文章编号:1007-6735(2012)04-0333-04

复杂系统可靠性优化的混合万有引力搜索算法求解

刘 勇[1,2], 马 良[1]
(1.上海理工大学 管理学院,上海 200093; 2.盐城工学院 基础教学部,盐城 224051)

摘要: 复杂系统可靠性优化问题是一类有约束限制且目标函数具有多个局部极值的非线性优化问题.为求解该类问题,提出了一种混合万有引力搜索算法的求解方法.算法利用基于万有引力定律的寻优机制指导群体进行全局搜索,并采用序列二次规划算法进行局部搜索,避免基本万有引力搜索算法陷入局部最优,改善优化性能,加快寻优速度.通过实例计算,并与蚁群优化算法、微粒群算法、蜂群算法和基本万有引力搜索算法等进行比较,验证了算法的可行性和有效性.

关键词: 系统可靠性; 万有引力搜索算法; 序列二次规划; 优化
中图分类号: N 94; O 221.2 **文献标志码:** A

Optimization of Complex System Reliability Based on Hybrid Gravitational Search Algorithm

LIU Yong[1,2], MA Liang[1]
(1. *Business School, University of Shanghai for Science and Technology, Shanghai 200093, China*;
2. *Department of Fundamental Teaching, Yancheng Institute of Technology, Yancheng, 224051, China*)

Abstract: Reliability optimization problem of complex system is a nonlinear optimization problem under the constraint conditions and in the case of that the objective function has a large number of local extreme values. Hybrid gravitational search algorithm was proposed to solve the model. In the algorithm, a searching mechanism based on the law of gravitation was used to find the global optimal solution. Sequential quadratic programming was employed as a local search method to avoid being trapped into local optimum in the basic gravitational search algorithm. The optimization performance is improved and the search speed is accelerated in the proposed algorithm. Computations on some pratical examples and comparisons with ant colony optimization algorithm, particle swarm optimization algorithm, artificial bee colony algorithm and basic gravitational search algorithm demonstrate the algorithm is feasible and effective.

Key words: *system reliability*; *gravitational search algorithm*; *sequential quadratic programming*; *optimization*

收稿日期: 2012-02-14
基金项目: 国家自然科学基金资助项目(70871081);上海市重点学科建设资助项目(S30504)
作者简介: 刘 勇(1982-),男,博士研究生.研究方向:智能优化、系统工程.E-mail:liuyong.seu@163.com
马 良(联系人),男,教授.研究方向:智能优化、系统工程.E-mail:maliang@usst.edu.cn

可靠性问题是系统设计、研究和运行过程中的一个重要因素，可靠性的高低直接影响系统的性能.若可靠性达不到相应的指标要求，系统越复杂出故障的可能性越大，造成的损失也越大[1].因此，如何对复杂系统可靠性进行优化已成为一个重要的研究课题.所谓复杂系统可靠性优化是指在满足一定的可靠性指标要求下使投资费用最小或在一定的资源约束条件下使系统的可靠度达到最大.由于其目标函数和约束条件都是非线性的，而且目标函数通常都是具有多个局部极值的函数，这些给问题的求解带来了一定的困难.采用传统的优化算法求解往往达不到全局最优，解决此类问题难度较大.近年来，智能算法在求解很多复杂困难的优化问题中表现出良好的性能，已成为一个研究热点[2-3].目前用于复杂系统可靠性优化的智能算法有模拟退火算法[4]、遗传算法[5]、蚁群算法[6]等.这些算法在一定程度上可求解复杂系统可靠性优化问题，但算法都存在着各自的局限性，求解质量有待进一步改进.因此，探寻有效的求解方法颇为重要.

基本万有引力搜索算法是一种新的基于群体智能的优化算法，由 Rashedi 等[7]在 2009 年提出.算法具有操作简单，参数设置少等特点，并在优化 benchmark 函数时表现出较好的性能.但研究表明，万有引力搜索算法全局搜索能力较强，可局部搜索优化能力较弱，易早熟收敛.为解决这一不足，本文将序列二次规划算法引入算法中，提出一种混合万有引力搜索算法.利用万有引力搜索算法寻找全局最优解，并利用序列二次规划算法进行局部搜索，实现全局探索和局部开发能力的平衡，避免基本万有引力搜索算法陷入局部极值.将算法用于求解复杂系统可靠性优化问题，通过数值实验和与现有智能算法的比较，表明算法可行有效.

