DOI: 10.13382/j.jemi.B2003593

基于历史认知的鲸鱼算法求解动态能耗*

罗 钧[1] 庞亚男[1,2] 刘建强[1]
(1. 重庆大学光电技术及系统教育部重点实验室 重庆 400030;2. 四川航天电子设备研究所 成都 610100)

摘 要:为提高嵌入式实时系统的能耗管理效率,降低传统动态电压缩放对系统稳定性的影响,提出了基于历史认知的鲸鱼算法支持下的动态能耗优化方案。首先提出非线性动态控制收敛因子的策略,有效加快了算法收敛速度。其次采用历史最优解作为收缩包围机制中的种群干扰因子,设计了混合引导策略来平衡算法的局部开发和全局搜索能力。最后根据动态电压缩放技术可以实时改变处理器频率的特征,利用改进算法对任务量 10、30 和 50 进行优化,验证了改进算法的有效性。

关键词: 能耗管理;动态电压缩放;历史认知;非线性收敛因子;改进鲸鱼算法
中图分类号: TP1; TN702 **文献标识码**: A **国家标准学科分类代码**: 510.50

Whale algorithm based on historical cognition for solving dynamic energy consumption

Luo Jun[1] Pang Yanan[1,2] Liu Jianqiang[1]
(1. Key Laboratory of Optoelectronic Technology and System of Ministry of Education, Chongqing University, Chongqing 400030, China; 2. Sichuan Aerospace Electronic Equipment Research Institute, Chengdu 610100, China)

Abstract: In order to improve the energy consumption management efficiency of the embedded real-time system and reduce the impact of traditional dynamic voltage scaling technology on system stability, a dynamic energy consumption optimization scheme supported by whale algorithm based on historical cognition is proposed. Firstly, a nonlinear dynamic convergence factor control strategy is proposed, which can effectively accelerate the convergence speed of the algorithm. Secondly, using the historical optimum solutions as interference factors, a hybrid guided strategy is designed in the constriction and envelopment mechanism to balance the local development and global search capability of the algorithm. Finally, the frequency characteristics of the processor can be changed in real time according to the dynamic voltage scaling technology, the tasks 10, 30 and 50 are optimized by the algorithm, so as to verify the effectiveness of the improved algorithm.

Keywords: energy consumption management; dynamic voltage scaling; historical cognition; nonlinear convergence factor; improved whale algorithm

0 引 言

随着芯片集成度不断提高,传统的只考虑性能的优化调度方法已不适用于对可靠性和低能耗等性能综合优化的情形,迫切需要研究新的能耗调度方案和优化算法。由于系统功耗增加将导致运行时间和稳定性大幅降低,且温度每升高 10 ℃将导致 50%的组件寿命损失[1],因此高效的电力管理方案对系统寿命和可靠性起到至关重要的作用。动态电压缩放(dynamic voltage scaling, DVS)作为一种高效的能量管理技术[2-3],通过在程序运行时动态降低处理器的供电电压或时钟频率来减少功耗,但电压或频率过低又将导致执行任务的时间变长、故障率增加等。所以平衡处理器电压、频率和执行任务时间的关系是优化能耗管理、保障实时系统可靠性的重要环节,成为了学者的研究热点。

随着仿生学的发展,具有较强全局最优搜索能力的群智能算法孕育而生。文献[4]开发了一种基于二进制

收稿日期: 2020-10-30 Received Date: 2020-10-30
*基金项目:国防科工局十二五技术基础科研项目(JSJL2014209B005)、工信部“两机”重大专项基础研究项目(Z20210208)资助

粒子群的通信优化算法用于嵌入式多线程代码生成器，以生成高质量的代码。文献[5]的 ILP 算法利用基于离散域的位置更新方法和迁移机制实现任务分配。然而，这些算法都单方面侧重于系统性能或能耗的提升，对系统性能至关重要的受实时性、可靠性约束的能耗管理却很少被考虑。文献[6]采用改进鸟群算法(IoBSA)实现动态能耗管理，但从数据分析，该方案的优化精度没有达到较好的结果。综上所述，研究兼顾可靠性和实时性的能耗优化管理算法成为了急需解决的问题。

鲸鱼优化算法(whale optimization algorithm, WOA)是 2016 年 Mirjalili 等[7]模拟鲸鱼的觅食行为提出的一种启发式[8]算法。与差分进化(differential evolution, DE)算法[9]和粒子群优化(particles warm optimization, PSO)算法[10]等经典的群智能优化算法相比，WOA 具有探测性强、调节参数少和收敛精度高等优势。尽管如此，从已有研究可以发现，WOA 算法的优化精度有提升空间。因此，近年来有不少学者尝试从不同角度改进 WOA，使其成功应用于各种工程优化问题，如电容器在径向分布网络中的最佳位置[11]，与能源相关的二氧化碳排放预测[12]等。

基于上述分析，本文提出一种基于历史认知改进的鲸鱼优化算法(IWOAHC)支持下的动态能耗管理方案，在保证测试函数收敛精度的前提下，兼顾大规模优化问题的应用效果。因此，提出非线性动态控制收敛因子策略，有效加快了算法的收敛速度。并采用历史认知解作为种群的空间结构，在收缩包围机制中，设计了一种混合引导策略来平衡算法的局部开发和全局搜索能力。通过选取文献[7]的 12 个国际标准测试函数，包括单峰和多峰函数，对改进算法进行测试。最后，为了平衡动态电压缩放技术中的稳定性、实时性和能耗指标，建立了 DVS 动态能耗管理模型，并把改进的算法应用到动态能耗管理中。实验结果表明基于历史认知的混合引导策略对算法收敛起到了良好的促进作用，具有一定的普适性，且该算法在实时性和可靠性约束下，能高效、快速地优化系统能耗。

1　DVS 能耗管理的原理分析

1.1　能耗模型的建立

对于基于 CMOS 技术的处理器，它的总能耗分为动态和静态能耗。静态能耗主要由处理器的工艺决定，是处理器的必要能耗；动态能耗则是处理器执行任务的过程中产生的能耗，也是主要的能耗点，因此需要通过调节动态能耗以达到节能的目的。动态功耗 P_i 由电压和频率确定，可以用以下数学模型表示：

$$P_i = C_{ef} V_{dd}^2 f_i \tag{1}$$

式中：V_{dd} 是电源电压[13]；C_{ef} 是有效开关电容；f_i 为处理器时钟频率。处理器的动态能耗根据等式 $E_i = P_i \cdot t_i$ (t_i 表示在频率 f_i 下的执行时间)，可以由式(2)计算。

$$E_i = C_{ef} V_{dd}^2 C_i \tag{2}$$

式中：C_i 为最坏情况执行时间(worst case execution time, WCET)下任务的周期数。由于处理器速度几乎随电源电压线性变化，$f_i = k \cdot (V_{dd} - V_t)^2 / V_{dd}$，其中 k 为大于 0 的常数，V_t 为阈值电压[13]。因此 V_{dd} 可通过 f_i 由式(3)给出。

$$V_{dd} = V_t + \frac{f_i}{2k} + \sqrt{\frac{V_t f_i}{k} + \frac{f_i^2}{4k^2}} \tag{3}$$

