文章编号:1671 - 6833(2018)06 - 0014 - 09

改进鲸鱼群算法及其在炼钢连铸调度中的应用

曾 冰,王梦雨,高 亮,董昊臻

(华中科技大学 机械科学与工程学院,湖北 武汉 430074)

摘 要:研究了一种新的群体智能优化算法——鲸鱼群算法(whale swarm algorithm,WSA).系统介绍了鲸鱼群算法的原理、基本步骤及与其他典型群体智能优化算法相比的特点,并针对多峰优化问题的特点改进了 WSA 的迭代规则,引入稳定性阈值和适应度阈值两个参数,提出带迭代计数器的 WSA(WSA with iterative counter,WSA-IC).实验证明,WSA-IC 在最优解数量、最优解质量和收敛速度方面均有着优秀的表现,将 WSA-IC 应用于炼钢连铸调度问题,通过实验验证 WSA-IC 具有良好的寻优能力和稳定性,并提出从理论研究和实际应用两方面深化鲸鱼群算法的研究.

关键词:群体智能优化算法;鲸鱼群算法;多峰优化;炼钢连铸调度

中图分类号:TP183 **文献标志码**:A **doi**:10.13705/j.issn.1671 - 6833.2018.06.001

0 引言

当前,群体智能优化算法在工程优化问题中的应用日益广泛,特别是针对 NP-hard 问题,比如车间调度问题[1-2]、WSNs 分簇问题[3-4]以及多处理器调度问题[5]等.这些工程优化问题通常是多峰优化问题,即它们的目标函数通常会有多个全局最优解.如果运用逐点式的传统数值优化方法来求解这些问题,需要重复计算多次,再从这些解中选出最优的解,但并不能保证该解是全局最优解[6].群体智能优化算法通过个体之间的合作与竞争来引导种群的迭代,从而向更优的解收敛;并且它们的迭代规则不依赖于函数的导数或梯度等信息,易于使用.因此,越来越多的群体智能优化算法被人们用来解决实际工程优化问题.

受鲸鱼群利用超声波通信进行捕食等行为的启发,华中科技大学的曾冰和高亮于 2017 年提出了一种新型群体智能优化算法——鲸鱼群算法[7](whale swarm algorithm,WSA),在函数优化问题的求解中取得了显著效果.

1 鲸鱼群算法的原理

鲸鱼群算法是一种基于群体智能的元启发式算法.类似于粒子群算法,鲸鱼群算法首先从定义域内随机生成一组初始解,即随机初始化种群各个体的位置,然后根据种群中各个体的位置和适应度值,以某种机制产生新一代种群.但不同于粒子群算法通过跟踪个体历史最优解和全局最优解生成新种群的机制,鲸鱼群算法模拟鲸鱼群以超声波为信息媒介进行捕食等群体行为,将每个解类比为一条鲸鱼,每条鲸鱼的移动由比它好(由适应度值判断)的鲸鱼中离它最近的鲸鱼引导,笔者将这种引导鲸鱼定义为“较优且最近”的鲸鱼.

2 鲸鱼群算法的基本步骤

在鲸鱼群算法中,每条鲸鱼表示解空间中的一个候选解,解的优劣根据由优化目标定义的适应度函数判断.WSA 先对鲸鱼群各个体的位置进行随机初始化,并在之后的迭代过程中,对种群中的每条鲸鱼 $\boldsymbol{X}$,判断其“较优且最近”的鲸鱼 $\boldsymbol{Y}$ 是否存在,若存在,则鲸鱼 $\boldsymbol{X}$ 在鲸鱼 $\boldsymbol{Y}$ 的引导下进行随机移动,如式(1)所示;否则,鲸鱼 $\boldsymbol{X}$ 的位置保持不变.WSA 的基本流程如图 1 所示.

$$x_i^{t+1} = x_i^t + \mathrm{rand}(0,\rho_0 \cdot \mathrm{e}^{-\eta \cdot d_{X,Y}}) \cdot (y_i^t - x_i^t), \quad (1)$$

式中:x_i^t 和 x_i^{t+1} 分别为 $\boldsymbol{X}$ 的第 i 个元素在第 t 步与 $t+1$ 步迭代时的位置;y_i^t 为 $\boldsymbol{Y}$ 的第 i 个元素在第 t

收稿日期:2018 - 05 - 03;**修订日期**:2018 - 07 - 02

基金项目:国家自然科学基金资助项目(51575212);华中科技大学学术前沿青年团队项目

通信作者:高亮(1974—),男,山东临清人,华中科技大学教授,博士,博士生导师,主要从事智能优化算法及其在设计与制造中的应用研究,E-mail:gaoliang@mail.hust.edu.cn.

步迭代时的位置;$d_{X,Y}$ 为 $\boldsymbol{X}$ 与 $\boldsymbol{Y}$ 之间的距离;ρ_0 为超声波源的强度,通常设置为 2;η 为衰减系数,需根据目标函数的特点设置,一般先由公式 $\eta = -20 \cdot \ln 0.25/d_{\max}$ 计算其初始近似值,在此基础上调整为最优值或近似最优值;$\mathrm{rand}(0, \rho_0 \cdot \mathrm{e}^{-\eta \cdot d_{X,Y}})$ 为 0 到 $\rho_0 \cdot \mathrm{e}^{-\eta \cdot d_{X,Y}}$ 的随机数.

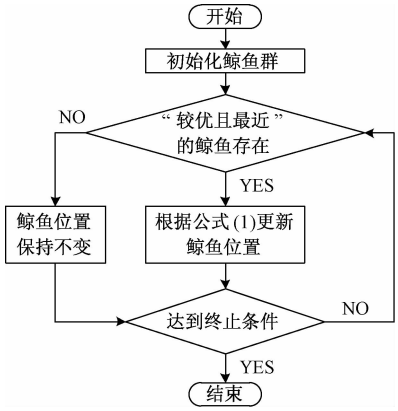

图 1　鲸鱼群算法的流程图

Fig. 1　Flow chart of whale swarm algorithm

根据式(1)可知,如果一条鲸鱼与它的“较优且最近”的鲸鱼之间的距离很小,该条鲸鱼将会积极地朝其“较优且最近”的鲸鱼随机移动;否则,它将消极地朝其“较优且最近”的鲸鱼随机移动,如图 2 所示. 图 2 中的目标函数维数为 2,六角星表示全局最优解,圆圈表示鲸鱼,用虚线标记的矩形区域是当前迭代中鲸鱼的可达区域.

