DOI: 10.3901/JME.2021.06.236

多阶段自适应蝙蝠-蚁群混合群智能算法^{*}

陈东宁^{1,2} 刘一丹^{1,2} 姚成玉³ 杨晓荣^{1,2}

- (1. 燕山大学河北省重型机械流体动力传输与控制重点实验室 秦皇岛 066004;
2. 先进锻压成形技术与科学教育部重点实验室(燕山大学) 秦皇岛 066004;
3. 燕山大学河北省工业计算机控制工程重点实验室 秦皇岛 066004)

摘要: 针对蝙蝠算法在优化过程中未充分利用蝙蝠间搜索信息交互影响的不足, 借鉴拟态物理学中的作用力规则, 基于阶段性搜索策略将搜索过程分为两个阶段, 分别构造符合算法阶段性搜索特点的作用力规则, 提出多形态作用力蝙蝠算法, 并利用 Benchmark 函数对所提算法与标准蝙蝠算法、变异蝙蝠算法、标准微粒群算法、两阶段微粒群算法进行性能对比测试, 结果表明, 所提算法具有更好的寻优能力。针对标准蚁群算法在离散空间优化时信息素更新机制单一、容易早熟收敛的不足, 结合蚁群的实际社会活动提出多阶段自适应信息素机制蚁群优化算法, 并在算法出现长时间停滞时, 引入混沌算子使算法跳出早熟收敛, 更好地发挥蚁群算法的优势, 相对于标准蚁群算法、引入差分进化算法交叉变异机制的混合微粒群算法、基于动态局部搜索蚁群算法, 所提算法在旅行商问题中具有更高的寻优精度、更好的稳定性。为综合不同群智能算法的优势, 针对多形态作用力蝙蝠算法全局搜索能力强、收敛速度快, 多阶段自适应信息素机制蚁群优化算法局部精细化能力强的特点, 将两种算法串行混合, 提出了多阶段自适应蝙蝠-蚁群混合群智能算法。最后, 通过液压系统可靠性优化和串-并联多态系统可靠性优化实例, 验证了所提混合群智能算法的有效性。

关键词: 蝙蝠算法; 蚁群算法; 混合群智能; 可靠性优化

中图分类号: TB114; TB18

Multi-stage Adaptive BA-ACO Hybrid Swarm Intelligence Algorithm

CHEN Dongning^{1,2} LIU Yidan^{1,2} YAO Chengyu³ YANG Xiaorong^{1,2}

- (1. Hebei Provincial Key Laboratory of Heavy Machinery Fluid Power Transmission and Control, Yanshan University, Qinhuangdao 066004;
2. Key Laboratory of Advanced Forging & Stamping Technology and Science(Yanshan University), Ministry of Education of China, Qinhuangdao 066004;
3. Key Laboratory of Industrial Computer Control Engineering of Hebei Province, Yanshan University, Qinhuangdao 066004)

Abstract: A multi-form force bat algorithm(MFBA) is proposed and searching process is divided into two stages based on periodical searching strategy drawing lessons from acting force rules in physics for dealing with the deficiency that interacting information among bats is not made full use of the bat algorithm(BA) in searching process, in addition, Benchmark functions are used to compare the performance of the proposed algorithm with standard BA, variation BA, standard particle swarm optimization(PSO) algorithm and two-stage force PSO(TFPSO) algorithm, the results show that better search ability of optimal solution can be obtained by the proposed algorithm. Aiming at the defection that pheromone updating mechanism is single and is easy to fall into premature convergence in the discrete space optimization of the standard ant colony optimization(ACO) algorithm, so a multi-stage adaptive pheromone ACO(MAPACO) algorithm combining with the actual ant social activities is proposed, when the proposed algorithm appears to be stagnant for a long time, the chaotic operator is introduced to help the algorithm to jump out of the premature convergence, and to give full play to the advantages of ACO algorithm, compared with the hybrid PSO based on cross variation mechanism of differential evolution algorithm, standard ACO and ACO based on dynamic local search, the proposed algorithm is

^{*} 国家自然科学基金(51975508, 51675460)和中国博士后科学基金(2017M621101)资助项目。20200315 收到初稿, 20201008 收到修改稿

proved to have higher search accuracy and better stability in traveling salesman problem. Aiming at the two methods' advantages that the MFBA algorithm have stronger global searching ability and faster convergence rate, while the MAPACO algorithm can deal with local problems more elaborately, a multi-stage adaptive MFBA-MAPACO hybrid swarm intelligence algorithm is proposed. Finally, the proposed hybrid swarm intelligence algorithm is applied in the reliability optimization of hydraulic system and reliability optimization of series parallel multi-state system, the effectiveness of the proposed MFBA-MAPACO hybrid swarm intelligence algorithm is further verified.

Key words: bat algorithm; ant colony optimization algorithm; hybrid swarm intelligence; reliability optimization

0 前言

蝙蝠算法(Bat algorithm, BA)是一种模拟蝙蝠搜索目标的新型群智能优化算法,是由剑桥大学 YANG 于 2010 年提出的。相对于其他元启发式算法,蝙蝠群通过发送频率、脉冲发射率和响度,调整与当前最优解的距离进行搜索寻优,利用频率调谐特性实现较快全局搜索效率^[1]。目前,蝙蝠算法的工程应用愈加广泛,如时间序列预测^[2]、工程设计优化^[3]、风速预测^[4]等。蝙蝠算法的寻优主要依靠蝙蝠个体间速度和位置的信息交互^[5]。标准蝙蝠算法在飞行速度、位置更新时只受到最优蝙蝠的影响,易陷入局部最优,受到局部极值约束后很难摆脱。为此,学者们从蝙蝠的飞行速度、位置更新机制研究并改进蝙蝠算法,例如,文献[6]采用两种脉冲发射向两个不同方向,并融入速度、位置更新中,来指导蝙蝠的飞行方向,提高蝙蝠的勘探和开发能力;文献[7]将惯性权重引入到蝙蝠算法的速度更新公式中,平衡算法的全局和局部搜索能力;文献[8]引入具有较好随机性的 Lévy 飞行,将其作用于蝙蝠算法的位置更新公式中,使算法跳出局部最优;文献[9]通过综合模拟退火思想以及高斯干扰来改变蝙蝠算法速度更新方式,在算法后期帮助跳出局部最优;文献[10]在速度更新公式中使用随机蝙蝠来引导个体蝙蝠飞行和随机搜索,帮助算法跳出局部最优。然而,上述改进算法虽然改善了飞行速度、位置的单一更新机制,丰富了搜索空间,减缓陷入局部最优,但在平衡搜索空间和收敛精度上存在提升的空间,收敛精度还有待提高。

蚁群(Ant colony optimization, ACO)算法是 20 世纪 90 年代意大利学者 DORIGO 参照蚂蚁的群体活动交流机制提出的。BONABEAU 等^[11]发表的蚁群算法研究论文,为该算法的学术前沿地位提供了基础,直至今日,蚁群算法以其极高的求解效率仍为学者研究的热点^[12]。针对蚁群算法收敛速度较慢、易早熟收敛等不足,学者们对算法进行了研究改进,根据蚁群算法中信息素更新机制对算法的寻

优性能起到关键性作用的特点,学者们主要从信息素更新机制对算法进行改进,例如,为防止蚁群出现早熟收敛,限定信息素最大、最小值,降低信息素间差距,改善算法只在搜索后期局部搜索的不足^[13];通过动态调整信息素挥发系数来提高收敛速度^[14];完成一次遍历搜索后,只有最优蚂蚁信息素更新,提高收敛速度^[15];采用不同搜索阶段自动调节信息素重要程度因子的自适应策略,提高收敛速度^[16];把蚁群的搜索过程划分为前期和后期两个阶段,采用多阶段搜索策略以加快算法初期收敛速度,同时也加快了各路径自己的信息素积累^[17];通过调整蚁群的初始信息素分配,引入局部信息素调控机制,提高收敛速度^[18]。以上算法难以同时改进收敛速度和改善早熟收敛,或限于改进收敛速度、或限于改善早熟收敛,且不能够完全真实地反映蚁群社会的协同寻优机制。