可以通过 f_i 进一步推导相应处理器的动态能耗 E_i，表达式为：

$$E_i(f_i) = C_{ef} C_i \left(\frac{f_i^2}{2k^2} + \frac{2v_t f_i}{k} + V_t^2 + \left(V_t + \frac{f_i}{2k}\right) \sqrt{\frac{4V_t f_i}{k} + \frac{f_i^2}{k^2}} \right) \tag{4}$$

为了简化又不失一般性，本文设置了 $C_{ef} = 1$，$k = 1$ 和 $V_t = 1$，由于它们是常数，对优化模型没有负面影响。所以最终的电源模块能耗表示为：

$$E_i(f_i) = C_i \left(\frac{f_i^2}{2} + 2f_i + 1 + \left(1 + \frac{f_i}{2}\right) \sqrt{4f_i + f_i^2} \right) \tag{5}$$

1.2　实时性分析

假设一组待执行的实时任务 $\{T_1, T_2, \cdots, T_n\}$，它们之间相互独立，具有各自的执行周期。每个任务 T_i 都可以被描述为 $T_i = (P_i, S_i, D_i, C_i)$，$P_i$ 表示任务的执行周期，S_i 为任务的就绪时间，同时任务被要求在截止时间 D_i 内完成。此外，最坏情况执行时间 C_i 用于评估任务 T_i 在极端频率 $f_{\max}$ 下的执行情况。即每个任务必须在下个任务到达前完成，才能满足最基本的实时性的要求。通常情况下，确定任务频率 f_i 后，理论上全速运行的核心需要 $C_i f_{\max} / f_i$ 时间才能完成任务 T_i。为保证该任务能在截止时间 D_i 内完成，并且满足嵌入式系统实时性要求，动态频率实时性约束数学模型为(其中 $f_{\max}$ 归一化到 1)：

$$D_i - \frac{C_i}{f_i} \geqslant 0 \tag{6}$$

1.3　故障率分析

实时系统执行任务中，难免因电磁干扰、软件错误和硬件故障等原因发生故障，这种暂态故障往往会导致任务失败。系统的稳定性定义为在外界因素影响下仍保持正常运行的能力，显然故障率是导致稳定性降低的重要因素。本文研究假设暂态故障满足泊松分布，执行不同任务时发生的暂态故障彼此独立，故障率的表达模型为：

$$\lambda(f_i) = \lambda_0 \cdot 10^{\frac{d(1-f_i)}{1-f_{\min}}}, d > 0 \tag{7}$$

其中，$\lambda_0 = 10^{-6}$[13] 为平均故障率，从式(7)可以看到，故障率与处理器运行频率是负相关的，频率越低，故障发生率越高，当 $f_i = f_{\min}$ 时故障率取最大值 $\lambda_0 10^d$。传统 DVS 单纯地通过降低处理器运行频率来降低能耗，带来的负面影响是暂态故障频繁发生，系统稳定性急剧下降，这种以稳定性换取最低能耗的方案显然是有缺陷的。

1.4　能耗优化管理问题描述

能耗管理问题的核心是实时优化频率分配方案，以减少周期性任务的功耗，达到动态调节能耗的效果。因此需要建立执行时间和故障率的数学模型来保证在最小能耗情况下的系统性能。假设处理器时钟频率 $f_i \in (f_{\min}, f_{\max})$，并且 $f_{\max}$ 被归一化为 1。对于一组独立的周期性实时任务 $\{T_1, T_2, \cdots, T_n\}$，为满足实时性要求，能量 E 需要在式(8)给定的周期内完成所有任务：

$$E = \sum_{i=1}^{n} E_i(f_i) \tag{8}$$

本文用新的调度算法实现能耗的最小耗损，所以问题的极小值描述为：

$$\min \sum_{i=1}^{n} E_i(f_i) \tag{9}$$

且需满足式(10)对应的实时性约束 g_1、式(11)形成的系统稳定性约束 g_2 和式(12)可行频率范围限制。

$$g_1 = D_i - \frac{C_i}{f_i} \geqslant 0 \tag{10}$$

$$g_2 = \lambda_0 10^{d(1-f_i)/(1-f_{\min})} - \lambda_k \leqslant 0 \tag{11}$$

$$f_{\min} \leqslant f_i \leqslant f_{\max}, 1 \leqslant i \leqslant n \tag{12}$$

式中：D_i 表示任务 T_i 的截止日期；λ_0 是对应于最大频率 $f_{\max} = 1$ 时的平均故障率；λ_k 是一个常数，可设置在 $[\lambda_0, \lambda_0 10^d]$。指数 $d(d>0)$ 是一个测得的常数，表明故障率对电压和频率缩放的敏感性。最大故障率 $\lambda_0 10^d$ 出现在最小频率 $f_{\min}$。然而最小功耗也出现在最小频率，因此需要通过优化处理器频率来降低故障率，提高可靠性并且满足减小系统功耗的需求。

为了最大程度降低系统功耗的同时，使所有的任务均满足可靠性和实时性约束，根据惩罚函数[14-15]的建立规则，将动态能耗管理转换为无约束的全局优化问题，最终能耗管理问题中待优化的目标函数 f_{obj} 模型如下：

$$f_{obj} = \min \sum_{i=1}^{n} E_i(f_i) + B \cdot | \min(g_1, 0) | + B \cdot | \max(g_2, 0) | \tag{13}$$

其中，B 为惩罚因子，决定算法对违反约束解的惩罚力度。最小值求解中，B 越大使不可用解远离最优解的程度越大，进而被算法自动剔除。但是过大的 B 会增加一定的计算量，因此，本文 B 取值为 5 000。并利用优化问题对不精确性和不确定性的容忍度，通过改进的 IWOAHC 算法对目标函数 f_{obj} 进行寻优。

2　IWOAHC 算法

2.1　WOA 算法

鲸鱼算法是一种模拟座头鲸觅食行为的优化算法，具体包含 3 个阶段：搜索猎物、包围猎物和环绕捕食。WOA 算法中，假设种群数量为 N，搜索空间为 d 维，式子 $\overrightarrow{X_i} = (X_i^1, X_i^2, \cdots X_i^d), i = 1,2,\cdots,N$，代表第 i 头鲸鱼在 d 维空间中的位置，而目标猎物的位置对应于优化问题的全局最佳解，WOA 的基本原理如下。