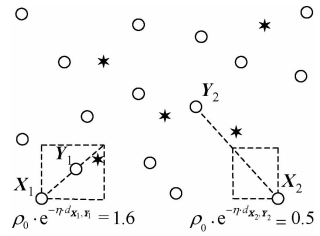

图 2　由“较优且最近”的鲸鱼引导的随机移动

Fig. 2　Random movement under the guidance of the“better and nearest” whale

3　鲸鱼群算法与典型群体智能优化算法的比较

群体智能优化算法主要通过模拟生物的群集行为,利用种群的群体智慧进行协同搜索. 这类算法对目标函数的性态(单调性、可导性、模态性等)几乎没有限制,使用简单高效,可广泛地应用于各类优化问题. 典型的群体智能优化算法有粒子群优化(particle swarm optimization, PSO)算法、蚁群优化(ant colony optimization, ACO)算法等.

受鸟群觅食行为的启发,PSO 采用速度 - 位置模型,通过动态跟踪个体历史最优解和全局最优解在解空间内进行搜索. PSO 主要应用于连续优化问题,所需设置的参数较少,收敛速度较快,但无法直接用于求解离散优化问题. ACO 模拟蚂蚁的觅食行为,通过动态更新各路径上的信息素,使蚂蚁在信息素的引导下找到最短路径. 相较于 PSO,ACO 适用于求解离散优化问题(如最短路径问题),但需要调整较多的参数. 而 WSA 主要针对连续优化问题设计,由于其框架易于扩充的特点,可通过简单的改进应用于离散优化问题. 将候选解视为鲸鱼,采用向“较优且最近”的鲸鱼随机移动的位置更新模型,在保持种群多样性的同时进行有效的局部搜索,有利于求解多个全局最优解. 此外 WSA 控制参数较少,易于使用.

WSA 与近几年提出的头脑风暴优化(brain storm optimization, BSO)算法[8-9]存在诸多相似之处:两者均充分利用种群个体的位置分布信息引导最优解的搜索. 其主要区别在于引导方式,WSA 主要通过个体向“较优且最近”的鲸鱼移动促使算法收敛至最优解;BSO 则基于问题特征和解数据的分析,采用数据挖掘技术引导算法搜索最优解.

4　鲸鱼群算法的改进

4.1　WSA 迭代规则的改进

为解决 WSA 中参数 η 设置的难题,假设超声波的强度在水中不会衰减,即 $\eta = 0$,则 $\rho_0 \cdot \mathrm{e}^{-\eta \cdot d_{X,Y}} = 2$. 当鲸鱼 $\boldsymbol{X}$ 在“较优且最近”的鲸鱼 $\boldsymbol{Y}$ 的引导下随机移动时,若 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 分别位于不同的全局最优解附近,如图 3 所示,$\boldsymbol{X}$ 很可能离开附近最优解所在区域,降低了算法的局部搜索能力. 因此,对 WSA 中鲸鱼的位置更新规则进行如下改进:为当前迭代的鲸鱼(即正本)生成一个副本,若副本在其“较优且最近”的鲸鱼引导下随机移动到优于正本的位置,则正本移至副本所在位置;否则正本保持原地不动. 这样图 3 中的鲸鱼 $\boldsymbol{X}$ 会以很大的概率保持原地不动,从而引导其他个体向其追随的最优解收敛. 可见,改进后的位置更新规则仍然适用于 $\eta = 0$ 的情况,可以在不引入任何小生境参数的前提下,有效保证多个子种群的形成,提高算法的局部搜索能力,这非常有利于求解多个全局最优解.

4.2　在迭代过程中识别并跳出已找到的极值点

为使 WSA 在迭代过程中有效识别并跳出已找到的极值点,提高算法的全局搜索能力,引入

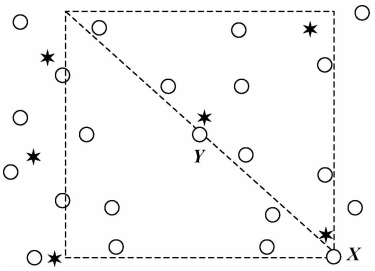

图 3 当 $\eta=0$ 时,由"较优且最近"的鲸鱼引导的随机移动

Fig. 3 Random movement under the guidance of the "better and nearest" whale when $\eta=0$

稳定性阈值 T_s 和适应度阈值 T_f 两个参数. 稳定性阈值 T_s 是一个预先定义的迭代次数,用来判断一条鲸鱼是否已经找到所跟随的极值点,达到稳定状态. 为实现上述判断,每条鲸鱼设置一个迭代计数器 c,以记录该鲸鱼未更新位置的迭代次数. 适应度阈值 T_f 用于识别当前全局最优解. 在每次迭代过程中,对于种群中未更新的鲸鱼 $\boldsymbol{X}$,需检查其迭代计数器:若 $\boldsymbol{X}$ 未达到稳定状态(即 $\boldsymbol{X}.c \neq T_s$),则其迭代计数器自增 1;反之,根据 T_f 和 $|fgbest - f(\boldsymbol{X})|$ 的大小($fgbest$ 为当前全局最优适应度;$f(\boldsymbol{X})$ 为 $\boldsymbol{X}$ 的适应度),判断是否更新当前全局最优解集合,接着初始化该鲸鱼,从而跳出已经找到的极值点. 基于上述对 WSA 的改进方案,改进的 WSA 被称为带迭代计数器的 WSA (WSA with iterative counter, WSA-IC).

4.3 WSA-IC 算法的流程

初始化鲸鱼包括初始化鲸鱼的位置以及将鲸鱼的迭代计数器置 0.

WSA-IC 算法伪代码如下.

输入:目标函数,鲸鱼群 $\boldsymbol{\Omega}$.