蝙蝠、蚁群算法作为单一的一种群智能算法,存在早熟收敛、易陷入局部最优、适用范围局限等不足,为了克服单一算法的局限性,实现两者的优势互补,通过设计一种混合算法来进一步提高算法的性能^[19],例如,文献[20]将混沌优化算法和蝙蝠算法相混合,利用混沌运动的遍历性和内在随机性,使蝙蝠算法跳出局部最优,实现全局优化;文献[21]把微粒群与蚁群这两种算法串联,将蚁群初始解范围设置为微粒群的优化结果,缩短了蚁群算法寻优时间;文献[22]和文献[23]在蚁群算法中结合了遗传算法的交叉变异规则,同时兼顾了蚁群保持反馈信息的优点和遗传算法的快速全局搜索能力。

为此,本文旨在更好模拟生物群体间的信息交互机制,改进蝙蝠算法和蚁群算法,并把改进后的两种算法进行混合互补。首先,针对蝙蝠算法未充分利用信息交互的不足,采用阶段性搜索策略将搜索过程划分为两个阶段,借鉴拟态物理学中的作用力规则,分别构造契合算法阶段性搜索要求的作用力规则,提出多形态作用力蝙蝠算法(Multi-form force BA, MFBA);然后,针对标准蚁群算法在离散空间优化时存在信息素更新机制单一的不足,基于蚁群社会活动特征提出多阶段自

适应信息素机制蚁群优化(Multi-stage adaptive pheromone ACO, MAPACO)算法,依照不同阶段对精英蚂蚁释放信息素浓度需求不同进行信息素浓度更新,并在搜索后期出现长时间停滞时,将混沌算子引入,以跳出早熟收敛,充分发挥蚁群算法优势;最后,在验证上述所提算法有更多样化的搜索形式、更容易跳出局部最优、更高的寻优精度的基础上,为综合蝙蝠和蚁群两种算法的优势,将 MFBA 算法与 MAPACO 算法进行串行混合,提出多阶段自适应蝙蝠-蚁群混合群智能算法——MFBA-MAPACO 算法,并用于解决可靠性优化问题。

1 多形态作用力蝙蝠算法

标准蝙蝠算法作用力规则单一,仅涉及当前蝙蝠与全局最优蝙蝠位置间的斥力作用,不能够很好地平衡寻优精度和收敛速度两者之间的关系。为此,我们提出多形态作用力蝙蝠算法——MFBA 算法。所提出的 MFBA 算法的思想是设计算法的作用力规则时要依据蝙蝠在不同阶段的搜索特点,具体为:在算法搜索前期,提高搜索范围多样性和全局搜索能力,使蝙蝠同时受到其他蝙蝠的引力与斥力作用,使蝙蝠与其他所有蝙蝠充分交流;当算法运行一定迭代次数后,让蝙蝠在速度、位移更新过程中受到两种引力作用,在搜索空间中进行集中式精细搜索,使算法快速集中到最优解附近。

1.1 标准蝙蝠算法

事实上,蝙蝠活动有诸多不确定,难以在算法上予以体现,为此蝙蝠算法有如下设定。

(1) 每个蝙蝠用回声定位感知距离,而且可区分食物和障碍物。

(2) 初始的飞行速度、方向随机,搜索过程会改变响度 A^i 和波长 λ ,并根据距离远近自动调节发射的脉冲波长、速率 $R \in [0,1]$ 。

设蝙蝠处于 n 维搜索空间中,蝙蝠 i 发出的频率表示为 f_i ,蝙蝠 i 的位置和速度向量分别表示为 $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$ 、 $v_i = (v_{i1}, v_{i2}, \dots, v_{in})$,蝙蝠 i 在第 t 次迭代中的速度和位置更新公式为式(1)~(3)

$$f_i = f_{\min} + \beta(f_{\max} - f_{\min}), \quad \beta \in [0,1] \quad (1)$$

$$v_{id}(t+1) = v_{id}(t) + f_i(x_{id}(t) - p_{gd}(t)) \quad (2)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (3)$$

式中, $f_{\min}$ 为最小频率; β 为 $[0, 1]$ 区间上均匀分布的随机向量; $f_{\max}$ 为最大频率; $v_{id}(t)$ 、 $v_{id}(t+1)$ 分

别为第 t 、 $t+1$ 代蝙蝠 i 的第 d 维速度; $p_{gd}(t)$ 为第 t 代全局最优蝙蝠 g 的第 d 维位置; $x_{id}(t)$ 、 $x_{id}(t+1)$ 分别为第 t 、 $t+1$ 代蝙蝠 i 的第 d 维位置。

蝙蝠在全局搜索同时,还进行局部精细化搜索:从当前个体最优解集中任意选择一个解 x_{old} , 蝙蝠在 x_{old} 附近产生一个新的位置 x_{new}

$$x_{new} = x_{old} + \varepsilon A(t) \quad (4)$$

式中, ε 为 $[-1, 1]$ 区间上的随机数; $A(t)$ 为在 t 次迭代中所有蝙蝠的平均响度。

在进行飞行速度更新、位置更新时,还进行脉冲发射响度 A_i 、速率 r_i 更新。发现猎物,速率提高、响度下降,初始速率 $r_i(1)$ 、响度 $A_i(1)$ 分别设定为较小、较大值,响度 A_i 、速率 r_i 的更新公式为

$$A_i(t+1) = \alpha A_i(t) \quad (5)$$

$$r_i(t+1) = r_i(1)[1 - \exp(-\gamma(t+1))] \quad (6)$$

式中, $A_i(t)$ 、 $A_i(t+1)$ 分别是蝙蝠 i 在第 t 代、 $t+1$ 代的响度; α 是响度衰弱因子,为 $[0, 1]$ 区间上的常量; $r_i(t+1)$ 是蝙蝠 i 在第 $t+1$ 代的脉冲速率; γ 是脉冲频度增加系数,为大于 0 的常量。

1.2 多形态作用力蝙蝠算法

1.2.1 前期平衡引斥力规则

为使蝙蝠算法更好进行全局搜索,参考拟态物理学中的虚拟力,受自然界中生物趋利避害的启发,使蝙蝠能够同时受其他蝙蝠的引力作用与斥力作用,构造引斥力规则如式(7)所示

$$\sum_{j \in B(i)} c_j s(p_{jd}(t) - x_{id}(t)) + \sum_{j \in W(i)} c_j s(x_{id}(t) - p_{jd}(t)) \quad (7)$$

式中, c_j 是加速常数; s 是在 $(0, 1)$ 区间上的随机数; $p_{jd}(t)$ 是为第 t 代蝙蝠 j 的个体最优蝙蝠第 d 维位置; $p_{jd}(t) - x_{id}(t)$ 、 $x_{id}(t) - p_{jd}(t)$ 分别是蝙蝠 j 的个体最优蝙蝠对蝙蝠 i 的引力、斥力作用; $B(i) = \{j | F(\mathbf{P}_j) < F(\mathbf{X}_i), j = 1, 2, \dots, N_b\}$ 、 $W(i) = \{j | F(\mathbf{P}_j) > F(\mathbf{X}_i), j = 1, 2, \dots, N_b\}$ 分别是存放比蝙蝠 i 适应值好或差的最优蝙蝠集合, N 为蝙蝠个数。

1.2.2 后期双引力规则

前期平衡引斥力规则用以提高全局搜索能力,在此基础上,后期构造双引力规则。

1.2.2.1 动力加速度

在标准蝙蝠算法中,蝙蝠仅在全局最优蝙蝠作用下移动,基于此,增加个体最优位置的作用,实现群体经验和个体经验的共同作用,为蝙蝠搜索提供新动力,将这种融合社会认知项和个体认知项的加速度项添加至位置更新公式。蝙蝠 i 第 d 维的动

力加速度 $a_{id}(t)$ 定义为

$$a_{id}(t) = s(p_{id}(t) - x_{id}(t)) + s(p_{gd}(t) - x_{id}(t)) \quad (8)$$

式中, $p_{id}(t)$ 是第 t 代蝙蝠 i 的个体最优蝙蝠第 d 维位置。

1.2.2.2 中值导向加速度

为使蝙蝠能够飞出局部最优解, 使蝙蝠在无其他蝙蝠提供动力时, 仍存在帮助蝙蝠飞出局部最优解的速度项, 在速度更新公式中添加中值导向加速度。蝙蝠 i 第 d 维的中值导向加速度 $m_{id}(t)$ 定义如式(9)所示

$$m_{id}(t) = b_i(t)[s(p_{id}(t) - p_{md}(t) - x_{id}(t)) + s(p_{gd}(t) - p_{md}(t) - x_{id}(t))] \quad (9)$$