1 数学模型

考虑一类典型的复杂系统可靠性优化问题[4-6]，对于一个给定的系统，在满足一定的可靠性约束条件下，通过为每一个子系统或部件选择合适的可靠性，使系统费用最小.

$$\min \quad C_s$$
$$\text{s.t.} \quad \begin{cases} R_{i,\min} \leqslant R_i \leqslant 1 (i = 1,2,\cdots,n) \\ R_{s,\min} \leqslant R_s \leqslant 1 \end{cases} \tag{1}$$

式中，C_s 为系统费用；R_i，$R_{i,\min}$ 为部件可靠性和部件可靠性下界；R_s，$R_{s,\min}$ 为系统可靠性和系统可靠性下界.

上述模型中，目标函数和约束函数均为非线性，而且目标函数具有多个局部极值，给寻找全局最优解带来了困难.解决这类问题，智能优化算法效果显著.因此，本文设计了一种混合万有引力搜索算法的求解方法.

2 求解方法

2.1 万有引力搜索算法

万有引力搜索算法（gravitational search algorithm，GSA）是由 Rashedi 等提出的，是一种模拟万有引力规律的智能优化算法.算法将优化问题的解视为一组在空间运行的物体，物体之间通过万有引力作用相互吸引，物体运动遵循动力学规律，万有引力的作用使得物体朝着质量最大的物体移动，而最大质量的物体占据最优位置，从而可求出优化问题的最优解.算法通过个体间的万有引力相互作用实现优化信息的共享，引导群体向最优解区域搜索.进一步研究发现，在迭代的早期，算法具有较好的寻优性能，但在后期算法会出现在局部最优解附近“振荡”的现象，趋同化严重，易早熟收敛.

2.2 序列二次规划

序列二次规划（sequential quadratic programming，SQP）最初由 Wilson 于 1963 年提出，是解无约束优化问题的牛顿法和拟牛顿法对约束优化问题的推广，具体是通过求解一系列二次规划的子问题逐步得到原问题的最优解[8].目前 SQP 算法是非线性约束优化研究中一个十分活跃的领域.SQP 具有保持局部超一次收敛等特性，但不足之处是在优化有多个局部极值的目标函数时，对初始点的选取非常敏感，若选择不当会极大影响算法的性能[9-10].

2.3 混合算法

基于万有引力搜索算法和序列二次规划各自的特点，提出一种混合万有引力搜索算法（hybrid gravitational search algorithm，HGSA）.算法首先利用万有引力搜索算法进行全局搜索，保存当前的最优位置，利用该最优位置作为 SQP 算法迭代的初始点，利用 SQP 进行局部搜索，有利于算法跳出局部极值，使算法全局搜索和局部开发的能力达到平衡.同时在当前最好位置附近进行局部搜索，找到全局最优点的可能性增大，加快算法寻优速度.

算法具体步骤如下：

第 1 步　初始化

考虑由 N 个物体组成的系统,每个物体定义在 D 维搜索空间中,物体的位置代表优化问题的解.第 i 个物体的位置定义为

$$X_i = (x_i^1,\cdots,x_i^d,\cdots,x_i^D), i = 1,2,\cdots,N \quad (2)$$

式中,x_i^d 为第 i 个物体在第 d 维空间的位置.

第 2 步　计算物体的质量

在时刻 t 时,物体的质量定义为

$$q_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \quad (3)$$

$$M_i(t) = q_i(t)/\sum_{j=1}^{N} q_j(t) \quad (4)$$

式中,$fit_i(t)$,$M_i(t)$分别为在时刻 t 时第 i 个物体的函数值和质量;$best(t)$,$worst(t)$分别为在时刻 t 时所有物体中最优函数值和最差函数值,对最小化问题其定义为

$$best(t) = \min_{j\in\{1,2,\cdots,N\}} fit_j(t) \quad (5)$$

$$worst(t) = \max_{j\in\{1,2,\cdots,N\}} fit_j(t) \quad (6)$$

第 3 步　计算万有引力

物体 i 和 j 之间的万有引力定义为

$$F_{ij}^d(t) = G(t)\frac{M_j(t)M_i(t)}{R_{ij}(t)+\varepsilon}(x_j^d(t) - x_i^d(t))(d = 1,2,\cdots,D) \quad (7)$$

式中,$G(t)$为在时刻 t 时万有引力常数,在算法中随着 t 逐渐减小;$R_{ij}(t)$为物体 i 和 j 之间欧式距离;ε 为一常数,防止分母为零.

第 4 步　计算合力

物体 i 所受的合力为

$$F_i^d(t) = \sum_{j=1,j\neq i}^{N} rand_j F_{ij}^d(t) \quad (8)$$

式中,$rand_j$ 为在[0,1]之间的一个随机数.