输出:当前全局最优解集 ***GloOpt***.

```
1: 开始
2: 初始化参数;
3: 初始化鲸鱼;
4: 计算鲸鱼的适应度值;
5: while 终止条件不满足 do
6:     for i = 1 to |Ω| do
7:         寻找 Ω_i 的"较优且最近"的鲸鱼 Y;
8:         if Y 存在 then
9:             生成 Ω_i 的副本 X';
10:            X'在 Y 的引导下根据公式(1)移动;
11:            计算 X'的适应度值 f(X');
12:            if f(X') < f(Ω_i) then
13:                Ω_i = X';
14:                Ω_i.c = 0;
15:            else
16:                检查 Ω_i 的迭代计数器;
17:            end if
18:        else
19:            检查 Ω_i 的迭代计数器;
20:        end if
21:    end for
22: end while
23: 判断 Ω 中的每条鲸鱼是否是当前全局最优解;
24: 返回 GloOpt;
25: 结束
```

4.4 WSA-IC 算法复杂度及收敛性分析

4.4.1 复杂度分析

WSA-IC 算法的时间复杂度分析见表 1,种群规模和目标函数维度分别为 m、n. 在复杂度最高情况下,即当鲸鱼在位置变换时没有发现适应度更优的个体且迭代计数器达到阈值 T_s,更新鲸鱼个体需执行 $4n+1$ 次加法、$2n$ 次乘法和 $2n+m+2$ 次比较. 因此,WSA-IC 算法的时间复杂度为 $O(n^2)$.

4.4.2 收敛性分析

由式(1)及 WSA-IC 算法伪代码可知,WSA-IC 算法可写成如下形式:

$$x_i^{t+1} = \begin{cases} Ax_i^t + By_i^t, & f(Ax_i^t + By_i^t) < f(x_i^t); \\ x_i^t, & f(Ax_i^t + By_i^t) \geqslant f(x_i^t). \end{cases} \tag{2}$$

式中:$A = 1 - \mathrm{rand}(0,2)$,$B = \mathrm{rand}(0,2)$.

因此可通过计算求得 $\mathrm{E}(A)=0$,$\mathrm{E}(B)=1$,$\mathrm{D}(A)=\mathrm{D}(B)=-\mathrm{E}(AB)=1/3$.

表 1 WSA-IC 算法复杂度分析

Tab. 1 Analysis of the algorithm complexity for WSA-IC

算法步骤		加法运算	乘法运算	比较	函数评价
(1)寻找"较优且最近"的鲸鱼		0	0	$m-1$	0
(2)"较优且最近"的鲸鱼存在	鲸鱼位置变换计算	$2n$	n	0	0
	边界修正	0	0	$2n$	0
	变换后鲸鱼位置评价	0	0	0	1
	与鲸鱼原始位置适应度的比较	0	0	1	0
(3)检查迭代计数器操作	未达到 T_s 值	1	0	0	0
	达到 T_s 值	$2n+1$	n	2	1

要证明 WSA-IC 算法收敛,只需证明式(2)收敛,即证明 x_i^{t+1} 的期望和方差收敛. 证明如下:

$$\mathrm{E}(x_i^{t+1}) = \mathrm{E}(Ax_i^t + By_i^t), \tag{3}$$

$$\mathrm{E}(x_i^{t+1})^2 = \mathrm{E}(Ax_i^t + By_i^t)^2. \tag{4}$$

由于 A、B 与 y_i^t 相互独立,y_i^t 可以看成是确定的,式(3)、(4)可作如下变形:

$$\mathrm{E}(x_i^{t+1}) = \mathrm{E}(A)\mathrm{E}(x_i^t) + \mathrm{E}(B)y_i^t, \tag{5}$$

$$\frac{1}{\mathrm{E}(B)}\mathrm{E}(x_i^{t+1}) - \frac{\mathrm{E}(A)}{\mathrm{E}(B)}\mathrm{E}(x_i^t) = y_i^t, \tag{6}$$

$$\mathrm{E}(x_i^{t+1})^2 = \mathrm{E}(A^2)\mathrm{E}(x_i^t)^2 + 2\mathrm{E}(AB)\mathrm{E}(x_i^t)y_i^t + \mathrm{E}(B^2)(y_i^t)^2. \tag{7}$$

由 $\frac{1}{\mathrm{E}(B)}\lambda - \frac{\mathrm{E}(A)}{\mathrm{E}(B)} = 0$ 得 $\mathrm{E}(x_i^t)$ 的收敛特征值 $\lambda = \mathrm{E}(A) = 0 < 1$,因此 $\mathrm{E}(x_i^t)$ 收敛.

由式(5)、(7)推出

$$\begin{aligned}\mathrm{D}(x_i^{t+1}) &= \mathrm{E}(x_i^{t+1})^2 - \mathrm{E}^2(x_i^{t+1}) = \\ &\mathrm{E}(A^2)\mathrm{E}(x_i^t)^2 - \mathrm{E}^2(A)\mathrm{E}^2(x_i^t) + \\ &2\mathrm{E}(AB)\mathrm{E}(x_i^t)y_i^t - 2\mathrm{E}(A)\mathrm{E}(B)\mathrm{E}(x_i^t)y_i^t + \\ &(\mathrm{E}(B^2) - \mathrm{E}^2(B))(y_i^t)^2.\end{aligned} \tag{8}$$

$$\begin{aligned}\mathrm{D}(x_i^{t+1}) - \mathrm{E}(A^2)\mathrm{D}(x_i^t) &= \\ \mathrm{D}(A)\mathrm{E}^2(x_i^t) + 2\mathrm{E}(AB)\mathrm{E}(x_i^t)y_i^t - &\\ 2\mathrm{E}(A)\mathrm{E}(B)\mathrm{E}(x_i^t)y_i^t + \mathrm{D}(B)(y_i^t)^2 &= \\ \mathrm{D}(A)(\mathrm{E}^2(x_i^t) - 2\mathrm{E}(x_i^t)y_i^t + (y_i^t)^2).&\end{aligned} \tag{9}$$

同理可证 $\mathrm{D}(x_i^t)$ 收敛. 因此,在 WSA-IC 算法中,鲸鱼总是收敛于较优且最近的个体.

4.5　WSA-IC 算法的参数设置

WSA-IC 算法包含 4 个参数,即超声波源强度 ρ_0、衰减系数 η、稳定性阈值 T_s 以及适应度阈值 T_f. 其中,ρ_0 和 η 分别被设置为 2 和 0;T_f 通常认为与给定的适应度误差(即精度)相等,当精度未给定时,可设置为 1.0×10^{-8};基于大量实验结果,T_s 的值一般设置为函数维度的 100 倍.