式中, $p_{md}(t)$ 是第 t 代蝙蝠 m 的第 d 维位置, $b_i(t)$ 是加速因子, 由式(10)、(11)计算得到

$$b_i(t) = \frac{E_i(t)}{\sum_{j=1}^N E_j(t)} \quad (10)$$

$$E_i(t) = \frac{F(x_i) - F_{\max}}{F_{\max} - F_{\min}} \quad (11)$$

式中, $F(x_i)$ 是第 t 代蝙蝠 i 的适应值; $F_{\min}$ 和 $F_{\max}$ 分别是中间和最大适应值。

1.2.3 速度和位置更新规则

基于所提 MFBA 算法的前期平衡引斥力规则以及后期双引力规则, 建立 MFBA 算法的速度和位置更新公式

$$v_{id}(t+1) = wv_{id}(t) + \xi \left(\sum_{j \in B(i)} c_j s(p_{jd}(t) - x_{id}(t)) + \sum_{j \in W(i)} c_j s(x_{id}(t) - p_{jd}(t)) \right) + \beta m_{id}(t) \quad (12)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) + \beta \frac{1}{2} a_{id}(t) \quad (13)$$

式中, ξ 、 β 分别是切换系数; w 是惯性权重, 可由式(14)求得

$$w = w_{\max} - \frac{t}{t_{\max}}(w_{\max} - w_{\min}) \quad (14)$$

$w_{\min}$ 、 $w_{\max}$ 分别为惯性权重最小值和最大值; t 、 $t_{\max}$ 分别为当前迭代次数和最大迭代次数。

切换系数 ξ 、 β , 分别由式(15)、(16)求得

$$\xi = \begin{cases} 1 & \gamma < \gamma_{\max} \\ 0 & \text{其他} \end{cases} \quad (15)$$

$$\beta = \begin{cases} 1 & \gamma \geq \gamma_{\max} \\ 0 & \text{其他} \end{cases} \quad (16)$$

式中, γ 为前期搜索的停滞次数, 算法停滞是指全局最优蝙蝠适应值连续若干代不再变化; $\gamma_{\max}$ 为最大停滞次数。

当 $\gamma \geq \gamma_{\max}$ 时, 表明前期搜索已不能够进一步优化, 需切换至后期搜索。 γ 由式(17)计算而得

$$\gamma = \begin{cases} \gamma + 1 & f(P_g(t)) \geq f(P_g(t-1)) \\ 0 & \text{其他} \end{cases} \quad (17)$$

式中, $P_g(t)$ 、 $P_g(t-1)$ 分别为第 t 、 $t-1$ 代全局最优蝙蝠 g 的位置。

MFBA 算法的流程如图 1 所示。

图 1 MFBA 算法流程

所提 MFBA 算法的具体执行步骤。

(1) 初始化即在搜索空间中 $[x_{\min}, x_{\max}]$ 和 $[v_{\min}, v_{\max}]$ 区间内随机初始化蝙蝠的位置和速度向量, 各蝙蝠发出的频率在 $[f_{\min}, f_{\max}]$ 区间内随机生成; 初始化脉冲速率、响度; 确定全局最优和个体最优蝙蝠。

(2) 计算 $B(i)$ 和 $W(i)$ 中个体最优蝙蝠对当前蝙蝠 i 的引力、斥力, 基于式(12)、式(13)更新蝙蝠 i 的位置 $x_i(t)$ 和速度 $v_i(t)$ 。

(3) 若随机数 $\text{rand} > r_i(t)$, 从当前个体最优解集任意选择一个解 x_{old} , 基于式(4)求解得到新解 x_{new} 。

(4) 根据新解, 求得目标函数的适应值。

(5) 若满足条件 $\text{rand} < A_i$ 且 $f(x_i) < f(p_i)$, 接受新解并更新全局最优和个体最优蝙蝠位置及脉冲速

率 $r_i(t)$ 和响度 $A_i(t)$ 。

(6) 判断算法前期搜索是否已结束, 若前期停滞次数 $\gamma < \gamma_{\max}$, 则返至步骤(2), $t = t + 1$; 否则, 执行步骤(7), $t = t + 1$ 。

(7) 计算蝙蝠 i 的动力加速度 $a_{id}(t)$ 与中值导向加速度 $m_{id}(t)$, 基于式(12)和式(13)更新蝙蝠的速度 $v_i(t)$ 和位置 $x_i(t)$; 执行步骤(3)~(5)。

(8) 判断算法终止条件, 如果 $t = t_{\max}$, 执行步骤(6), $t = t + 1$; 否则, 返至步骤(4), $t = t + 1$ 。

(9) 输出算法的优化结果。

1.3 算法测试

为测试所提 MFBA 算法的性能, 用表 1 的 4 个 Benchmark 函数, 同时并与标准蝙蝠、标准微粒群 (Particle swarm optimization, PSO) 算法对比测试, 三种算法中, 蝙蝠数、微粒数均为 30, 最大迭代 500 次, 测试函数维数为 30, 各算法独立运行 50 次, 其他参数的设置见表 2。将测试结果与文献[24]所提变异蝙蝠算法和文献[25]所提两阶段作用力微粒群 (Two-stage force PSO, TFPSO) 算法进行优化结果对比。所得各函数优化结果的最优解、平均值和标准差见表 3。

表 1 4 个 Benchmark 函数

Benchmark 函数	范围
$f_1(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i , \min f_1(x) = f_1(0, 0, \dots, 0) = 0$	$[-10, 10]$
$f_2(x) = \sum_{i=1}^n x_i^2, \min f_2(x) = f_2(0, 0, \dots, 0) = 0$	$[-100, 100]$
$f_3(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$ $\min f_3(x) = f_3(0, 0, \dots, 0) = 0$	$[-5.12, 5.12]$
$f_4(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos \frac{x_i}{\sqrt{i}} + 1$ $\min f_4(x) = f_4(0, 0, \dots, 0) = 0$	$[-600, 600]$

表 2 算法参数设置

参数	PSO	BA	MFBA
加速常数 c	1.49	1.49	1.49
最小惯性权重 $w_{\min}$	0.4	0.4	—
最大惯性权重 $w_{\max}$	0.9	0.9	—
蝙蝠频率区间 $[f_{\min}, f_{\max}]$	—	$[0, 2]$	$[0, 2]$
蝙蝠初始速率 $r_i(1)$	—	0.1	0.1
蝙蝠响度衰减因子 α	—	0.95	0.95
蝙蝠初始响度 $A_i(1)$	—	1.5	1.5
脉冲发射率控制参数 γ	—	0.95	0.95
进化停滞次数阈值 $\gamma_{\max}$	—	—	50

表 3 优化结果

算法	测试指标	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
标准 BA	最优解	2.5×10^1	2.5×10^0	2.1×10^2	2.3×10^2
	平均值	2.6×10^1	8.0×10^0	2.3×10^2	4.0×10^2
	标准差	4.6×10^0	4.0×10^0	1.9×10^1	7.8×10^1
变异 BA ^[24]	最优解	2.7×10^{-1}	1.3×10^{-3}	7.3×10^1	4.7×10^1
	平均值	9.5×10^0	2.0×10^{-3}	1.5×10^2	1.29×10^2
	标准差	2.5×10^1	5.6×10^{-4}	3.3×10^1	4.2×10^1
PSO	最优解	2.6×10^1	5.2×10^2	1.4×10^2	1.1×10^1
	平均值	2.8×10^1	8.1×10^2	1.5×10^2	1.4×10^1
	标准差	4.7×10^0	3.0×10^2	1.0×10^1	1.1×10^1
TFPSO ^[25]	最优解	3.1×10^{-33}	5.2×10^{-80}	0^*	0^*
	平均值	4.8×10^{-21}	4.6×10^{-16}	1.2×10^0	1.8×10^{-17}
	标准差	2.6×10^{-20}	2.5×10^{-15}	2.4×10^0	4.8×10^{-17}
MFBA	最优解	$4.7 \times 10^{-45*}$	$2.8 \times 10^{-92*}$	0^*	0^*
	平均值	$2.8 \times 10^{-33*}$	$1.5 \times 10^{-40*}$	0^*	0^*
	标准差	$1.1 \times 10^{-32*}$	$8.5 \times 10^{-40*}$	0^*	0^*