第 5 步　计算加速度

根据牛顿运动定律,在时刻 t 物体 i 在第 d 维的加速度为

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \quad (9)$$

第 6 步　更新速度和位置

物体 i 速度和位置的更新方程为

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \quad (10)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (11)$$

式中,$rand_i$ 表示在[0,1]之间的一个随机数.

第 7 步　以当前群体的最优位置 $Lbest$ 作为初始点,利用 SQP 方法进行搜索.

第 8 步　如果没有达到最大迭代次数,转第 2 步;否则,停止计算,输出当前的最优解.

由上述算法步骤很容易得到算法的时间复杂度,设群体规模为 N,问题规模为 D,最大迭代次数为 T.基本万有引力搜索算法的主要操作是计算质量、计算引力、更新加速度及速度和位置,时间复杂度为 $O(T\times N\times D)$.混合万有引力搜索算法增加的操作是利用 SQP 进行局部搜索,设其最大迭代次数为 Maxiter,则混合算法的时间复杂度为 $O(T\times N\times D)+O(T\times \text{Maxiter})$,第 2 项的时间复杂度比第 1 项小的多,因此在计算时间方面混合算法不会比基本万有引力搜索算法有显著增加.

3　数值实验

采用一种典型的复杂系统——非串联-并联系统[4-6]进行分析,系统结构如图 1 所示.

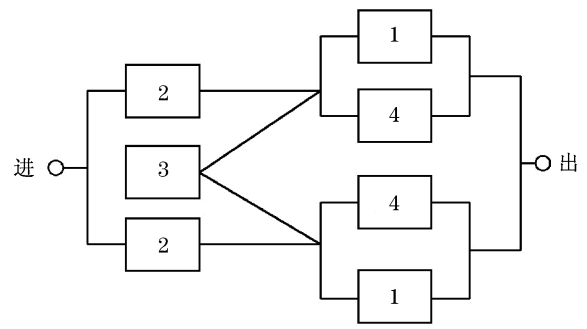


图 1　非串联-并联系统

Fig.1　Non-Series-Parallel Systems

系统可靠性

$$R_s = 1 - R_3[(1-R_1)(1-R_4)]^2 - (1-R_3)\{(1-R_2[1-(1-R_1)(1-R_4)]\}^2$$

系统费用 C_s 有两种形式:

a. $C_s = 2\sum_{i=1}^{4} k_i R_i^{\alpha_i}$

其中,$k_1 = 1$,$k_2 = 100$,$k_3 = 200$,$k_4 = 150$,$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.6$,$R_{i,\min} = 0.5$,$R_{s,\min} = 0.9$.

b. $C_s = \sum_{i=1}^{4} k_i[\tan(\frac{\pi}{2}R_i)]^{\alpha_i}$

其中,$k_1 = 25$,$k_2 = 25$,$k_3 = 50$,$k_4 = 37.5$,$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 1$,$R_{i,\min} = 0.5$,$R_{s,\min} = 0.99$.

在求解上述模型时,先将问题转换为无约束优化问题的形式.罚函数的形式一般为和的形式、和的平方形式及乘积形式等.为方便起见,本文采用和的形式,这样在计算过程中可直接考虑对目标函数的优化,可行性的要求由罚函数的作用来强制其满足.

采用混合万有引力搜索算法(hybrid gravitational search algorithm,HGSA)进行求解,并与蚁群优化算法[11-12](ant colony optimization algorithm,ACO)、微粒群算法[13](particle swarm optimization algorithm,PSO)、人工蜂群算法[14-15](artificial bee colony algorithm,ABC)的求解性能进行比较. ACO 和 PSO 属于经典的智能优化算法,ABC 是最近提出的一种新的智能算法,这 3 种算法已成功用于求解很多复杂困难的优化问题,有关这 3 种算法的详细信息可参考文献[11-15].

本文算法采用 Matlab 7.1 语言编程实现,在 Windows XP 操作系统中,CPU 为 P4 2.4 GHz 和内存为 1 G 的计算机上运行. 为方便起见,5 种算法设置相同的群体规模和最大迭代次数,群体规模和最大迭代次数均为 50. 在 ACO 算法中,信息启发因子 $\alpha=1$,能见度因子 $\beta=1$,轨迹的持久性参数 $\rho=0.7$,信息素常数 $Q=1$;在 PSO 算法中,学习因子 $c_1=c_2=2$,惯性权重 w 从 0.9 线性递减到 0.2;在 ABC 算法中,引领蜂规模和跟随蜂的规模均为 25,侦察蜂规模为 1,阈值 $\lim it=100$,采用轮盘赌的选择机制;GSA 算法和 HGSA 算法参数设置相同:万有引力常数 G 随迭代次数递减,即 $G(t)=G_0\exp\left(-\alpha\dfrac{t}{T}\right)$,其中 $G_0=100$,$\alpha=1$,t 表示当前迭代次数. 每个算例独立运行 20 次并统计最优结果,具体结果如表 1 和表 2 所示.