4.6　数值实验

采用如下 6 个多峰优化算法作为对比算法:locally informed PSO(LIPS)[6]、fitness-euclidean distance ratio PSO(FERPSO)[10]、speciation-based DE(SDE)[11]、crowding DE(CDE)[12]、speciation-based PSO(SPSO)[13]以及 WSA. 所有算法都是在 Microsoft Visual Studio 2015 中用 C++编程语言实现,在相同的环境下运行,停止条件均为 CPU 计算时间.

4.6.1　测试函数

测试函数采用 15 个 CEC2015 多峰基准测试函数,其基本信息如表 2 所示. 表中 n、o_1、o_2 分别表示函数维度、全局最优解个数、局部最优解个数. 所有的测试函数都是最小化问题,其搜索空间均为 $[-100, 100]^n$.

表 2　CEC2015 多峰测试函数

Tab.2　CEC2015 multimodal functions for the experiment

函数序号	函数名称	n	o_1	o_2
F1	Expanded Two-Peak Trap	5	1	15
F2	Expanded Five-Uneven-Peak Trap	5	32	0
F3	Expanded Equal Minima	4	625	0
F4	Expanded Decreasing Minima	5	1	15
F5	Expanded Uneven Minima	3	125	0
F6	Expanded Himmelblau's Function	4	16	0
F7	Expanded Six-Hump Camel Back	6	8	0
F8	Modified Vincent Function	3	216	0
F9	Composition Function 1	10	10	0
F10	Composition Function 2	10	1	9
F11	Composition Function 3	10	10	0
F12	Composition Function 4	10	10	0
F13	Composition Function 5	10	10	0
F14	Composition Function 6	10	1	19
F15	Composition Function 7	10	1	19

4.6.2　参数设置

算法的主要参数设置如表 3、表 4 所示. 在表 3 中,m_1/m_2 表示 WSA-IC 算法的种群大小/其他算法的种群大小,CPU 计算时间以 s 为单位. 由于 WSA-IC 算法在迭代过程中可以有效识别并跳出已经找到的极值点,因此可以设置相对较小的种群规模,以降低算法的计算复杂度.

表 3　CEC2015 多峰测试函数相关参数的设置

Tab.3　Setting of parameters associated with CEC2015 multimodal functions for the experiment

函数序号	适应度误差	m_1/m_2	CPU/s
F1	0E-08	40/200	6
F2	0E-08	60/200	200
F3	0E-08	50/2 000	1 500
F4	0E-08	30/100	180
F5	0E-08	40/800	80
F6	0E-08	40/200	20
F7	0E-06	30/100	30
F8	0E-04	100/1 000	1 500
F9	0E-08	500/3 000	1 800
F10	0E-08	500/1 000	500
F11	0E-08	100/800	800
F12	0E-08	400/600	800
F13	0E-08	50/300	260
F14	0E-08	400/2 000	500
F15	0E-08	100/600	1 000

表 4 7 种算法主要参数的设置

Tab. 4 Setting of main parameters of seven algorithms

算法	参数
LIPS	$\omega = 0.729\ 844$, $nsize = 2 \sim 5$
FERPSO	$\chi = 0.729\ 844$, $\varphi_{\max} = 4.1$
SDE	$CR = 0.9$, $F = 0.5$, $m = 10$
CDE	$CR = 0.9$, $F = 0.5$, $CF = m$
SPSO	$\chi = 0.729\ 844$, $\varphi_1 = 2.05$, $\varphi_2 = 2.05$
WSA	$\rho_0 = 2$
WSA-IC	$\rho_0 = 2$, $\eta = 0$, $T_s = 100 \cdot n$, $T_f = \varepsilon_f$

注:ω 为惯性权重;$nsize$ 为邻居个数;χ 为收缩因子;$\varphi_{\max}$ 为系数;CR 为交叉概率;F 为缩减因子;n 为种群大小;CF 为拥挤因子;φ_1、φ_2 为系数.

4.6.3 评价指标

为保证比较的公正性,每个算法在每个测试函数上独立运行 51 次,并从最优解数量、最优解质量、收敛速度三方面来度量算法的性能.

4.6.4 实验结果与分析

(1)最优解的数量. 算法在最优解数量方面的表现通过平均最优解个数(average number of optima found,ANOF)衡量. 算法的 ANOF 值及显著性水平为 0.05 时的 Z 检验结果见表 5. 表中,avg ± sd 表示算法取得的 ANOF 均值 ± 标准差;δ 表示 Z 检验结果,且符号“ + ”(“ - ”)表示 WSA-IC 算法的结果优于(劣于)对比算法且差异显著,符号“ = ”表示 WSA-IC 算法和对比算法结果的差异不显著. 总体看来,符号“ - ”的数量只有 1 个,针对每种对比算法,符号“ + ”的数量占多数,意味着 WSA-IC 算法在最优解数量上的表现远好于其他算法.