注: 标注 “*” 者表示对应指标的最优解。

MFBA 算法最优解、寻优精度、稳定性均为最好。尤其对于 $f_3(x)$ 、 $f_4(x)$, 所提的 MFBA 算法能够使这 2 个测试函数的优化结果稳定在函数最优解, 而标准蝙蝠算法、变异蝙蝠算法不善于解决存在多个局部极小点的多模态问题。由此可见, MFBA 算法在多样化的优化问题中都能够达到理想的优化效果。

2 多阶段自适应信息素机制蚁群算法

虽然蚁群算法正反馈机制能够强化较好解, 然而却易使得算法出现停滞现象。因而, 提出多阶段自适应信息素机制蚁群优化算法: 在搜索前期, 为促使后面蚂蚁有更多路径可选择, 加入最差路径对当代最优路径的退步项; 在算法搜索中期, 为区别好的与差的路径, 加入当前最优路径对最差路径的进步项; 在算法搜索后期, 为扩展搜索范围、防止过早停滞、提高全局搜索能力, 加入混沌扰动。

2.1 标准蚁群算法

蚁群算法自提出以来, 在解决旅行商问题 (Traveling salesman problem, TSP) 中有突出优势。两城市 i 、 j 的距离设为 $d_{ij}(i, j = 1, 2, \dots, n)$, 蚂蚁个数设为 m , 第 t 次迭代蚂蚁 k 由城市 i 转移至城市 j 的概率 $P_{ij}^k(t)$ 为

$$P_{ij}^k = \begin{cases} \frac{[\tau_{ij}(t)]^v [\eta_{ij}(t)]^\chi}{\sum_{s \in \text{allow}_k} [\tau_{is}(t)]^v [\eta_{is}(t)]^\chi} & s \in \text{allow}_k \\ 0 & s \notin \text{allow}_k \end{cases} \quad (18)$$

式中, $\tau_{ij}(t)$ 为第 t 次迭代城市 i 与城市 j 连接路径上的信息素浓度, 且 $\tau_{ij}(0) = \tau_0$; v 为信息素重要程度因子; $\eta_{ij}(t) = 1/d_{ij}$ 表示蚂蚁由城市 i 转至城市 j 的期望程度, 称为启发函数; $\eta_{is}(t)$ 代表蚂蚁从城市 i 转移到城市 s 的期望程度; χ 是启发函数重要程度因子; $\text{allow}_k (k=1,2,\dots,m)$ 是蚂蚁 k 待访问城市的集合。

用信息素挥发因子 $\rho (0 < \rho < 1)$ 来表示蚂蚁信息素挥发浓度, 当所有蚂蚁循环一次后, 更新公式如式(19)~(21)所示

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij} \quad (19)$$

$$\Delta\tau_{ij} = \sum_{k=1}^n \Delta\tau_{ij}^k \quad (20)$$

$$\Delta\tau_{ij}^k = \begin{cases} Q/L_k, & \text{第 } k \text{ 只蚂蚁从市 } i \text{ 访问市 } j \\ 0, & \text{其他} \end{cases} \quad (21)$$

式中, $\tau_{ij}(t+1)$ 、 $\tau_{ij}(t)$ 分别为第 $t+1$ 、 t 次迭代的信息素浓度; $\Delta\tau_{ij}^k$ 、 $\Delta\tau_{ij}$ 分别为第 k 只、所有蚂蚁释放的信息素浓度; L_k 为蚂蚁完成一次循环后经过的所有路径长度; Q 为影响系数。

2.2 多阶段自适应信息素机制蚁群算法

2.2.1 局部信息素更新机制

引入局部信息素在线更新机制, 提高了全局最优搜索能力, 扩大了搜索空间。

(1) 将每条路径的信息素浓度设在 $(\tau_{\min}, \tau_{\max})$ 区间, 限制信息素浓度范围, 防止某一路径信息素浓度过大或过小。

(2) 局部信息素更新公式为

$$\tau_{ij}(t+1) = (1-\phi)\tau_{ij}(t) + \eta\tau_0 \quad (22)$$

式中, $\phi \in [0, 1]$ 为信息素局部挥发因子; τ_0 为各路径初始信息素浓度, 令 $\tau_0 = \tau_{\max}$; $\eta \in [0, 1]$ 为初始信息素遗留因子。

2.2.2 多阶段自适应全局信息素更新机制

更新机制具体如下。

(1) 算法搜索前期, 添加最差路径对当代最优路径的退步项, 使后面蚂蚁能有更多路径可选择, 信息素更新公式为

$$\tau_p(t+1) = (1-\rho)\tau_p(t) + Q_1(L_g(t) - L_w(t)) \quad 1 \leq t \leq t_1 \quad (23)$$

算法搜索中期, 添加当前最优路径对最差路径的进

步项, 使好的和差的路径可区别开, 提高收敛效率, 信息素更新公式为

$$\tau_p(t+1) = (1-\rho)\tau_p(t) + Q_2(L_w(t) - L_g(t)) \quad t_1 < t \leq t_2 \quad (24)$$

式中, $\tau_p(t+1)$ 、 $\tau_p(t)$ 分别为迭代次数为 $t+1$ 、 t 时当代最优路径信息素; t_1 、 t_2 分别为前期、中期搜索停滞阈值; ρ 为信息素全局挥发因子; $L_g(t)$ 、 $L_w(t)$ 分别为迭代 t 次全局最优、最差路径长度; Q_1 、 Q_2 为影响系数。

2.2.3 添加混沌扰动的全局信息素更新机制

在添加当前最优路径对最差路径进步项的基础上, 为抑制算法后期因最优最差路径上信息素差距剧增引起的搜索停滞, 在信息素更新方程中添加混沌扰动因子, 引入一种能够有效跳出局部最优的混沌搜索算法。这里, 采用 Logistic 混沌映射

$$u_{v+1} = \mu u_v (1 - X_v), 0 < u_v < 1, v=1,2,\dots \quad (25)$$

式中, μ 为分支参数, μ 的取值范围为 $3.85 \sim 4.00$; u_v 、 u_{v+1} 分别为第 v 、 $v+1$ 次混沌迭代的数值。

全局信息素更新公式为

$$\tau_p(t+1) = (1-\rho)\tau_p(t) + Q_3(L_w(t) - L_g(t)) + \delta z_p(t) \quad t_2 < t \leq t_{\max} \quad (26)$$

混沌映像伪载波机制如式(27)所示

$$chz_p = \frac{z_p - z_{\min}}{z_{\max} - z_{\min}} \quad (27)$$

进而更新 z_p , 则如式(28)所示

$$z_p = z_{\min} + chz_p(z_{\max} - z_{\min}) \quad (28)$$

式中, Q_3 为影响系数; δ 为扰动系数; z_p 为加入扰动后的信息素; chz_p 为 Logistic 混沌映像产生的混沌序列; $z_{\min}$ 、 $z_{\max}$ 分别为扰动量最小值和最大值。

2.3 算法测试

用旅行商问题标准测试库的五个问题(chn31、att48、eil51、ch130、ch150)进行测试, 考察所提 MAPACO 算法的优化性能, 这五个问题城市规模分别为 31、48、51、130、150, 前两个属小规模问题, 第三个属中规模问题, 后两个属大规模问题。将 MAPACO 算法与 ACO 算法、文献[26]引入差分进化(Differential evolution, DE)交叉变异的混合微粒群算法——PSO-DE 算法, 进行测试对比。三种算法参数设置为: 蚂蚁/微粒个数 $m=20$, 最大迭代次数 $t_{\max}=2000$ 。ACO、MAPACO 算法中,