1)收缩包围机制

由于座头鲸能够感知彼此的位置，因此 WOA 算法将当前最佳候选解作为目标猎物或逼近目标的位置。假设当前最优位置为 $\overrightarrow{X^*} = (X^{1*}, X^{2*}, \cdots, X^{d*})$，包围阶段是所有个体接近最优解的过程。这种行为表现为：

$$\overrightarrow{X}(t+1) = \overrightarrow{X^*}(t) - A \cdot \overrightarrow{D} \tag{14}$$

式中：t 为当前迭代次数；$\overrightarrow{X^*}(t)$ 为此时最佳解的位置；$\overrightarrow{D} = (D^1, D^2, \cdots, D^d), D^j = | C \cdot \overrightarrow{X^{j*}}(t) - \overrightarrow{X^j}(t) |, j = 1,2,\cdots,d$，表示当前位置在第 j 维空间与最佳搜索代理间的距离，$|\cdot|$ 是绝对值运算。如果位置 $\overrightarrow{X}(t+1)$ 比 $\overrightarrow{X^*}(t)$ 更优，$\overrightarrow{X^*}(t)$ 便会更新当前位置。A 和 C 的表达式如下：

$$a = 2 - \frac{2t}{T_{\max}} \tag{15}$$

$$A = 2a \cdot r - a \tag{16}$$

$$C = 2 \cdot r \tag{17}$$

式中：r 是 $[0,1]$ 内的随机数；$T_{\max}$ 为最大迭代次数；a 随迭代次数由 2 线性递减为 0。

2)搜索机制

座头鲸会根据彼此的位置随机寻找猎物。A 的值与座头鲸的搜索方式密切相关，当 $|A| \geqslant 1$ 时，WOA 通过随机选择来更新搜索代理的位置，此约束机制使 WOA 在这个阶段有机会探索更多潜在的未知领域，表现为全局搜索。其数学模型描述如下：

$$\overrightarrow{X}(t+1) = \overrightarrow{X}_{rand}(t) - A \cdot \overrightarrow{D}_{rand} \tag{18}$$

式中：$\overrightarrow{X}_{rand}(t)$ 为父代种群中随机的位置；$\overrightarrow{D}_{rand} = (D_{rand}^1, D_{rand}^2, \cdots, D_{rand}^d), D_{rand}^j = | C \cdot \overrightarrow{X_{rand}^j}(t) - \overrightarrow{X^j}(t) |, j = 1,2,\cdots,d$，表示当前位置在第 j 维空间与随机搜索代理间的距离，$|\cdot|$ 是绝对值运算。

3)气泡网攻击机制

通过模拟座头鲸气泡网狩猎行为，WOA 设置了收缩包围和螺旋更新两种策略。其中收缩包围策略通过改变式(15)和(16)中收敛因子 a 的值实现，当 $|A| < 1$ 时，算法根据式(14)实现对猎物的收缩包围。当采用螺旋方

式更新位置时,数学模型表示如下:

$$\vec{X}(t+1)=\vec{D'}\cdot e^{bl}\cdot\cos(2\pi l)+\vec{X^*}(t) \tag{19}$$

式中,b 为常量系数;l 为[-1,1]内的随机数;$\vec{D'}=(D^{1'},D^{2'},\cdots D^{d'})$,$D^{j'}=|X^{j*}(t)-X^j(t)|$,$j=1,2,\cdots,d$,表示在第 j 维空间座头鲸和猎物位置间的距离。由于座头鲸沿着螺旋路径运动的同时收缩包围圈,因此 WOA 算法通过相同概率(均为 0.5)[7]来模拟收缩包围和螺旋更新的同步过程。

2.2　IWOAHC 算法原理

1)非线性收敛因子

全局搜索和局部开发是基于群体迭代进化的启发式优化算法的共同特点。全局搜索代表群体需在未知领域扩大进化范围,使群体保持良好的多样性,避免出现早熟。局部开发意味着群体要在有开发潜力的空间精确探索,提高算法的收敛精度。由对文献[7]和式(14)的分析可知,参数 A 在标准 WOA 中对调节算法的全局探索和局部开发能力起到至关重要的作用,而参数 A 的变化与 a 相关(式(16)),这意味着 WOA 算法的全局探索能力和局部开发能力的平衡与 a 的取值密切相关。原始 WOA 中,a 从 2 到 0 线性递减,算法优化早期,a 的取值较大有助于扩大搜索空间,避免早熟;算法优化后期,a 取值较小有利于实现局部精准开发,提高收敛精度。

然而,通过研究分析发现随着迭代的不断进行,鲸鱼算法的进化搜索过程是非线性的,因此,参数 a 的线性变化过程不能准确体现出算法的整个进化搜索过程。受 PSO 中惯性权重引导种群进化思想的启发[10],本文设计了一种收敛因子 a 随迭代次数非线性变化的方案:

$$a=2-\frac{4t}{T_{\max}}+\frac{2t^2}{T_{\max}^2} \tag{20}$$

由式(20)可知,在算法迭代的早期,a 取值较大并且下降速度快,有助于加快收敛速度;在算法迭代后期,a 取值较小且下降速度慢,有助于精准开发并提高收敛精度,从而平衡好 IWOAHC 算法的全局搜索和局部开发的能力。这种平衡全局搜索和局部搜索的设计思路,使该算法尤为适用于沿着自变量方向会产生大量的局部极值,具有较高寻优难度的函数。本文研究将这种改进方法称为基于非线性收敛因子的鲸鱼算法(IWOA)。

2)基于历史认知的混合引导策略

标准 WOA 在算法进化后期,随着收敛因子 a 的减小,整个种群的进化不断向本次迭代的最优个体靠近,如果该个体不是全局最优解,那么整个种群极易向局部最优解靠近,导致算法早熟,陷入局部最优,降低收敛精度。此外,所有个体向最佳个体靠近时并未考虑种群的“历史”[16]寻优轨迹,极易导致潜在的最优解由于单纯的向最优个体聚集的行为被忽略,造成算法错过全局最优,陷入局部最优。

本文受到生物种群捕食过程中在向最优个体学习的同时兼顾种群的“历史”轨迹,考虑过去对现在的影响的启发,提出一种基于历史认知的混合引导进化策略,在收缩包围机制中保留向最优个体学习的同时增加向“历史”的学习行为。并且引入调节因子 ω,在寻优过程中实时调整种群的“历史”对本次进化的价值。其表达式及收缩包围机制中个体更新方程为:

$$\begin{aligned}&\vec{X_1}(t+1)=\vec{X^*}(t)-A\cdot\vec{D}\\&\vec{X_2}(t+1)=\vec{X_2^*}(t)-A\cdot\vec{D_2}\\&\vec{X}(t+1)=(1-\omega)\cdot\vec{X_1}(t+1)+\omega\cdot\vec{X_2}(t+1)\end{aligned} \tag{21}$$