表 5 7 种算法在测试函数上获得的 ANOF 值与 Z 检验结果

Tab. 5 ANOF of seven algorithms on test functions and results of Z-test

函数	WSA-IC	LIPS		FERPSO		SPSO		SDE		CDE		WSA	
序号	avg ± sd	avg ± sd	δ	avg ± sd	δ	avg ± sd	δ	avg ± sd	δ	avg ± sd	δ	avg ± sd	δ
F1	**1 ± 0**	0.22 ± 0.41	+	0.67 ± 0.47	+	0 ± 0	+	**1 ± 0**	=	0 ± 0	+	0.12 ± 0.32	+
F2	**32 ± 0**	21.02 ± 1.80	+	7.43 ± 1.42	+	0 ± 0	+	13.77 ± 1.55	+	0 ± 0	+	1.96 ± 0.74	+
F3	**625 ± 0**	257.98 ± 8.81	+	148.14 ± 7.32	+	0 ± 0	+	163.73 ± 212.52	+	1.94 ± 1.31	+	10.06 ± 1.42	+
F4	**1 ± 0**	0.33 ± 0.47	+	0.43 ± 0.50	+	0.33 ± 0.47	+	0.80 ± 0.40	+	**1 ± 0**	=	0 ± 0	+
F5	**125 ± 0**	82.82 ± 4.21	+	84.29 ± 4.36	+	0 ± 0	+	15.65 ± 1.54	+	0 ± 0	+	6.22 ± 0.89	+
F6	**16 ± 0**	14.16 ± 0.85	+	11.39 ± 1.36	+	0.20 ± 0.40	+	4.37 ± 0.48	+	**16 ± 0**	=	2.73 ± 0.79	+
F7	**8 ± 0**	5.24 ± 0.92	+	2.73 ± 1.07	+	0.24 ± 0.42	+	7.47 ± 0.70	+	**8 ± 0**	=	0.51 ± 0.50	+
F8	**216 ± 0**	86.28 ± 5.51	+	70.29 ± 4.23	+	11.31 ± 3.54	+	23.53 ± 1.67	+	204.59 ± 3.86	+	10.84 ± 1.41	+
F9	**4.22 ± 1.13**	1.04 ± 0.82	+	3.55 ± 1.05	+	0 ± 0	+	0 ± 0	+	0 ± 0	+	2.59 ± 0.69	+
F10	**0 ± 0**	**0 ± 0**	=	**0 ± 0**	=	**0 ± 0**	=	**0 ± 0**	=	**0 ± 0**	=	**0 ± 0**	=
F11	**0.80 ± 0.40**	0.16 ± 0.36	+	0.10 ± 0.30	+	0.35 ± 0.48	+	0.67 ± 0.46	=	0 ± 0	+	0.39 ± 0.49	+
F12	0.14 ± 0.34	0.20 ± 0.40	=	0.02 ± 0.14	=	0 ± 0	=	**0.82 ± 0.38**	-	0 ± 0	=	0 ± 0	=
F13	0.94 ± 0.24	0.77 ± 0.42	=	0.96 ± 0.19	=	0 ± 0	+	**1 ± 0**	=	**1 ± 0**	=	0.04 ± 0.19	+
F14	**0 ± 0**	**0 ± 0**	=	**0 ± 0**	=	**0 ± 0**	=	**0 ± 0**	=	**0 ± 0**	=	**0 ± 0**	=
F15	**1 ± 0**	**1 ± 0**	=	**1 ± 0**	=	0 ± 0	+	0.98 ± 0.14	=	0.12 ± 0.32	+	0.06 ± 0.24	+

(2)最优解的质量. 将算法取得的最优解适应度值减去 100 与相应函数序号的乘积,由此得到的最优解均值(标准差)及 Z 检验结果如表 6 所示,分别用 avg(sd)和 δ 表示. 显然,符号“ - ”的数量远少于符号“ + ”和“ = ”的数量,说明 WSA-IC 算法取得了较高的最优解精度.

(3)收敛速度. 在比较算法的求解效率时,以算法在测试函数 F14 上的收敛曲线为例进行分析,如图 4 所示. 横坐标为评价次数,纵坐标为算法多次实验找到的当前全局最优解适应度均值. 由于 FERPSO 和 WSA 的种群可能会过早收敛而停止迭代,对比算法不包括这两个算法. 从图 4 可以看出,相比于其他算法,WSA-IC 算法在 F14 上以更快的速度收敛到更优的解.

综上所述,WSA-IC 算法能高效地搜索到更多更优的解,这主要得益于对 WSA 两方面的改进:其一,改进迭代规则,使 WSA-IC 算法在保证多个子种群形成的同时保持局部搜索能力;其二,在迭代过程中识别并跳出已经找到的极值点,加速 WSA-IC 算法向全局最优解收敛,尽可能地提高算法的全局搜索能力.

表 6　7 种算法在测试函数上获得的最优解质量与 Z 检验结果

Tab. 6　Quality of optima found by seven algorithms on test functions and results of Z-test

函数序号	WSA-IC	LIPS		FERPSO		SPSO		SDE		CDE		WSA	
	avg(sd)	avg(sd)	δ	avg(sd)	δ	avg(sd)	δ	avg(sd)	δ	avg(sd)	δ	avg(sd)	δ
F1	**0.00E+00** (**0.00E+00**)	5.39E+00 (1.30E+01)	+	1.33E+01 (1.89E+01)	+	3.66E−02 (7.61E−02)	+	3.62E−10 (1.80E−09)	=	4.32E−01 (2.95E−01)	+	5.73E+01 (3.19E+01)	+
F2	**1.39E−16** (**9.85E−16**)	5.86E−12 (3.28E−11)	=	6.11E−14 (6.71E−14)	=	2.60E−01 (2.71E−01)	+	5.51E−10 (4.50E−10)	=	2.40E−04 (3.04E−04)	=	6.75E−01 (4.77E+00)	+
F3	3.25E−14 (1.83E−13)	1.12E−10 (4.36E−11)	=	2.41E−12 (9.77E−12)	=	3.57E−03 (1.88E−03)	=	2.52E−10 (5.00E−10)	=	1.10E−08 (3.29E−09)	=	**0.00E+00** (**0.00E+00**)	=
F4	**3.34E−15** (**1.75E−14**)	2.51E−02 (3.47E−02)	=	7.23E−02 (8.17E−02)	+	2.39E−01 (9.09E−01)	=	2.11E−02 (4.59E−02)	=	1.45E−14 (2.48E−14)	=	1.29E+00 (5.35E−01)	+
F5	2.12E−15 (9.89E−15)	1.35E−10 (8.79E−11)	=	2.58E−12 (1.51E−11)	=	5.49E−04 (3.72E−04)	=	3.26E−10 (3.66E−10)	=	8.58E−05 (5.74E−05)	=	**0.00E+00** (**0.00E+00**)	=
F6	**2.55E−14** (**5.29E−14**)	6.71E−12 (3.52E−11)	=	4.69E−14 (6.00E−14)	=	2.37E−02 (2.84E−02)	=	3.56E−10 (7.27E−10)	=	3.32E−11 (1.15E−10)	=	2.69E−11 (1.13E−10)	=
F7	**5.58E−07** (**3.34E−15**)	5.64E−07 (2.59E−08)	=	**5.58E−07** (**6.06E−14**)	=	1.22E−02 (1.61E−02)	=	6.15E−07 (6.42E−08)	=	**5.58E−07** (**1.29E−14**)	=	1.79E+00 (1.97E+00)	+
F8	**2.36E−08** (**6.89E−08**)	3.85E−06 (1.61E−06)	=	2.12E−07 (3.95E−07)	=	6.38E−05 (9.64E−06)	=	5.92E−06 (4.10E−06)	=	1.31E−06 (5.85E−07)	=	3.02E−06 (3.47E−06)	=
F9	**6.86E−14** (**5.99E−14**)	1.12E−07 (3.98E−07)	=	1.47E−10 (4.23E−10)	=	1.53E+00 (3.57E−09)	+	1.53E+00 (7.90E−14)	+	1.53E+00 (2.31E−09)	+	5.21E−10 (1.00E−09)	=
F10	3.71E+01 (1.27E+01)	**3.00E+01** (**3.58E−02**)	−	4.95E+03 (6.63E+03)	+	1.26E+04 (4.06E+03)	+	3.41E+01 (1.03E+01)	−	5.10E+01 (1.13E+01)	+	9.30E+03 (6.54E+03)	+
F11	4.17E−02 (8.79E−02)	**6.87E−05** (**3.05E−04**)	=	2.65E−02 (6.26E−02)	=	4.52E−02 (5.74E−02)	=	1.64E−01 (2.89E−01)	=	6.55E−02 (2.64E−02)	=	3.11E−03 (8.68E−03)	=
F12	4.52E−01 (2.59E−01)	3.98E+00 (1.74E+01)	+	1.66E−01 (5.91E−02)	−	5.04E+01 (7.69E+01)	+	**1.28E−04** (**3.43E−04**)	−	1.36E−01 (2.08E−02)	−	1.49E+01 (5.66E+01)	+
F13	1.19E+01 (4.76E+01)	3.54E+00 (2.21E+01)	−	8.08E+00 (4.17E+01)	−	4.98E+01 (1.31E+01)	+	**1.78E−13** (**1.93E−13**)	−	4.50E−13 (9.54E−14)	−	3.25E+02 (7.85E+01)	+
F14	**6.27E+01** (**2.88E+01**)	2.23E+02 (1.99E+02)	+	1.05E+02 (9.29E+01)	+	3.07E+02 (2.38E+01)	+	1.39E+02 (4.01E+01)	+	6.51E+02 (3.16E+01)	+	4.59E+02 (1.79E+02)	+
F15	**0.00E+00** (**0.00E+00**)	6.05E−12 (3.26E−11)	=	4.46E−14 (9.03E−14)	=	4.92E+01 (1.34E+01)	+	3.73E+00 (2.64E+01)	+	1.06E−06 (2.25E−06)	=	2.37E+02 (7.19E+01)	+