信息素全局挥发因子 $\rho=0.4$, 信息素重要程度因子 $v=1$, 启发函数重要程度因子 $\chi=5$, 信息素局部挥发因子 $\phi=0.2$, 初始信息素遗留因子 $\eta=0.1$, 扰动系数 $\delta=1.2$, 影响系数 Q_1 、 Q_2 、 Q_3 分别为 1×10^{-9} 、 5×10^{-7} 、 5×10^{-5} , 前期、中期搜索停滞阈值 t_1 、 t_2 分别为 200、500, 分支参数 $\mu=4$, 扰动量最小值 $z_{\min}$ 、最大值 $z_{\max}$ 分别为 0.000 1、1。三种算法、五个路径优化问题的优化结果, 如图 2 所示。

图 2 最优路径长度

可见, ACO 算法在开始时收敛最快, 收敛精度不高, 难跳出局部最优解; PSO-DE 在解决小规模和中规模问题时收敛精度较好, 算法在开始时收敛较慢, 解决大规模问题远差于 ACO、MAPACO 算法; MAPACO 算法能够多次跳出局部最优解, 开始收敛较早, 可以适用于 TSP 问题, 在上述问题均能够达到很好的优化效果。

为进一步说明 MAPACO 算法的优化性能, 将其对比文献[27]算法——基于动态局部搜索蚁群算法(Dynamic local ant colony algorithm, DLACA), 算法参数设置与文献[27]一致, 表 4 是上述四种算法对上述五个路径优化问题独立运行 10 次的结果。

可见, 城市规模越大 PSO-DE 算法的优化能力越差, PSO-DE 算法受城市规模影响较大; 相对于 MAPACO 算法, ACO 算法的标准差略优, 但寻找最优路径的能力较差; DLACA 算法的平均值和最优解均优于 ACO 算法, 说明该算法的改进是有效的; MAPACO 算法均能找到最接近已知理论最优解的解, 且标准差较好。综上所述, MAPACO 算法在寻优精度和寻优稳定性上均有较大优势。

表 4 试验结果

TSP 实例	ACO			PSO-DE			DLACA ^[27]			MAPACO			理论最优解
	平均值	标准差	最优解	平均值	标准差	最优解	平均值	标准差	最优解	平均值	标准差	最优解	
chn31	15 623.5	68.4*	15 601.9	15 912.8	443.2	15 476.5	—	—	—	15 415.5*	69.6	15 377.8*	15 377
att48	35 115.3	154.5*	34 891.6	37 196.3	1 268.9	35 505.7	33 953.0	—	33 549.0	33 775.2*	190.9	33 523.7*	33 522
eil51	446.2	2.3*	436.9	478.4	15.0	455.4	436.9	—	431.2	431.8*	3.5	428.9*	426
ch130	6 435.2	53.6*	6 337.8	13 730.0	757.8	12 778.8	6 317.5	—	6 270.7	6 255.2*	66.6	6 159.8*	6 110
ch150	6 813.7	18.9*	6 790.2	16 420.8	453.4	15 791.9	6 698.0	—	6 617.3	6 671.9*	67.7	6 570.8*	6 528

注:标注“*”者表示对应指标的最优解。

3 混合蝙蝠-蚁群算法

蝙蝠算法收敛速度快但搜索精度低;蚁群算法初始搜索时间较长,把蚁群的搜索行为集中至最优解附近能够提高搜索速度和求解质量。

结合前面所提的 MFBA 算法全局性搜索能力强、而 MAPACO 算法局部精细化搜索能力强的特点,发挥两种算法的优势,提出将 MFBA 算法和 MAPACO 算法顺序串联的 MFBA-MAPACO 混合群智能算法,可以克服单种生物种群先天局限所致的优化与适用不足。用 MFBA 算法进行前期优化,在搜索停滞时,用 MFBA 算法所求得的最优解位置周边作为 MAPACO 算法的初始搜索范围,进而再利 MAPACO 算法后期搜索,帮助跳出前期搜索到的局部最优解,并向全局最优解靠拢。

MFBA-MAPACO 混合群智能算法继承了 MFBA 和 MAPACO 算法的各自优点,实现两种算法的协同配合,能够克服 MFBA 算法易快速收敛到局部最优的不足,又可以改善 MAPACO 算法在搜索初期因信息素差异不大所导致的需要较长时间聚集在最优解附近的不足,形成优势互补。

当 MFBA-MAPACO 算法的迭代次数达到前后期切换迭代次数 t_c 时,前期搜索结束, t_c 需满足三个条件,如式(29)所示

$$\begin{cases} F(P_g(t_c)) = F(P_g(t_{hmax})) \\ F(P_g(t_c - 1)) \neq F(P_g(t_{hmax})) \\ \frac{F(P_g(t_{hmax})) - F(P_g(t_c))}{t_{hmax} - t_c} = 0 \end{cases} \quad (29)$$

式中, t_{hmax} 为 MFBA-MAPACO 算法的最大迭代次数; $P_g(t_c)$ 、 $P_g(t_{hmax})$ 、 $P_g(t_c - 1)$ 分别为第 t_c 、 t_{hmax} 、 $t_c - 1$ 代全局最优蝙蝠 g 的位置。

MFBA-MAPACO 算法流程图如图 3 所示。

图 3 MFBA-MAPACO 算法流程图

4 可靠性优化

选取文献[21]液压系统可靠性优化与文献[28]串-并联系统可靠性优化实例,对所提 MFBA-MAPACO 算法求解实际复杂优化问题的能力进行验证。

4.1 液压系统可靠性优化

分体式巷道运输车是用于在巷道内进行运输的液压自走式设备,由两部分组成:执行运输工作的液压动力运输车、提供动力源的液压动力源车。该

运输车液压系统由液压源、工作系统和行走系统三部分组成,针对液压动力运输车的液压工作系统,运用提出的多阶段自适应蝙蝠-蚁群混合群智能算法进行可靠性优化对比测试。

4.1.1 液压工作系统 T-S 故障树

针对液压工作系统,利用 T-S 故障树进行可靠性建模。T-S 故障树包括基本事件、中间事件和顶事件三个部分,在液压工作系统中,基本事件是会引起系统故障的各基本元件,中间事件是液压工作系统的转向系统、支腿系统和悬挂系统等子系统,顶事件是指整个液压工作系统,建造的液压工作系统 T-S 故障树如图 4 所示,其中, $G_1 \sim G_4$ 为 T-S 门。基本事件对应的液压元件见表 5。

图 4 液压工作系统 T-S 故障树

表 5 基本事件对应的液压元件

基本事件	液压元件	基本事件	液压元件
x_1	斜行液压缸	x_6	多路阀
x_2	直行液压缸	x_7	球阀
x_3	支腿液压缸	x_8	管路防爆阀
x_4	液压锁	x_9	悬挂液压缸
x_5	隔爆电磁换向阀	x_{10}	液控单向阀

基本事件 $x_1 \sim x_4$ 、 x_6 和中间事件 y_1 、 y_2 及顶事件 T 的无故障、轻度故障和完全故障三种故障状态分别用 0、0.5、1 表示;其他事件为无故障、完全故障两种故障状态,分别用 0、1 表示。

4.1.2 可靠性优化对比

对液压工作系统进行可靠性优化,即研究系统可靠性和费用的关系,以实现在满足系统可靠性要求的前提下,使得费用尽量低。基于所建造的 T-S 故障树,以最少的系统可靠性费用为目标,通过系统故障率进行约束,建立可靠性优化模型如式(30)所示

$$\begin{aligned} \min C &= \sum_{i=1}^{10} \alpha_i \left[\frac{1}{\lambda(x_i)} \right]^{\beta_i} \\ \text{s.t. } \lambda(T_q) &= \sum_{l=1}^{108} \lambda_0^l P^l(T_q) \leq \lambda_s \end{aligned} \quad (30)$$

式中, C 为系统费用; $\lambda(T_q)$ 为故障状态为 T_q 下的系统故障率; $\lambda(x_i)$ 为基本事件 x_i 的故障率;

$$\begin{aligned} \lambda(y_i) &= \prod_{a_i=2}^{s_i} \lambda(y_i^{a_i}), \text{ 当 } s_i=3 \text{ 时, } y_i^1=0, y_i^2=0.5, \\ y_i^3 &=1; \text{ 当 } s_i=2 \text{ 时, } y_i^1=0, y_i^2=1. \alpha_1=6.21 \times 10^{-5}, \\ \alpha_2 &=5.45 \times 10^{-5}, \alpha_3=6.28 \times 10^{-5}, \alpha_4=8.57 \times 10^{-5}, \\ \alpha_5 &=9.68 \times 10^{-5}, \alpha_6=5.68 \times 10^{-5}, \alpha_7=4.45 \times 10^{-5}, \\ \alpha_8 &=5.79 \times 10^{-5}, \alpha_9=64.03 \times 10^{-5}, \alpha_{10}=1.72 \times 10^{-5}, \\ \beta_i &=1.5 (i=1, 2, \dots, 10); \lambda_s \text{ 为系统允许最大故障率} \\ \lambda_s &=0.0002; \lambda_0^l \text{ 为执行 T-S 规则 } l \text{ 的阈值,} \\ \lambda_0^l &= \lambda(x_5^{a_5}) \times \lambda(x_6^{a_6}) \times \prod_{i=1}^3 \lambda(y_i^{a_i}), P^l(T_q) \text{ 为规则 } l \text{ 中} \end{aligned}$$

故障状态为 T_q 的概率。

由于对该系统建立的可靠性优化模型为有约束问题,为了便于进行算法处理,引入罚函数将该约束问题转化为无约束问题。在算法求出的解不在约束范围内时,添加一惩罚项 $C_{\max}$, 使不满足约束条件的解能被淘汰,即

$$\text{fitness} = \begin{cases} C & \lambda(T_q) \leq \lambda_s \\ C + C_{\max} & \text{其他} \end{cases} \quad (31)$$

将 MAPACO、MFBA、MFBA-MAPACO 算法与文献[21]的 ACO、ACO-PSO、PSO、PSO-ACO 等 4 种算法进行对比测试,为保证对比的公平性,所提算法参数设置尽可能与文献[21]中算法的参数保持一致,种群维数为 10,最大迭代次数取 200,分别比较种群规模 $N=6$ 和 $N=20$ 的优化结果。文献[21]中 4 种算法和所提 MAPACO、MFBA、MFBA-MAPACO 算法可靠性费用对比见表 6。

表 6 可靠性费用对比

种群规模 $N=6$	优化结果	种群规模 $N=20$	优化结果
ACO ^[21]	19400	ACO ^[21]	16384
ACO-PSO ^[21]	6355	ACO-PSO ^[21]	4468
PSO ^[21]	7316	PSO ^[21]	2549
PSO-ACO ^[21]	4080	PSO-ACO ^[21]	1971
MAPACO	18712	MAPACO	15754
MFBA	3622	MFBA	2520
MFBA-MAPACO	1454	MFBA-MAPACO	1317

以 $N=6$ 时的一次优化结果为例,给出文献[21]所提到的 4 种算法和 MAPACO、MFBA、MFBA-MAPACO 算法的液压工作系统液压元件的故障率 $\lambda(x_i)$ 、液压工作系统轻度故障时故障率 $\lambda(T_2)$ 、液压工作系统完全故障时故障率 $\lambda(T_3)$ 和液压工作系统最低的可靠性费用 C 数据见表 7。

表 7 $N=6$ 时 7 种算法优化结果对比

优化参数	ACO ^[21]	ACO-PSO ^[21]	PSO ^[21]	PSO-ACO ^[21]	MAPACO	MFBA	MFBA-MAPACO
$\lambda(x_1)$	7.06×10^{-6}	1.65×10^{-5}	9.99×10^{-6}	1.44×10^{-5}	8.75×10^{-6}	2.34×10^{-5}	4.18×10^{-5}
$\lambda(x_2)$	3.85×10^{-6}	1.02×10^{-5}	1.18×10^{-5}	1.53×10^{-5}	9.32×10^{-6}	1.79×10^{-5}	4.18×10^{-5}
$\lambda(x_3)$	7.27×10^{-6}	1.91×10^{-5}	1.36×10^{-5}	1.87×10^{-5}	8.50×10^{-6}	2.43×10^{-5}	4.01×10^{-5}
$\lambda(x_4)$	9.97×10^{-6}	1.27×10^{-5}	8.59×10^{-5}	1.23×10^{-5}	8.16×10^{-6}	1.82×10^{-5}	3.21×10^{-5}
$\lambda(x_5)$	9.98×10^{-6}	1.32×10^{-5}	3.09×10^{-5}	1.39×10^{-5}	6.42×10^{-6}	2.43×10^{-5}	5.05×10^{-5}
$\lambda(x_6)$	8.22×10^{-6}	2.08×10^{-5}	1.12×10^{-5}	3.69×10^{-5}	9.87×10^{-6}	2.64×10^{-5}	4.31×10^{-5}
$\lambda(x_7)$	6.69×10^{-6}	1.70×10^{-5}	1.41×10^{-5}	1.94×10^{-5}	5.91×10^{-6}	2.37×10^{-5}	4.32×10^{-5}
$\lambda(x_8)$	7.89×10^{-6}	1.61×10^{-5}	1.94×10^{-5}	2.22×10^{-5}	8.22×10^{-6}	2.34×10^{-5}	4.19×10^{-5}
$\lambda(x_9)$	8.78×10^{-6}	1.76×10^{-5}	2.26×10^{-5}	2.81×10^{-5}	9.38×10^{-6}	2.21×10^{-5}	3.76×10^{-5}
$\lambda(x_{10})$	2.69×10^{-6}	1.34×10^{-5}	5.16×10^{-6}	2.36×10^{-5}	8.55×10^{-6}	1.68×10^{-5}	3.33×10^{-5}
C	19 400	6 355	7 316	4 080	18 712	3 622	1 454
$\lambda(T_2)$	2.17×10^{-5}	4.36×10^{-5}	4.00×10^{-5}	5.91×10^{-5}	2.54×10^{-5}	6.71×10^{-5}	1.18×10^{-4}
$\lambda(T_3)$	6.94×10^{-5}	1.39×10^{-4}	1.31×10^{-4}	1.81×10^{-4}	8.16×10^{-5}	2.12×10^{-4}	3.92×10^{-4}
运行时间/s	7.48	9.12	0.75	0.87	8.15	0.77	2.45

由表 6、7 可知, 种群规模无论是 6 还是 20, MFBA-MAPACO 算法的可靠性费用最低, 虽运行时间不是最短, 但与最短运行时间相差不大; MFBA 算法优化效果仅次于 MFBA-MAPACO 算法; 文献[21]的 ACO、ACO-PSO、PSO、PSO-ACO 等 4 种算法中 PSO-ACO 算法搜索到的系统可靠性费用最低、结果最优, 但效果仍没有 MFBA 算法和 MFBA-MAPACO 算法好。本文所提的三种算法与文献[21]PSO-ACO 算法优化的可靠性费用曲线, 如图 5 所示。图 5 直观地展示了 MFBA-MAPACO 算法的优化能力, 更接近于全局最优解, 能有效跳出局部收敛。

图 5 可靠性优化曲线

4.2 串-并联多态系统可靠性优化

串-并联多态系统由 4 个串联子系统构成, 各子系统 $i(i=1, 2, 3, 4)$ 含有 H_i 种并联版本部件可选择, 各子系统可以选择的版本数分别为 $H_1=5$ 、

$H_2=4$ 、 $H_3=6$ 、 $H_4=5$ 。各部件的状态数用 $S_{ij}(j=1, 2, \dots, H_i)$ 表示, 状态概率和性能分别用 p_{ij} 和 g_{ij} 表示, 用 c_{ij} 来表示子系统 i 第 j 个版本部件的价格。系统的性能约束 D_0 为 1.0, 可靠度约束 R_0 分别为 0.990、0.960 和 0.900, 各子系统最大的并联部件数 n_i 限制为 8。部件参数见表 8。