式中:$\vec{X^*}(t)$ 为此时最佳解的位置;$\vec{D}=(D^1,D^2,\cdots,D^d)$,$D^j=|C\cdot\vec{X^{j*}}(t)-\vec{X^j}(t)|$,$j=1,2,\cdots,d$;$\vec{X_2^*}(t)$ 为上一次迭代中最优解;$\vec{D_2}=(D_2{}^1,D_2{}^2,\cdots,D_2{}^d)$,$D_2{}^j=|C\cdot\vec{X_2^{j*}}(t)-\vec{X_2^j}(t)|$,$j=1,2,\cdots,d$;$\vec{X_1}(t+1)$、$\vec{X_2}(t+1)$、$\vec{X}(t+1)$ 分别为向当前最优个体学习、向历史最优个体学习和综合历史最优和当前最优的结果。ω 为常数。此外由式(21)可知该算法的核心并非单纯的向最优个体学习来提高收敛速度,而是兼顾整个种群的迭代进化,通过调节因子 ω 使种群的历史有一定的参考意义,逐渐增强对“历史”的学习,通过迭代的方式衡量“历史”轨迹对本次进化的价值,优化寻优路径,增强局部开发能力,提高收敛精度。本文中 ω 的取值为 0.5。本文将这种改进策略称为 WOAHC 算法。

综上所述,本文提出的 IWOAHC 算法如下:

1)参数初始化,设置种群规模 N,最大迭代次数 $T_{\max}$;

2)在解空间内随机产生初始的种群,令 $t=0$;

3)计算种群中每个个体的适应度值 $f(x_i)$,并记录当前最优个体及其位置;

4)如果 $t<T_{\max}$;

5)根据式(20)计算收敛因子 a,并根据式(16)、(17)计算 A、C 的值,并产生[0,1]的随机数 p;

6)$p<0.5$ 时,若 $|A|<1$,根据式(21)更新个体位置;若 $|A|\geqslant 1$,根据式(18)更新个体位置;

7)$p\geqslant 0.5$ 时,根据式(19)更新个体位置;

8)记录上代种群产生的最优个体,作为“历史”最优解,并计算当前个体的适应度,更新当前最优解;

9)$t=t+1$。

3　实验与结果分析

为验证 IWOAHC 算法的有效性和通用性,将算法与

原始 WOA[7]、PSO[10]、LSHADE[17]通过基准测试函数进行对比测试。动态能耗管理性能的验证增加了文献[6]提出的改进鸟群算法(IoBSA)对比实验,旨在验证基于IWOAHC算法的能耗管理对比现有能耗管理方案的优越性。本文算法仿真实验平台为 Windows 10(64 位)操作系统、8 GB 内存、i3-2120,编程软件采用 MATLAB R2014a。

表 1　基准测试函数

Table 1　Benchmark test functions

函数	特征	函数表达式	范围	最优值
Sphere	U	$F_1 = \sum_{i=1}^{n} x_i^2$	[-100,100]	0
Schwefel 2.22	U	$F_2 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10,10]	0
Schwefel 2.21	U	$F_3 = \max\{\lvert x_i \rvert, 1 \leqslant i \leqslant n\}$	[-100,100]	0
Schwefel 1.2	U	$F_4 = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	[-100,100]	0
Quartic	U	$F_5 = \sum_{i=1}^{n} i x_i^4 + rand(0,1)$	[-1.28,1.28]	0
Rastrigin	M	$F_6 = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	[-5.12,5.12]	0
Ackley	M	$F_7 = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	[-32,32]	0
Griewank	M	$F_8 = \frac{1}{4\,000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600,600]	0
Alpine	M	$F_9 = \sum_{i=1}^{n} (\lvert x_i \cdot \sin(x_i) + 0.1x_i \rvert)$	[-10,10]	0
Rosenbrock	M	$F_{10} = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2) + (x_i - 1)^2]$	[-30,30]	0
Schwefel 2.26	M	$F_{11} = \sum_{i=1}^{n} [-x_i \sin(\sqrt{\lvert x_i \rvert})]$	[-500,500]	-418.982 9×30
Zakharov	M	$F_{12} = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$	[-5,10]	0

表 2　与经典智能算法性能对比

Table 2　Compare the performance of classical intelligent algorithms

函数	指标	WOA	LSHADE	PSO	IWOAHC	函数	指标	WOA	LSHADE	PSO	IWOAHC
F_1	Mean	6.43×10^{-73}	1.89×10^{-50}	9.33×10^{-11}	0.00	F_2	Mean	1.43×10^{-50}	4.65×10^{-26}	2.09×10^{-05}	2.14×10^{-316}
	Std	2.92×10^{-72}	5.35×10^{-50}	8.16×10^{-12}	0.00		Std	7.73×10^{-49}	6.99×10^{-26}	7.34×10^{-05}	6.14×10^{-315}
F_3	Mean	30.4648	6.78×10^{-4}	3.36×10^{-1}	3.45×10^{-271}	F_4	Mean	1.13×10^{4}	8.77×10^{-7}	5.05	0.00
	Std	26.2435	1.02×10^{-3}	1.07×10^{-1}	7.12×10^{-270}		Std	6.02×10^{3}	2.52×10^{-6}	2.27	0.00
F_5	Mean	5.56×10^{-2}	2.07×10^{-3}	4.38×10^{-2}	1.90×10^{-5}	F_6	Mean	1.89×10^{-15}	2.84×10^{-16}	3.97×10^{1}	0.00
	Std	6.08×10^{-2}	5.77×10^{-3}	1.94×10^{-2}	6.66×10^{-4}		Std	1.03×10^{-14}	6.57×10^{-15}	1.06×10^{1}	0.00
F_7	Mean	4.09×10^{-15}	9.30×10^{-15}	1.04×10^{-6}	8.88×10^{-16}	F_8	Mean	1.17×10^{-2}	6.57×10^{-4}	7.55×10^{-3}	0.00
	Std	2.53×10^{-15}	3.30×10^{-15}	1.37×10^{-6}	1.64×10^{-15}		Std	4.46×10^{-2}	2.58×10^{-3}	8.52×10^{-3}	0.00
F_9	Mean	6.86×10^{-79}	7.85×10^{-12}	9.25×10^{-5}	1.31×10^{-316}	F_{10}	Mean	2.72×10^{1}	1.70	4.21×10^{1}	2.64×10^{1}
	Std	3.76×10^{-78}	3.00×10^{-11}	4.55×10^{-4}	4.72×10^{-315}		Std	3.45×10^{-1}	1.62	3.47×10^{1}	3.13×10^{-1}
F_{11}	Mean	-9.16×10^{3}	-1.18×10^{4}	-2.45×10^{3}	-1.25×10^{4}	F_{12}	Mean	4.60×10^{2}	6.93×10^{-9}	9.63	0.00
	Std	1.42×10^{3}	1.13×10^{3}	4.72×10^{2}	5.46×10^{1}		Std	8.64×10^{1}	1.86×10^{-8}	4.02	0.00