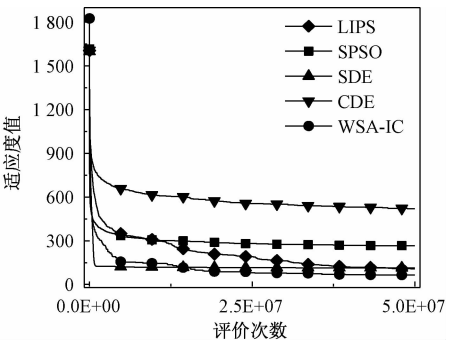

图 4　算法在函数 F14 上的收敛情况

Fig. 4　Convergence rate of algorithms on F14

(4)参数的影响. 稳定性阈值 T_s 是 WSA-IC 算法最为关键的参数,其值需根据优化问题的特点设定. 对取不同 T_s 值的 WSA-IC 算法进行了实验,结果如表 7 所示. 由表 7 可知,当 $T_s = 100n$ 时,WSA-IC 算法在大部分测试函数上都能取得最优的 ANOF 值. 因此,对于几乎所有的连续优化问题,T_s 均可以设置为 $100n$.

5　鲸鱼群算法的应用

目前鲸鱼群算法已在某些工程优化领域(如无线传感器网络能效优化)获得了成功的应用. 下面以炼钢连铸调度问题为例介绍鲸鱼群算法的具体应用.

5.1　炼钢连铸调度问题

以某钢厂的工艺流程为例介绍炼钢连铸调度问题. 如图 5 所示,有 N 个待加工浇次,各浇次包括一定数目的炉次,需按同一工艺路线(EAF-AOD-LF1-LF2-CC)进行加工,要求在保证连铸阶段相邻浇次预留准备时间,同一浇次内炉次连

表 7 WSA-IC 算法在不同的 T_s 取值下获得的 ANOF 值

Tab. 7 ANOF of WSA-IC with different values of T_s

函数序号	$T_s=20n$	$T_s=40n$	$T_s=60n$	$T_s=80n$	$T_s=100n$	$T_s=120n$	$T_s=140n$	$T_s=160n$	$T_s=180n$	$T_s=200n$
F1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
F2	**32**	**32**	**32**	**32**	**32**	**32**	**32**	**32**	31.98	**32**
F3	472.94	586.94	622.20	624.92	**625**	**625**	**625**	624.96	624.90	624.96
F4	0.69	0.92	0.96	0.98	**1**	**1**	**1**	**1**	**1**	**1**
F5	**125**	**125**	**125**	**125**	**125**	**125**	**125**	124.94	124.94	124.90
F6	**16**	**16**	**16**	**16**	**16**	15.98	15.94	15.94	15.96	15.96
F7	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	7.98	**8**
F8	**216**	215.98	**216**	**216**	**216**	215.92	215.88	215.88	215.88	215.82
F9	**5.69**	4.69	4.24	3.71	4.22	3.84	3.86	4.04	4.06	3.71
F10	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F11	**0.82**	0.76	0.63	0.65	0.80	0.73	0.67	0.73	0.73	0.67
F12	0.08	0.02	0.06	0.06	**0.14**	0.08	0.08	0.08	0.04	0.06
F13	0.69	0.82	0.63	0.67	**0.94**	0.61	0.69	0.49	0.51	0.49
F14	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F15	0.94	0.80	0.92	0.88	**1**	0.92	0.65	0.67	0.71	0.55

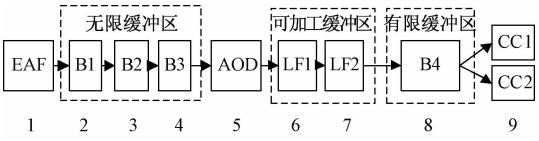

图 5 钢铁生产工艺流程图

Fig. 5 Flow chart of the steel production process

续加工的前提下,确定浇次顺序、连铸机分配和各工序的开始时间及结束时间,以最小化最大完工时间. 此外,特定工序间设置一定数量的无限缓冲区、有限缓冲区和可加工缓冲区[14]. 其中,无限缓冲区不限制炉次的缓存时间;有限缓冲区中炉次的缓存时间受限;可加工缓冲区既可加工炉次又可缓存炉次且缓存时间不受限. 这类问题早已获得某些学者的关注[14-15],但不同于已有的研究,笔者考虑连铸阶段存在 2 台并行机的情况,将缓冲区视为加工时间为 0 的生产阶段,则笔者所研究的炼钢连铸调度问题实质上是 9 阶段零等待混合流水车间调度问题.