表 8 部件参数

子系统 i	版本 j	概率 p_{ij}	状态性能 g_{ij}	价格 C_{ij}
1	1	[0.970, 0.030]	[0.50, 0]	0.520
	2	[0.964, 0.036]	[0.80, 0]	0.620
	3	[0.980, 0.020]	[0.80, 0]	0.720
	4	[0.969, 0.031]	[1.00, 0]	0.890
	5	[0.960, 0.040]	[1.50, 0]	1.020
2	1	[0.967, 0.033]	[0.20, 0]	0.516
	2	[0.914, 0.086]	[0.50, 0]	0.916
	3	[0.960, 0.040]	[0.50, 0]	0.967
	4	[0.953, 0.047]	[0.75, 0]	1.367
3	1	[0.959, 0.041]	[0.60, 0]	0.214
	2	[0.970, 0.030]	[0.90, 0]	0.384
	3	[0.959, 0.041]	[1.80, 0]	0.534
	4	[0.960, 0.040]	[2.00, 0]	0.614
	5	[0.970, 0.030]	[2.00, 0]	0.783
4	6	[0.960, 0.040]	[2.40, 0]	0.813
	1	[0.989, 0.011]	[0.25, 0]	0.683
	2	[0.979, 0.021]	[0.25, 0]	0.645
	3	[0.980, 0.020]	[0.30, 0]	0.697
	4	[0.960, 0.040]	[0.70, 0]	1.190
	5	[0.980, 0.020]	[0.70, 0]	1.260

建立优化模型如式(32)所示, 部件数 n_{ij} 为自变量, 目标函数是系统费用 C 最低

$$\begin{aligned} \min \quad & C = \sum_{i=1}^5 \sum_{j=1}^{H_i} n_{ij} c_{ij} \\ \text{s.t.} \quad & R \geq R_0 \\ & 1 \leq n_i \leq 8 \\ & n_{ij} \in \mathbb{Z} \quad n_{ij} \geq 0 \end{aligned} \quad (32)$$

可靠度 R 利用通用生成函数来计算, 步骤如下。

(1) 子系统 i 的第 j 个版本部件的通用生成函数如式(33)所示

$$U_{ij}(z) = \sum_{l=1}^{S_{ij}} p_{ij,l} \cdot z^{g_{ij,l}} \quad (33)$$

式中, $g_{ij,l}$ 为子系统 i 的版本 j 部件的状态性能; $p_{ij,l}$ 为部件 j 状态性能对应的概率。

(2) 由 H_i 个版本并联组成子系统 i , 子系统 i 的通用生成函数 $U_i(z)$ 如式(34)所示

$$U_i(z) = \prod_{j=1}^{H_i} [U_{ij}(z)]^{n_{ij}} \quad (34)$$

将式(34)经过一系列数学运算, 最终化简得到公式如式(35)所示

$$U_i(z) = \sum_{l=1}^{S_i} p_{i,l} \cdot z^{g_{i,l}} \quad (35)$$

式中, 子系统 i 的状态数为 S_i , 子系统 i 的状态性能为 $g_i = \{g_{i,1}, g_{i,2}, \dots, g_{i,S_i}\}$, $g_{i,l}$ 是子系统 i 处于状态 l 时的性能; 子系统 i 对应各状态的概率为 $p_i = \{p_{i,1}, p_{i,2}, \dots, p_{i,S_i}\}$, $p_{i,l}$ 为子系统 i 在状态 l 的概率。

(3) 系统的通用生成函数 $U_{SP}(z)$ 如式(36)所示

$$U_{SP}(z) = \sum_{l_1=1}^{S_1} \sum_{l_2=1}^{S_2} \dots \sum_{l_N=1}^{S_N} [p_{1,l_1} p_{2,l_2} \dots p_{N,l_N} z^{\min(g_{1,l_1}, g_{2,l_2}, \dots, g_{N,l_N})}] \quad (36)$$

经过对式(36)的一系列数学运算, 最终化简得到公式如式(37)所示

$$U_{SP}(z) = \sum_{l=1}^{S_{SP}} p_{SP,l} \cdot z^{g_{SP,l}} \quad (37)$$

式中, S_{SP} 表示整个系统的状态个数, 系统的状态性能用 $g_{SP} = \{g_{SP,1}, g_{SP,2}, \dots, g_{SP,S_{SP}}\}$ 表示, $g_{SP,l}$ 是状态 l 的系统性能; 系统的各状态概率为 $p_{SP} = \{p_{SP,1}, p_{SP,2}, \dots, p_{SP,S_{SP}}\}$, $p_{SP,l}$ 是状态 l 的概率。

(4) 将系统的可靠度 R 定义如式(38)所示

$$R = \sum_{\{l: g_{SP,l} \geq D_0\}} p_{SP,l} \quad (38)$$

式中, $g_{SP,l} \geq D_0$ 代表系统的性能满足给定需求性能 D_0 。

用 MFBA-MAPACO 混合算法对串-并联多态系统冗余分配问题进行优化求解, 同时与文献[28]PSO/LS 算法和文献[25]TFPSO 算法进行对比, 其中, PSO/LS 算法是文献[28]提出的算法——基于微粒群算法和局部搜索(Local search, LS)的混合算法, 文献[28]中, PSO/LS 算法已被验证优于相关文献的蚁群算法、遗传算法、禁忌搜索等算法。设置 MFBA-MAPACO 算法迭代次数为 500, 蝙蝠个数、蚂蚁个数为 20, 算法切换停滞阈值为 50, 对上述三种不同可靠度约束的优化问题求解, 算法独立运行 20 次, 求得的最优解见表 9。

表 9 求得的最优解

算法	可靠度 约束 R_0	可靠度 R	费用 C	优化结果 X
TFPSO ^[25]		0.993	7.516	[0,0,0,1,1; 0,0,2,0; 3,0,0,0,0,0; 3,0,0,0,2]
PSO/LS ^[28]	0.990	0.992	8.180	[3,0,0,0,0; 0,0,3,0; 3,0,0,0,0,0; 0,0,1,2,0]
MFBA- MAPACO		0.995	7.232	[0,0,0,1,1; 0,0,2,0; 3,0,0,0,0,0; 3,0,1,0,0]
TFPSO ^[25]		0.970	6.440	[2,1,0,0,0; 0,2,0,0; 2,0,0,0,0,0; 0,0,0,0,2]
PSO/LS ^[28]	0.960	0.963	7.009	[3,0,0,0,0; 0,1,2,0; 3,0,0,0,0,0; 0,0,1,0,1]
MFBA- MAPACO		0.982	6.225	[2,0,0,0,1; 0,2,0,0; 2,0,0,0,0,0; 0,0,0,2,0]
TFPSO ^[25]		0.903	5.408	[1,1,0,0,0; 0,1,1,0; 2,0,0,0,0,0; 0,0,1,0,1]
PSO/LS ^[28]	0.900	0.901	5.423	[0,0,0,1,0; 0,0,2,0; 3,0,0,0,0,0; 0,0,1,0,1]
MFBA- MAPACO		0.904	5.285	[1,0,1,0,0; 0,1,1,0; 2,0,0,0,0,0; 0,0,1,0,1]

从表 9 可以看出, 这三种不同的可靠度约束值下, 所提 MFBA-MAPACO 算法都有最优解, 优于文献[25]和[28]算法。以 $R_0=0.990$ 时的优化结果为例, 给出如图 6 所示的由所提算法确定的最优系统结构。

图 6 最优系统结构

5 结论

(1) 针对蝙蝠算法飞行方向只受单一作用力,易陷入局部最优解等不足,提出了多形态作用力蝙蝠算法,把算法划分为前期和后期进行阶段搜索,在搜索前期提高搜索范围的多样性,提高算法的全局搜索能力;在搜索后期,让蝙蝠受到动力加速度和中值导向加速度的双引力作用,使蝙蝠更好地趋于全局最优解。测试验证了所提算法有更强的优化能力,能够满足寻优搜索需求。

(2) 针对蚁群算法信息素更新机制单一,易早熟收敛的不足,结合蚁群社会活动,提出了全面调整机制的多阶段自适应信息素机制蚁群算法。在出现长时间搜索停滞时,通过引入混沌算子,帮助跳出早熟收敛,达到了扩展蚁群的搜索范围、防止蚁群过早停滞、优化蚁群全局搜索能力的效果。结合旅行商问题测试对比,验证了所提算法具有更好的寻优精度和稳定性。

(3) 针对上述所提的多形态作用力蝙蝠算法具有的全局搜索能力强、收敛速度快的优势,以及多阶段自适应信息素机制蚁群算法具有的局部精细化能力强的优势,将上述两种智能算法串行混合,提出了多阶段自适应蝙蝠-蚁群混合群智能算法,进而用于液压系统可靠性优化问题和串-并联多态系统复杂的可靠性优化问题中,并与其他算法优化结果对比,验证了所提混合群智能算法的有效性和更强的寻优能力。