3.1　基准函数对比试验结果分析

为了有效验证提出的 IWOAHC 算法的优化性能,本文通过表 1 中的 30 维测试函数进行验证,其中 $F_1 \sim F_5$ 表示单模态函数,$F_6 \sim F_{12}$ 表示复杂的非线性多模态函数。单模态函数只有一个显著的峰值,可以有效验证算法的收敛能力;多模态函数存在多个局部最优峰值,可以有效验证算法的全局搜索能力。其他对比算法参数的设置均与相应原始文献相同。为了客观对比,所有算法种群规模设为 30,迭代次数为 500,独立运行 30 次。采用最优解的平均值(mean)和标准差(std)作为性能评价依据,测试结果如表 2 所示。此外,为了更加直观地反映 IWOAHC 算法跳出局部最优的能力和收敛速度,各算法收敛曲线对比如图 1 所示。

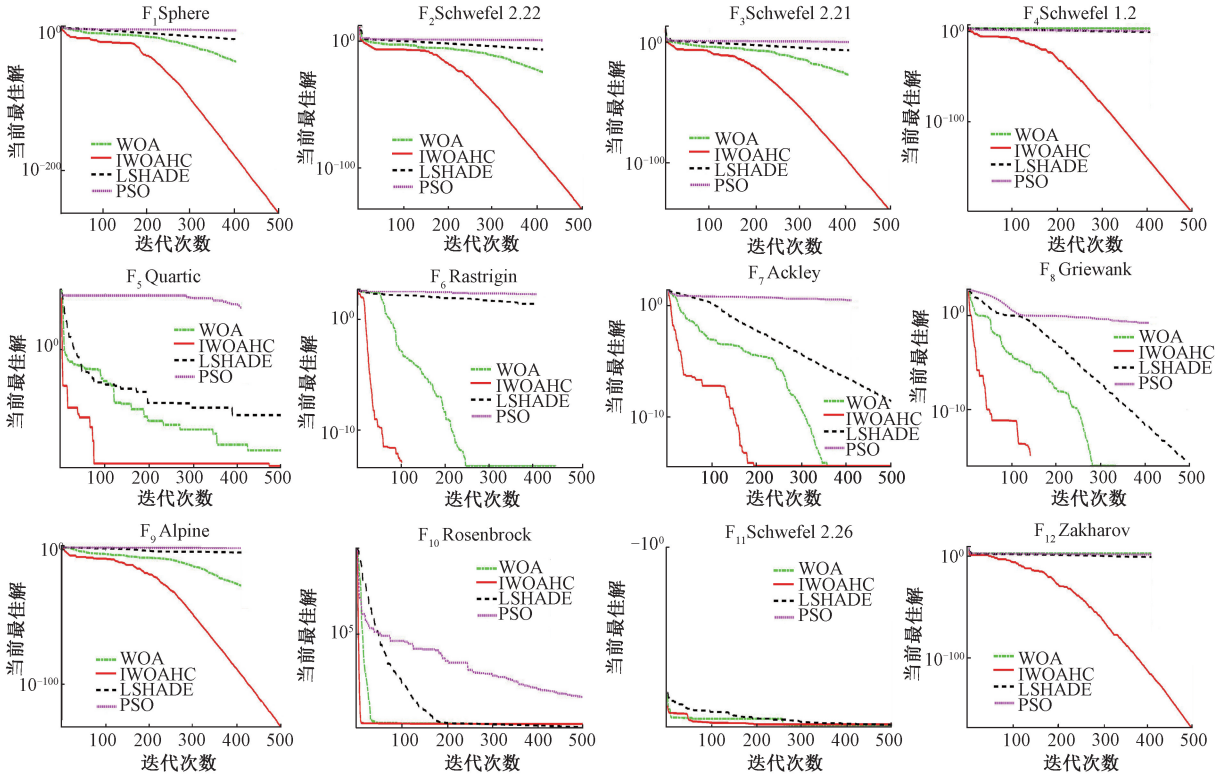

图 1　算法收敛曲线图

Fig. 1　Convergence curves of the algorithms

由表 2 和图 1 可知,本文提出的 IWOAHC 算法在 11 个测试函数中寻优结果都是最好的,仅函数 F_{10} 的 mean 指标劣于 LSHADE。以最优解的平均值和标准差为评价标准可以发现,IWOAHC 算法的求解精度更高、收敛速度更快。尤其对于沿自变量方向会产生大量局部极值,具有较高寻优难度的多模态函数,IWOAHC 算法求解精度明显优于其他几种智能算法且鲁棒性强。这充分表明 IWOAHC 算法在平衡探索与开发的综合能力上远胜于其他算法,有良好的普适性。

3.2　不同改进策略对算法性能影响

为了比较两种改进策略对 IWOAHC 算法性能的影响,分别利用 IWOA、WOAHC 对表 1 的测试函数进行优化。测试结果如表 3 所示。

由表 3 分析可知,仅采用非线性控制收敛因子策略(IWOA)和仅采用“历史认知解”作为干扰因子的策略(WOAHC)对原始算法性能的改进有限,而两种策略的结合使算法性能有较大提升。因此,说明非线性收敛因子和“历史认知解”对 WOA 算法的优化能力有较大影响。然而,对于多数基准测试函数,仅采用其中一种改进策略对算法性能的改善与两种策略结合的 IWOAHC 算法存在较大差距。所以,IWOAHC 综合了两种改进策略的优势,对多数基准测试函数的优化精度明显优于单一

表 3　不同改进策略的算法性能对比

Table 3　Property comparison of different improved strategy

函数	指标	IWOA	WOAHC	IWOAHC
F_1	Mean	1.46×10^{-151}	0.00	0.00
	Std	7.89×10^{-151}	0.00	0.00
F_2	Mean	1.44×10^{-101}	2.59×10^{-278}	2.14×10^{-316}
	Std	7.90×10^{-101}	3.71×10^{-277}	6.14×10^{-315}
F_3	Mean	4.82×10^{1}	4.84×10^{-231}	3.45×10^{-271}
	Std	2.72×10^{1}	2.81×10^{-230}	7.12×10^{-270}
F_4	Mean	1.85×10^{4}	0.00	0.00
	Std	8.57×10^{3}	0.00	0.00
F_5	Mean	1.37×10^{-3}	4.35×10^{-4}	1.90×10^{-5}
	Std	1.68×10^{-3}	7.79×10^{-4}	6.66×10^{-4}
F_6	Mean	0.00	0.00	0.00
	Std	0.00	0.00	0.00
F_7	Mean	3.61×10^{-15}	3.14×10^{-15}	8.88×10^{-16}
	Std	2.22×10^{-15}	1.74×10^{-15}	1.64×10^{-15}
F_8	Mean	2.02×10^{-03}	0.00	0.00
	Std	1.10×10^{-2}	0.00	0.00
F_9	Mean	1.01×10^{-102}	8.05×10^{-280}	1.31×10^{-316}
	Std	5.31×10^{-102}	6.35×10^{-279}	4.72×10^{-315}
F_{10}	Mean	2.74×10^{1}	2.71×10^{1}	2.64×10^{1}
	Std	6.90×10^{-1}	4.97×10^{-1}	3.13×10^{-1}
F_{11}	Mean	-1.13×10^{4}	-1.12×10^{4}	-1.25×10^{4}
	Std	1.50×10^{3}	1.48×10^{3}	5.46×10^{1}
F_{12}	Mean	4.99×10^{2}	3.02×10^{-286}	0.00
	Std	1.23×10^{2}	8.37×10^{-285}	0.00

改进策略的 IWOA 和 WOAHC 算法。且在对基准测试函数取同样数量级的优化精度时，IWOAHC 的收敛速度更快，从而证明算法采用两种改进策略的有效性及合理性。

3.3　与其他改进 WOA 算法的优化性能对比

为全面评估 IWOAHC 算法的优化性能，引用文献[18-20]的相关实验数据，将本文算法与其他改进 WOA 算法的优化性能进行对比。设置种群规模为 30，迭代次数为 500，基准测试函数为 F_1～F_8，独立运行 30 次实验。以最优解的均值和标准差来评价，对比结果如表 4 所示（“–”为缺失数据）。

表 4　与其他改进算法性能对比

Table 4　Compare the performance of other improved algorithms

函数	指标	MSWOA[18]	CWOA[19]	DEOBWOA[20]	IWOAHC
F_1	Mean	0.00	0.00	3.61×10^{-231}	0.00
	Std	0.00	0.00	9.11×10^{-246}	0.00
F_2	Mean	7.17×10^{-203}	4.56×10^{-226}	3.32×10^{-122}	2.14×10^{-316}
	Std	1.64×10^{-203}	3.60×10^{-225}	5.59×10^{-127}	6.14×10^{-315}
F_3	Mean	4.59×10^{-139}	3.60×10^{-265}	2.10×10^{-107}	3.45×10^{-271}
	Std	2.51×10^{-138}	3.60×10^{-263}	7.10×10^{-112}	7.12×10^{-270}
F_4	Mean	0.00	—	2.68×10^{-182}	0.00
	Std	0.00	—	6.35×10^{-215}	0.00
F_5	Mean	—	3.61×10^{-5}	3.23×10^{-4}	1.90×10^{-5}
	Std	—	1.00×10^{-3}	5.83×10^{-3}	6.66×10^{-4}
F_6	Mean	0.00	0.00	—	0.00
	Std	0.00	0.00	—	0.00
F_7	Mean	0.00	8.88×10^{-16}	—	8.88×10^{-16}
	Std	0.00	4.01×10^{-31}	—	1.64×10^{-15}
F_8	Mean	0.00	0.00	0.00	0.00
	Std	0.00	0.00	0.00	0.00

由表 4 可知，除了函数 F_7，当 MSWOA、CWOA 和 DEOBWOA 对基准测试函数的优化精度达到理论最优时，IWOAHC 算法同样能够取到最优解。此外，IWOAHC 算法仅对函数 F_7 的优化指标劣于 MSWOA，其余测试函数优化求解精度均优于或等同于其他 3 种改进算法。IWOAHC 算法对函数 F_1、F_4、F_6、F_8 最优解的均值和标准差均取到 0，达到其理论值；对函数 F_5 的优化效果优于其他算法，对函数 F_2、F_3 求解的最优解均值高出其他算法多个量级。

综上所述，本文提出的 IWOAHC 算法不仅相对于经典智能算法在收敛速度和优化精度方面有显著的提升，而且对比目前较新的改进 WOA 算法，IWOAHC 仍具有明显的优势。

3.4　参数 ω 敏感性分析

分析 IWOAHC 算法机理可知，由式(18)描述的调节因子 ω 的取值是本文中的一个关键参数，通过选取不同的 ω 值，即 $\omega=0.3$、$\omega=0.4$、$\omega=0.5$、$\omega=0.6$、$\omega=0.7$ 进行实验，分析其对算法性能的影响。表 5 为不同 ω 取值时 IWOAHC 算法获得的平均精度值比较，此外，除了参数 ω，其他参数均相同。

分析表 5 结果可知，当调节因子 ω 选取 5 个不同的值时，IWOAHC 算法的全局优化性能变化不明显。表明该算法对调节因子 ω 不是特别敏感。总体来看，参数 $\omega=0.5$ 是较合理的取值。

3.5　基于 IWOAHC 算法的能耗管理实验

式(7)中常数 $d(d>0)$，表明故障率对 DVS 的影响度，系统的参数设置如第 1 节所述，其中处理器频率 f_i 为

表 5　不同 ω 值对函数的平均精度的影响

Table 5　Influence of different ω values on the average accuracy of the functions

ω	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}
0.3	0.00	3.83×10^{-282}	1.72×10^{-233}	0.00	8.33×10^{-3}	0.00	3.38×10^{-15}	0.00	2.23×10^{-268}	2.76×10^{1}	-1.03×10^{4}	0.00
0.4	0.00	1.82×10^{-297}	1.48×10^{-258}	0.00	8.52×10^{-4}	0.00	3.97×10^{-15}	0.00	1.67×10^{-286}	2.84×10^{1}	-1.08×10^{4}	0.00
0.5	0.00	2.14×10^{-316}	3.45×10^{-271}	0.00	1.90×10^{-5}	0.00	8.88×10^{-16}	0.00	1.31×10^{-316}	2.64×10^{1}	-1.25×10^{4}	0.00
0.6	0.00	2.0332×10^{-301}	1.20×10^{-257}	0.00	7.63×10^{-4}	0.00	3.26×10^{-15}	0.00	6.3929×10^{-298}	2.76×10^{1}	-1.12×10^{4}	0.00
0.7	0.00	1.403×10^{-287}	7.42×10^{-251}	0.00	7.18×10^{-3}	0.00	3.38×10^{-15}	0.00	1.0593×10^{-277}	2.78×10^{1}	-1.02×10^{4}	0.00

(0.1,1)内的连续分布(即 f_{max} 归一化为 1)，同时根据式(6)和式(7)，假设 f_{max} 可以满足可靠性和实时性约束。本文对 d 在 1～6 取整进行实验，不同常数 d 和频率 f 下故障率变化情况如图 2 所示。可见故障率随频率增加呈线性下降，而且 d 越大，最大故障率越大。当 $d=1$ 时，最大的故障率为 10^{-5}，故式(7)中 $\lambda_k=10^{-5}$ 为故障率约束。