5.2 基于 WSA-IC 的炼钢连铸调度算法

由于调度问题的离散特性,WSA-IC 算法不能直接用于求解炼钢连铸调度问题,因此从如下 4 个方面对 WSA-IC 算法进行了改进:①设计了一种离散的鲸鱼个体编码与解码方案;②采用汉明距离计算个体间的距离;③改进了鲸鱼个体的移动规则;④设计了一种有效的自适应邻域搜索策略. 由于该算法不需要保存在迭代过程中求得的多个当前全局最优解,所以该算法不需要 T_f 参数.

基于 WSA-IC 的炼钢连铸调度算法伪代码如下.

输入:目标函数,鲸鱼群 $\boldsymbol{\Omega}$,当前全局最优解 ***GBest***,当前全局最优解的适应度值 f_{gbest}.

输出:最优解.

```
开始
初始化参数;
初始化鲸鱼;
计算鲸鱼的适应度值;
while 终止条件不满足 do
    for i = 1 to |Ω| do
        寻找 Ω_i 的"较优且最近"的鲸鱼 Y;
        if Y 存在 then
            生成 Ω_i 的副本 X';
            X'根据个体移动规则向 Y 移动;
            计算 X'的适应度值 f(X');
            if f(X') < f(Ω_i) then
                Ω_i = X';
                Ω_i.c = 0;
            else
                if Ω_i.c ≠ T_s then
                    Ω_i.c = Ω_i.c + 1;
                else
                    重新初始化 Ω_i;
                    计算 Ω_i 的适应度值;
                end if
            end if
        else
            生成的 Ω_i 副本 X';
            对 X' 执行自适应邻域搜索;
            if f(X') < f(Ω_i) then
                Ω_i = X';
```

```
                Ω_i.c = 0;
            else
                if Ω_i.c ≠ T_s then
                    Ω_i.c = Ω_i.c + 1;
                else
                    if f(Ω_i) < f_gbest then
                        f_gbest = f(Ω_i);
                        GBest = Ω_i;
                    end if
                    重新初始化 Ω_i;
                    计算 Ω_i 的适应度值;
                end if
            end if
        end if
    end for
end while
```

判断最后一代种群中是否有比 ***GBest*** 好的鲸鱼,有则替换 ***GBest***;

返回最优解 ***GBest***;

结束

5.2.1　个体编码与解码

算法基于工件进行编码,即将浇次编号的排序作为编码.解码采用文献[14]所提方法,根据编码所示浇次序列,以浇次为单位安排作业计划,其中连铸机的分配依据最早空闲机器优先规则.

5.2.2　个体间的距离计算

在上述离散个体编码方案中,个体维度都相同,个体之间的距离计算与等长字符串之间的距离计算非常相似,故采用汉明距离[16]计算不同鲸鱼个体间的距离.假设两条鲸鱼个体 $\boldsymbol{X}_1$、$\boldsymbol{X}_2$ 如图 6 所示,$\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 有 4 个元素不相同,所以,$\boldsymbol{X}_1$ 和 $\boldsymbol{X}_2$ 之间的汉明距离为 4.

$$\boldsymbol{X}_1=(1,\begin{matrix}3\\1\end{matrix},\begin{matrix}2\\1\end{matrix},5,\begin{matrix}4\\1\end{matrix},\begin{matrix}6\\1\end{matrix})$$
$$\boldsymbol{X}_2=(1,\begin{matrix}2\\3\end{matrix},\begin{matrix}3\\3\end{matrix},5,\begin{matrix}6\\4\end{matrix},\begin{matrix}4\\1\end{matrix})$$

图 6　两条鲸鱼个体示意图

Fig. 6　Sketch map of two whales

5.2.3　个体移动规则

由于个体采用离散编码方案,WSA 算法的个体位置迭代式(1)不再适用,因此需针对炼钢连铸调度问题,改进个体移动规则.为了保证当前个体向它的"较优且最近"的个体随机移动,引入一个新的参数——选择概率 λ.鲸鱼 $\boldsymbol{\Omega}_u$ 的副本 $\boldsymbol{X}'$ 在它的"较优且最近"的鲸鱼 $\boldsymbol{Y}$ 的引导下,进行如下移动:首先比较个体 $\boldsymbol{X}'$ 和 $\boldsymbol{Y}$ 中对应位置的元素,如果不同,则将该元素的值和位置分别存入集合 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 中;遍历这些不同元素所在的位置,生成一个 0 到 1 之间的随机数 P,若 P 小于 λ 且 $\boldsymbol{Y}$ 中该位置的元素属于集合 $\boldsymbol{A}$(存储未被选择的元素),则将 $\boldsymbol{Y}$ 中该元素赋值给 $\boldsymbol{X}'$ 中对应位置的元素,否则从集合 $\boldsymbol{A}$ 中随机选择一个值赋给 $\boldsymbol{X}'$ 中对应位置的元素.注意到已经赋值给 $\boldsymbol{X}'$ 的元素应从集合 $\boldsymbol{A}$ 中移除.

5.2.4　自适应邻域搜索策略

为增强算法的局部搜索能力,当种群中不存在当前鲸鱼个体的"较优且最近"的鲸鱼时,对当前个体进行结合移位算子和交换算子的自适应邻域搜索.自适应邻域搜索方法将种群不同个体与生成该个体的邻域搜索算子关联起来.首先比较区间[0,1]内的随机数 a 和自适应邻域搜索概率 ϑ 的大小,若 $a<\vartheta$,选择关联的邻域算子进行局部搜索;反之,则从交换算子和移位算子中任选一种邻域算子实现局部搜索.

5.3　实例仿真与分析

为验证本文算法的有效性,随机生成 5 个不同规模的案例,选择遗传算法(genetic algorithm, GA)、人工蜂群算法(artificial bee colony, ABC)、传统 WSA 算法作为比较算法.WSA-IC 算法的参数设置如下:种群规模 PS 为 50,选择概率 λ 为 0.5,自适应邻域搜索概率 ϑ 为 0.75,稳定性阈值 T_s 为 $200+|Z|/2$($|Z|$为浇次数).所有算法以相同的迭代次数为终止条件,每组实验运行 10 次,结果见表 8.由表 8 知,对于测试的 5 个案例,WSA-IC 算法均能找到优于 ABC、GA 和 WSA 算法求得的解,并取得较小的标准差,且这种差异随着案例规模的增大而增大.由此说明,笔者提出的算法具有良好的寻优能力和稳定性.图 7 为所示算法的收敛曲线(横坐标为迭代次数,纵坐标为每次迭代求得的最优解适应度值).从图 7 可知,相较于 GA、ABC 和 WSA 算法,WSA-IC 算法能更快地收敛到更优的解.