表 1 模型 1 的实验结果

Tab.1 Computing results for model 1

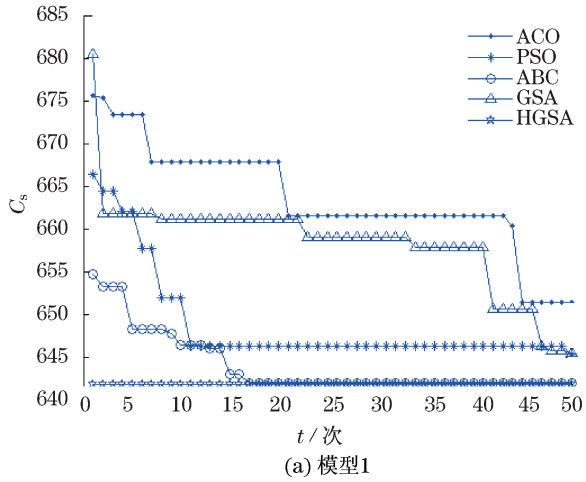
结果	ACO	PSO	ABC	GSA	HGSA
C_s	653.856 7	645.384 9	641.856 9	645.444 0	641.823 5
R_1	0.515 5	0.543 8	0.500 0	0.500 0	0.500 0
R_2	0.777 9	0.784 4	0.839 1	0.867 2	0.838 9
R_3	0.545 3	0.500 1	0.500 0	0.500 0	0.500 0
R_4	0.544 9	0.516 1	0.500 0	0.500 0	0.500 0
R_s	0.903 0	0.900 1	0.900 0	0.907 6	0.900 0

表 2 模型 2 的实验结果

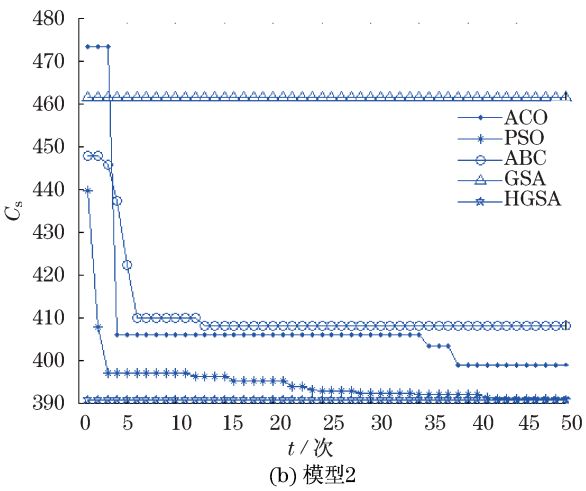
Tab.2 Computing results for model 2

结果	ACO	PSO	ABC	GSA	HGSA
C_s	398.757 1	390.868 5	408.097 0	461.358 7	390.570 7
R_1	0.789 1	0.8220 7	0.869 5	0.630 0	0.825 5
R_2	0.885 1	0.888 1	0.862 9	0.912 9	0.890 1
R_3	0.713 8	0.648 4	0.737 2	0.621 3	0.627 6
R_4	0.736 3	0.726 3	0.640 7	0.859 9	0.728 8
R_s	0.990 0	0.990 0	0.990 1	0.991 4	0.990 0

由表 1 和表 2 可知,对模型 1 而言,HGSA 算法的最优结果小于 ABC 算法结果,远小于 GSA 算法、PSO 算法及 ACO 算法的结果;对模型 2 而言,HGSA 算法的最好结果小于 PSO 算法的结果,比 GSA 算法、ABC 算法及 ACO 算法的结果小的多. HGSA 算法表现出较高的优化精度,为进一步分析算法的性能,图 2 给出了 5 种算法的寻优曲线图.



(a) 模型1



(b) 模型2

图 2 寻优曲线图

Fig.2 The varying curves

由图 2 可知,在这 5 种算法中,HGSA 算法具有最快的优化速度,算法只需很少的迭代次数就能达到最优并趋向稳定,算法具有较好的稳定性. HGSA 算法在优化精度和寻优速度两方面均表现优异. 分析原因,基本 GSA 算法全局搜索能力强,局部搜索能力弱,HGSA 算法中嵌入了 SQP 算法,改善局部搜索能力,能避免算法早熟收敛,实现全局探索和局部开发能力的平衡,提高优化性能.

(下转第 342 页)