本文采用 3 组任务量(分别为 10、30 和 50)来模拟实际的多任务处理。任务的 C_i(WCET)均匀分布在(20,50)之间。任务截止时间 $D_i\in(20,220)$ 并随机生成。为了避免随机性，每个实验执行 20 次。表 6 为本文的实验

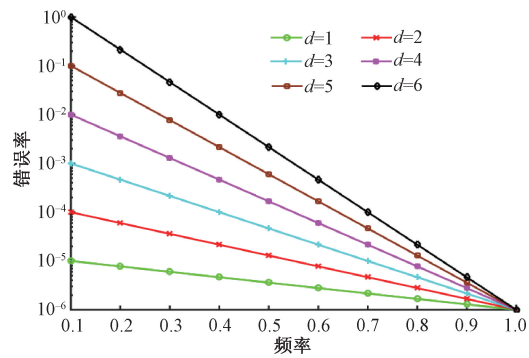

图 2　频率故障率的关系

Fig. 2　Relationship between frequency and failure rates

参数设置。表 7 是任务量为 10、30 和 50 的实验结果，优化效果通过最佳值、最差值、平均值和标准差反映。其中 *NPM_Val* 为无电源管理（no power management，NPM）的理想结果，由式(5)和(6)可知，不同任务规模下的 *NPM_Val* 可由如下公式得到：

$$NPM\_Val_{\min} = t_s \cdot 37.556\ 6$$
$$NPM\_Val_{\max} = t_s \cdot 342.705\ 1 \tag{22}$$

式中：t_s 为任务标度（10、30 和 50），$NPM\_Val_{\min}$ 和 $NPM\_Val_{\max}$ 分别表示理想最小能耗和最大能耗，根据式(5)当 $f_i = f_{\min} = 0.1$，$C_i = C_{\min} = 20$ 单个任务的理想最小能耗为 37.556 6；当 $f_i = f_{\max} = 1$，$C_i = C_{\max} = 50$ 时，单个任务最大能耗为 342.705 1。理论上 $NPM\_Val_{\min}$ 是不可达到的。

表 6　实验参数列表

Table 6　Experimental parameter List

参数	f_i	t_s	实验次数	C_i(WECT)	D_i	T_{max}
取值	(0.1,1)	10,30,50	20	20~50	20~220	500

表 7　不同任务量的优化结果比较

Table 7　Comparison of optimization results of different task volumes

算法	任务量 10				任务量 30				任务量 50			
	$NPM\_Val_{\min}$= 375.47		$NPM\_Val_{\max}$=3 427.05		$NPM\_Val_{\min}$=1126.40		$NPM\_Val_{\max}$=10 281.15		$NPM\_Val_{\min}$=1 877.33		$NPM\_Val_{\max}$=17 135.25	
	最优解	最差解	均值	标准差	最优解	最差解	均值	标准差	最优解	最差解	均值	标准差
IWOAHC	**726.40**	**920.11**	**885.76**	**18.47**	**3 529.43**	**4 718.99**	**4 076.50**	**91.73**	**7 620.70**	**9 301.00**	**8 572.97**	**165.79**
WOA	815.03	1 114.79	961.82	74.12	3 975.70	4 996.86	4 349.25	286.21	7 625.72	9 976.12	8 816.97	574.52
LSHADE	867.99	1 032.51	962.60	44.66	4 305.89	4 749.05	4 477.56	110.58	7 607.57	16 680.17	11 896.50	8 358.99
PSO	无效	无效	无效	36.11	无效	无效	无效	105.14	无效	无效	无效	3 169.42
文献[6]	821.52	1 040.01	913.04	57.66	3 642.20	4 936.64	4 368.30	345.31	8 281.54	10 023.18	9 319.57	535.50

由表 7 可知，提出的 IWOAHC 算法在实时性和稳定性约束下，求解结果明显更加接近理论值 NPM_Val_{min}。特别是随着任务规模的增加，IWOAHC 的优势更加明显。表中的无效值表示解决方案超出了频率范围，然而，无论任务规模如何，IWOAHC 仍然保持着强大的优化能力。IWOAHC 算法的均值在任务量为 10、30 和 50 时比传统的 WOA 算法分别减少了 7.9%、6.3%和 2.8%的能耗，本质上具有更好的局部最优规避能力和平衡全局搜索和局部开发的能力，能够以较低的计算成本获得可接受的解决方案

此外，为了更加直观地反映本文提出的 IWOAHC 算法优化能耗管理问题的有效性，将该算法与原始 WOA 算法的收敛曲线进行了对比，对比结果如图 3 所示。3 幅图的优化任务量分别为 10，30 和 50。分析图 3 可以发现，IWOAHC 算法的收敛速度更快、优化的目标函数值更小，即在可靠性和实时性约束下系统功耗最低，能快速、有效地调节系统功耗。

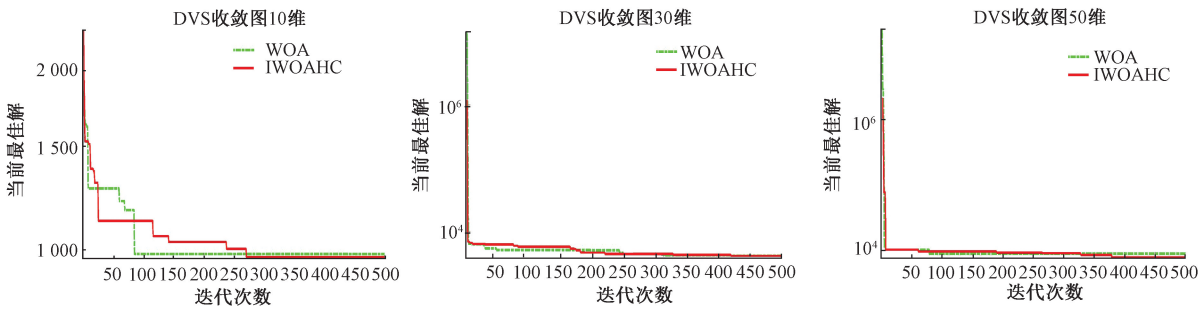

图 3　能耗管理收敛曲线

Fig. 3　Convergence curves of energy consumption management

4　结　论

本文针对嵌入式实时系统能耗管理中 DVS 技术的应用引发系统故障率增加的问题，提出了一种 IWOAHC 算法的动态能耗优化管理方案。研究分析了 WOA 算法的缺点，提出了非线性动态控制收敛因子策略，有效加快了算法的收敛速度。并采用历史认知解作为种群的空间