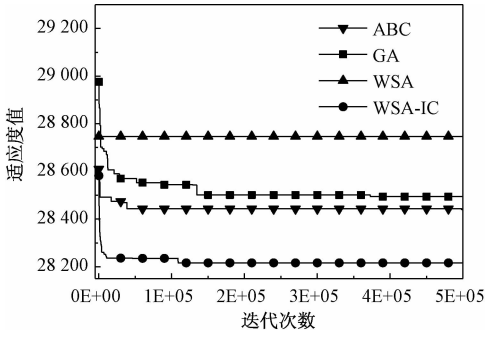

图 7　算法调度 90 个浇次时的收敛情况

Fig. 7　Convergence rate of algorithms for the case of scheduling 90 casts

表 8 WSA-IC 算法与其他算法的实验结果

Tab. 8 Experimental results of WSA-IC and other algorithms

浇次数	ABC		GA		WSA		WSA-IC	
	均值	标准差	均值	标准差	均值	标准差	均值	标准差
30	8 601.30	8.72	8 614.30	24.42	8 705.00	128.22	8 580.10	5.56
45	11 361.30	20.79	11 437.50	37.13	11 558.00	114.68	11 275.00	4.90
60	17 426.30	21.31	17 529.40	90.96	17 698.10	192.93	17 295.10	8.30
75	20 583.00	59.58	20 739.30	94.40	20 925.00	351.09	20 387.00	11.75
90	28 413.20	47.58	28 542.40	71.89	28 739.40	186.38	28 214.40	11.59

6 结束语

研究了一种新兴的群体智能算法——鲸鱼群算法(WSA).针对多峰函数优化问题的特点,提出了改进的鲸鱼群算法(WSA-IC).WSA-IC 改进了 WSA 的迭代规则,并引入了“稳定性阈值”和“适应度阈值”两个参数,使算法能在迭代过程中有效地识别并跳出已找到的极值点.结果表明,WSA-IC 在求解多峰函数优化问题上有着优异的性能,且 WSA-IC 的参数极易设置.此外,以炼钢连铸调度问题为例,阐述了 WSA 在工程优化领域的具体应用,表明 WSA 具有较大的应用价值.

当前关于 WSA 的研究仍处于起步阶段,未来可以将 WSA 与其他元启发式算法相结合以提高算法的优化性能;本文实验已证明 WSA 对多峰连续优化问题以及炼钢连铸调度问题的适用性,未来可以研究 WSA 在其他实际工程优化问题(特别是 NP-hard 问题)中的应用.

参考文献:

[1] ZHANG G H, GAO L, SHI Y. An effective genetic algorithm for the flexible job-shop scheduling problem [J]. Expert systems with applications, 2011, 38 (4): 3563 – 3573.

[2] 吴秀丽, 张志强. 求解柔性作业车间调度问题的细菌算法对比及改进[J]. 郑州大学学报(工学版), 2018, 39 (3): 34 – 39.

[3] KUILA P, JANA P K. A novel differential evolution based clustering algorithm for wireless sensor networks [J]. Applied soft computing, 2014, 25: 414 – 425.

[4] ELHABYAN R S Y, YAGOUB M C E. Two-tier particle swarm optimization protocol for clustering and routing in wireless sensor network [J]. Journal of network and computer applications, 2015, 52: 116 – 128.

[5] HOU E S H, ANSARI N, REN H. A genetic algorithm for multiprocessor scheduling [J]. IEEE transactions on parallel and distributed systems, 1994, 5(2): 113 – 120.

[6] QU B Y, SUGANTHAN P N, DAS S. A distance-based locally informed particle swarm model for multimodal optimization [J]. IEEE transactions on evolutionary computation, 2013, 17 (3): 387 – 402.

[7] ZENG B, GAO L, LI X Y. Whale swarm algorithm for function optimization [C]// Proceedings of the 13th international conference on intelligent computing. Liverpool: Springer, 2017: 624 – 639.

[8] 程适, 陈俊风, 孙奕菲, 等. 数据驱动的发展式头脑风暴优化算法综述[J]. 郑州大学学报(工学版), 2018, 39(3): 22 – 28.

[9] SHI Y H. An optimization algorithm based on brainstorming process [J]. International journal of swarm intelligence research, 2011, 2 (4): 35 – 62.

[10] LI X D. A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio [C]// Proceedings of the 9th Annual conference on Genetic and Evolutionary Computation. Washington: ACM, 2007: 78 – 85.

[11] LI X D. Efficient differential evolution using speciation for multimodal function optimization [C]// Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation. Washington: ACM, 2005: 873 – 880.

[12] THOMSEN R. Multimodal optimization using crowding-based differential evolution [C]// Proceedings of the 2004 Congress on Evolutionary Computation. Portland: IEEE, 2004: 1382 – 1389.

[13] LI X D. Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization [C]// Proceedings of the 6th Annual Conference on Genetic and Evolutionary Computation. Berlin: Springer, 2004: 105 – 116.

[14] 唐秋华, 郑鹏, 张利平, 等. 融合启发式规则和文化基因算法的多缓冲炼钢—连铸生产调度 [J]. 计算机集成制造系统, 2015, 21 (11): 2955 – 2963.

[15] PACCIARELLI D, PRANZO M. Production scheduling in a steelmaking-continuous casting plant [J]. Computers and chemical engineering, 2004, 28 (12): 2823 – 2835.

[16] HAMMING R W. Error detecting and error correcting codes [J]. Bell system technical journal, 1950, 29 (2): 147 – 160.

(下转第 35 页)

A Multi-swarm Artificial Bee Colony Algorithm for Function Optimization

WANG Shouna[1,2], LIU Hong[1,2], GAO Kaizhou[3]

(1. School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China; 2. Shandong Provincial Key Laboratory for Distributed Computer Software Novel Technology, Jinan 250358, China; 3. Maritime Institute, