6 的平均耗时相差只有 1 毫 s 左右. 事实上, 通过分析算法 1 就会发现: 获得了问题的全部解后到算法结束的一段运行所耗时间很少.

2) 在表 1 中, 基点的了望点的最大阶数的值是照顾到基点、了望点产生策略及基本了望步长的所有组合情况而设定的, 故在大多数情况下稍嫌大了一些. 为了减少算法 1 的第 2 层循环的循环次数、提高算法的收敛速度, 根据该参数的含义, 可以对算法 1 稍作修改: 在第 2 层循环开始的前 1 行, 根据基点、了望点产生策略、基本了望步长和可行域的结构动态设置基点的了望点的最大阶数为一个尽可能小的值(这一点不难做到, 而且动态设置的值大多数情况下比表 1 设置的值要小).

3) 基本了望步长越小, 了望算法对可行域将搜索得越仔细, 但耗时增加; 反之, 耗时减少, 但可能遗漏重要的搜索区域. 因此需要对基本了望步长进行合理的设置. 基本了望步长设置原则: 当可行域较小时, 基本了望步长应取得较小(如问题 1~ 3), 反之应取得较大(如问题 4). 当然, 也可以在算法执行过程中对每个基点动态地确定基本了望步长.

4) 在同一精度下对了解算法和遗传算法进行时间耗费比较似乎更加客观。但是, 我们的测试结果表明, 遗传算法有些测试进行了 10 多个小时而最好解仍未达到事先预定的精度要求, 实际上陷入局部最优, 造成无法确定该测试的时间耗费。本节所列的 4 个问题都曾出现过这种久不收敛的测试。

6 结论

了解算法的原理和大量的测试表明, 它具有如下特点:

- 1) 收敛率高。已知的测试表明, 了解算法收敛率接近 100%, 高于遗传算法。
- 2) 全局搜索能力强。已知的测试表明, 了解算法获得问题全部解的概率超过 0.99, 较遗传算法有一定的优势。了解算法利用了望点产生策略使迭代进程逃出局部最优, 保证搜索的多样性, 进而实现全局搜索, 能保证迭代过程中所得到的迭代点的质量逐步变好。
- 3) 速度快。在大多数情况下, 了解算法的时间耗费比遗传算法要少。特别是了解算法的记忆机制的应用能极大地减少时间耗费。
- 4) 鲁棒性。了解算法的鲁棒性表现在 3 个方面: 首先是它所能解决的问题的广泛性, 只要问题形如式 (1); 其次是它对初始点几乎没有依赖性; 最后是它的控制参数选择简单。
- 5) 并行性。了解算法并行化非常容易, 例如用某个处理机(作为主处理机)对算法进程进行控制和协调、产生了望点等, 而其它处理机构造了望点处的局部问题并用局部寻优算法求解之。
- 6) 经过一些简单的改造, 了解算法能有效地利用目标函数是单峰函数的优化问题的方法进行局部寻优, 而后者经过半个多世纪的研究获得了不少的快速算法。这一优势是基本遗传算法所不具有的。

致谢: 感谢邹谷山同志所作的部分测试工作。

参考文献:

- [1] Metropolis N, Rosenbluth A W, Rosenbluth M N, Teller A H, Teller E. Equations of state calculations by fast computing machines[J]. J Chemical Physics, 1953, 23: 1087-1091.
- [2] Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by simulated annealing[J]. Science, 1983, 220: 671-680.
- [3] 蔡延光, 钱积新, 孙优贤. 多重运输调度问题的模拟退火算法[J]. 系统工程理论与实践, 1998, 18(10): 11-15.
- [4] Ji M, Tang H. Application of chaos in simulated annealing[J]. Chaos, Solitons and Fractals, 2004, 21(4): 933-941.
- [5] Holland J H. Adaptation in Nature and Artificial Systems[M]. Ann Arbor MI: University of Michigan Press, 1975.
- [6] 蔡延光, 钱积新, 孙优贤. 多重运输调度问题的遗传算法与遗传局部搜索[J]. 系统工程理论与实践, 1997, 17(12): 101-107.
- [7] Baker B M, Ayechew M A. A genetic algorithm for the vehicle routing problem[J]. Computers and Operations Research, 2003, 30(5): 787-800.
- [8] Hopfield J J, Tank D W. Neural computation of decision in optimization problems[J]. Biological Cybernetics, 1985, 52: 141-152.
- [9] Zhang Y. Global exponential convergence of recurrent neural networks with variable delays[J]. Theoretical Computer Science, 2004, 312(23): 281-293.
- [10] Glover F. Heuristic for integer programming using surrogate constraints[J]. Decision Science, 1977, 8: 156-166.
- [11] 蔡延光, 钱积新, 孙优贤. 带时间窗的多重运输调度问题的自适应 Tabu Search 算法[J]. 系统工程理论与实践, 2000, 20(12): 42-50.

- [12] Caricato P, Ghiani G, Grieco A, Guerriero E. Parallel tabu search for a pickup and delivery problem under track contention[J]. *Parallel Computing*, 2003, 29(5): 631-639.
- [13] Ho S C, Haugland D. A tabu search heuristic for the vehicle routing problem with time windows and split deliveries[J]. *Computers & Operations Research*, 2004, 31(12): 1947-1964.
- [14] Colomi A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies[C]// *Proc. of the First European Conf. On Artificial Life*, Paris: Elsevier Publishing, 1991: 134-142.
- [15] Badr A, Fahmy A. A proof of convergence for ant algorithms[J]. *Information Sciences*, 2004, 160(14): 267-279.
- [16] 王凌. 智能优化算法及其应用[M]. 北京: 清华大学出版社, 2001.

Outlook Algorithm for Global Optimization

CAI Yan-guang¹, QIAN Ji-xin², SUN Yong-xian²

(1. Faculty of Automation, Guangdong University of Technology, Guangzhou 510090, China

2. National Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China)

Abstract: Outlook algorithm is presented to solve global optimization problems in this paper. Based on common knowledge that one decides the highest point of mountains by outlook, by employing supervision mechanism of outlook, strategies of generating outlook points and mechanisms of constructing and solving local problems, outlook algorithm can solve any global optimization problem in a relatively short time. A large number of tests show that outlook algorithm is of higher convergence ratio, stronger capacity to obtain all solutions of global optimization problems, little dependence on initial solution and simplicity in deciding its control parameters. It can be ensured that the quality of iterative points will gradually improve in the iterative process of outlook algorithm. The three-level memory mechanism of outlook algorithm greatly increases its convergence rate. A large number of contrast tests also show that outlook algorithm has advantage over genetic algorithm in convergence ratio and capacity of global search, and spends less time than genetic algorithm in most cases. Since outlook algorithm simulates human behavioral and inferential intelligence, it exploits a brand new way to solve global optimization problems.

Key words: outlook algorithm; global optimization; intelligent algorithm