结构,在收缩包围机制中,通过混合引导策略来平衡算法的局部开发和全局搜索能力。与其他先进算法在基本测试函数和动态能耗管理模型上进行实验比较,通过实验数据和进化曲线证明,改进的 IWOAHC 算法在测试函数和实时系统的能耗优化管理中都具有更快的收敛速度和更高的优化精度。因此,本文提出的算法具有良好的普适性,且适用于能耗优化管理。把本文的调度策略和算法以较小的开销和较好的扩展性应用到当前或未来的新兴复杂场景中将会成为下一步的研究工作内容。

参考文献

[1] 沈长青, 雷飘, 冯毅雄, 等. 基于自适应流形嵌入动态分布对齐的轴承故障诊断[J]. 电子测量与仪器学报, 2021, 35(2): 33-40.
SHEN CH Q, LEI P, FENG Y X, et al. Bearing fault diagnosis based on adaptive manifold embedding dynamic distribution alignment [J]. Journal of Electronic Measurement and Instrumentation, 2021, 35(2): 33-40.

[2] SALEHI M E, SAMADI M, NAJIBI M, et al. Dynamic voltage and frequency scheduling for embedded processors considering power/performance trade offset [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2011, 19(10): 1931-1935.

[3] TERZOPOULOS G, KARATZA H. Performance evaluation and energy consumption of a real-time heterogeneous grid system using DVS and DPM [J]. Simulation Modelling Practice and Theory, 2013, 36: 33-43.

[4] ZHANG X, HUANG K, YU M, et al. BFCO: A BPSO-based fine-grained communication optimization method for MPSoC[J]. IEEE Access, 2018, 6: 18771-18785.

[5] XUE L, WANG Y, CHANG N, et al. Concurrent task scheduling and dynamic voltage and frequency scaling in a real-time embedded system with energy harvesting[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2016, 35(11): 1890-1902.

[6] 罗钧, 刘泽伟, 张平, 等. 基于非线性因子的改进鸟群算法在动态能耗管理中的应用[J]. 电子与信息学报, 2020, 42(3): 729-736.
LUO J, LIU Z W, ZHANG P, et al. Application of improved bird swarm algorithm based on nonlinear factor in dynamic energy management [J]. Journal of Electronics and Information, 2020, 42(3): 729-736.

[7] MIRJALILI S, LEWIS A. The whale optimization algorithm[J]. Advances in Engineering Software, 2016, 95: 51-67.

[8] 刘浩然, 孙美婷, 李雷, 等. 基于蚁群节点寻优的贝叶斯网络结构算法研究[J]. 仪器仪表学报, 2017, 38(1): 143-150.
LIU H R, SUN M T, LI L, et al. Research on structural algorithm of bayesian network based on ant colony node optimization [J]. Chinese Journal of Scientific Instrument, 2017, 38(1): 143-150.

[9] KAVEH A, FARHOUDI N. A new optimization method: Dolphin echolocation [J]. Advances in Engineering Software, 2013, 59(5): 53-70.

[10] KENNEDY J, EBERHART R. Particle swarm optimization[C]. Proceedings of ICNN'95 IEEE International Conference on Neural Networks, 2011: 1942-1948.

[11] PRAKASH D B, LAKSHMI N C. Optimal siting of capacitors in radial distribution network using whale optimization algorithm [J]. Alexandria Engineering Journal, 2017, 56(4): 499-509.

[12] ZHAO H, GUO S, ZHAO H. Energy-related CO_2 emissions forecasting using an improved LSSVM model optimized by whale optimization algorithm[J]. Energies, 2017 10(7): 874-888.

[13] ABID A, HAQ Z U, KHAN M T. Fault detection using negative selection and genetic algorithms [J]. Instrumentation, 2019, 6(3): 39-51.

[14] DISSANAYAKE H, PERERA S. Design of a tree ring structure analysis system to estimate the accurate age of tree species in Sri Lanka [J]. Instrumentation, 2020, 7(3): 50-59.

[15] KIRAN M S. Particle swarm optimization with a new update mechanism[J]. Applied Soft Computing, 2017, 60: 670-678.

[16] ERTAS A H. Optimization of fiber-reinforced laminates for a maximum fatigue life by using the particle swarm optimization [J]. Part II. Mechanics of Composite Materials, 2013, 49: 107-116.

[17] TANABE R, FUKUNAGA A S. Improving the search performance of SHADE using linear population size reduction[C]. Evolutionary Computation, IEEE, 2014, doi: 10.1109/CEC.2014.6900380.

[18] 徐航, 张达敏, 王依柔, 等. 混合策略改进鲸鱼优化算法[J]. 计算机工程与设计, 2020, 41(12): 3397-3404.
XU H, ZHANG D M, WANG Y R, et al. The whale optimization algorithm for mixed strategy to improve[J]. Computer Application Research, 2020, 41(12): 3397-3404.

[19] 王坚浩, 张亮, 史超, 等. 基于混沌搜索策略的鲸鱼优化算法[J]. 控制与决策, 2019, 34(9):

1893-1900.
WANG J H, ZHANG L, SHI CH, et al. Whale optimization algorithm based on chaotic search strategy[J]. Journal of Control and Decision-Making, 2019, 34 (9): 1893-1900.
[20] 许瑜飞, 钱锋, 杨明磊, 等. 改进鲸鱼优化算法及其在渣油加氢参数优化的应用[J]. 化工学报, 2018, 69(3): 891-899.
XU Y F, QIAN F, YANG M L, et al. Improved whale optimization algorithm and its application in residual oil hydrogenation parameter optimization [J]. Chinese Journal of Chemical Industry, 2018, 69(3): 891-899.

作者简介

罗钧,1983 年于重庆大学获得学士学位,1993 年于重庆大学获得硕士学位,现为重庆大学教授,博导,主要研究方向为人工智能、精密机械及测试计量、智能信息处理。
E-mail:luojun@ cqu. edu. cn

Luo Jun received his B. Sc. degree from Chongqing University in 1983, M. Sc. degree from Chongqing University in 1993. Now he is a professor at Chongqing University. His main research interests include artificial intelligence, precision machinery and measurement, intelligent information processing.

庞亚男,2018 年于重庆大学获得学士学位,现为重庆大学硕士研究生,主要研究方向为嵌入式系统。
E-mail:473187923@ qq. com

Pang Yanan received her B. Sc. degree from Chongqing University in 2018. Now she is a M. Sc. candidate at Chongqing University. Her main research interest includes embedded system.

刘建强,2018 年于重庆大学获得学士学位,现为重庆大学硕士研究生,主要研究方向为图像处理。
E-mail:1172707725@ qq. com

Liu Jianqiang received his B. Sc. degree from Chongqing University in 2018. Now he is a M. Sc. candidate at Chongqing University. His main research interest includes image processing.