参考文献

- [1] 徐华,张庭.混合离散蝙蝠算法求解多目标柔性作业车间调度[J].机械工程学报,2016,52(18):201-212.
XU Hua, ZHANG Ting. Hybrid discrete bat algorithm for solving the multi-objective flexible job shop scheduling problem[J]. Journal of Mechanical Engineering, 2016, 52(18): 201-212.
- [2] JADDI N S, ABDULLAH S, HAMDAN A R. Optimization of neural network model using modified bat-inspired algorithm[J]. Applied Soft Computing, 2015, 37: 71-86.
- [3] GANDOMI A H, YANG X S, ALAVI A H, et al. Bat algorithm for constrained optimization tasks[J]. Neural Computing & Applications, 2013, 22(6): 1239-1255.
- [4] XIAO L, QIAN F, SHAO W. Multi-step wind speed forecasting based on a hybrid forecasting architecture and an improved bat algorithm[J]. Energy Conversion and Management, 2017, 143: 410-430.
- [5] BEN A M S, SAKLY A. FPGA based hardware implementation of bat algorithm[J]. Applied Soft Computing, 2017, 58: 378-387.
- [6] CHAKRI A, KHELIF R, BENOURET M, et al. New directional bat algorithm for continuous optimization problems[J]. Expert Systems with Applications, 2017, 69: 159-175.
- [7] YILMAZ S, KÜÇÜKSİLLE E U. A new modification approach on bat algorithm for solving optimization problems[J]. Applied Soft Computing, 2015, 28: 259-275.
- [8] 谢健,周永权,陈欢.一种基于 Lévy 飞行轨迹的蝙蝠算法[J].模式识别与人工智能,2013,26(9):829-837.
XIE Jian, ZHOU Yongquan, CHEN Huan. A bat algorithm based on Lévy flights trajectory[J]. Pattern Recognition and Artificial intelligence, 2013, 26(9): 829-837.
- [9] HE X, DING W J, YANG X S. Bat algorithm based on simulated annealing and Gaussian perturbations[J]. Neural Computing & Applications, 2014, 25(2): 459-468.
- [10] 陈梅雯,钟一文,王李进.一种求解多维全局优化问题的改进蝙蝠算法[J].小型微型计算机系统,2015,36(12):2749-2753.
CHEN Meiwen, ZHONG Yiwen, WANG Lijin. Improved bat algorithm for solving multi-dimensional global optimization problems[J]. Journal of Chinese Computer Systems, 2015, 36(12): 2749-2753.
- [11] BONABEAU E, DORIGO M, THERAULAZ G. Inspiration for optimization from social insect behaviour[J]. Nature, 2000, 406: 39-42.
- [12] 宿彪,黄向明,任莹晖,等.基于蚁群算法的工程机械再制造优化选配方法研究[J].机械工程学报,2017,53(5):60-68.
SU Biao, HUANG Xiangming, REN Yinghui, et al. Research on selective assembly method optimization for construction machinery remanufacturing based on ant colony algorithm[J]. Journal of Mechanical Engineering, 2017, 53(5): 60-68.
- [13] AL-SHIHABI S. A hybrid of max-min ant system and linear programming for the k-covering problem[J]. Computers & Operations Research, 2016, 76: 1-11.
- [14] 吴长友,戴宁,刘浩,等.基于改进蚁群算法的分形刀轨连接技术[J].中国机械工程,2016,27(1):58-65.
WU Changyou, DAI Ning, LIU Hao, et al. Fractal tool path connection technology based on improved ant colony algorithm[J]. China Mechanical Engineering, 2016, 27(1): 58-65.
- [15] 马文龙,王铮,赵燕伟.基于改进蚁群算法的制造云服务组合优化[J].计算机集成制造系统,2016,22(1):113-121.

- MA Wenlong, WANG Zheng, ZHAO Yanwei. Optimizing services composition in cloud manufacturing based on improved ant colony algorithm[J]. Computer Integrated Manufacturing Systems, 2016, 22(1): 113-121.
- [16] LIN Y, GONG Y J, ZHANG J. An adaptive ant colony optimization algorithm for constructing cognitive diagnosis tests[J]. Applied Soft Computing, 2017, 52: 1-13.
- [17] 张铁, 苏杰汶. 基于改进蚁群算法的机器人末端路径排序优化[J]. 中国机械工程, 2016, 27(19): 2624-2629. ZHANG Tie, SU Jiewen. Path sorting optimization of robotic end-effector by improved ACA[J]. China Mechanical Engineering, 2016, 27(19): 2624-2629.
- [18] 黄凤立, 左春桢, 顾金梅, 等. 基于加工操作单元的多态蚁群装夹规划方法[J]. 机械工程学报, 2017, 53(7): 164-172. HUANG Fengli, ZUO Chuncheng, GU Jinmei, et al. Polymorphic ant colony clamping planning method based on the machining operation unit[J]. Journal of Mechanical Engineering, 2017, 53(7): 164-172.
- [19] 赵诗奎. 基于新型邻域结构的混合算法求解作业车间调度[J]. 机械工程学报, 2016, 52(9): 141-150. ZHAO Shikui. A hybrid algorithm with a new neighborhood structure for the job shop scheduling problem[J]. Journal of Mechanical Engineering, 2016, 52(9): 141-150.
- [20] ADARSH B R, RAGHUNATHAN T, JAYABARATHI T, et al. Economic dispatch using chaotic bat algorithm[J]. Energy, 2016, 96: 666-675.
- [21] 陈东宁, 张瑞星, 姚成玉. 基于混合 PSO-ACO 算法的液压系统可靠性优化[J]. 机床与液压, 2013, 41(23): 157-161. CHEN Dongning, ZHANG Ruixing, YAO Chengyu. Reliability optimization of hydraulic system based on hybrid PSO-ACO algorithm[J]. Machine Tool & Hydraulics, 2013, 41(23): 157-161.
- [22] DONG G F, FU X L, LI H H, et al. Cooperative ant colony-genetic algorithm based on spark[J]. Computers and Electrical Engineering, 2016, 60: 66-75.
- [23] 谢颖, 李吉兴, 杨忠学, 等. 改进遗传蚁群算法及其在电机结构优化中的研究[J]. 电机与控制学报, 2015, 19(10): 64-70. XIE Ying, LI Jixing, YANG Zhongxue, et al. Improved genetic ant colony algorithm and research on motor structure optimization[J]. Electric Machines and Control, 2015, 19(10): 64-70.
- [24] 李枝勇, 马良, 张慧珍. 蝙蝠算法收敛性分析[J]. 数学的实践与认识, 2013, 43(12): 182-190. LI Zhiyong, MA Liang, ZHANG Huizhen. Convergence analysis of bat algorithm[J]. Mathematics in Practice and Theory, 2013, 43(12): 182-190.
- [25] YAO C Y, TAN X Y, CHEN D N, et al. Two-stage force particle swarm optimization algorithm and application in hydraulic system reliability optimization[C]//International Conference on Fluid Power and Mechatronics, August 5-7, 2015, Harbin. IEEE, 2015: 1291-1296.
- [26] PANDIT M, CHAUDHARY V, DUBEY H M, et al. Multi-period wind integrated optimal dispatch using series PSO-DE with time-varying Gaussian membership function based fuzzy selection[J]. Electrical Power and Energy Systems, 2015, 73: 259-272.
- [27] QIN H S, ZHOU S L, HUO L, et al. A new ant colony algorithm based on dynamic local search for TSP[C]//Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies, April 4-6, 2015, Gwalior. IEEE, 2015: 913-917.
- [28] WANG Y, LI L. Heterogeneous redundancy allocation for series-parallel multi-state systems using hybrid particle swarm optimization and local search[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2012, 42(2): 464-474.
-
- 作者简介: 陈东宁(通信作者), 女, 1978 年出生, 博士, 副教授。主要研究方向为可靠性分析及优化。
E-mail: dncchen@ysu.edu.cn
- 刘一丹, 女, 1992 年出生。主要研究方向为群智能算法改进与应用。
- 姚成玉, 男, 1975 年出生, 博士后, 教授。主要研究方向为系统可靠性及故障诊断。
- 杨晓荣, 女, 1993 年出生。主要研究方向为群智能算法改进与应用。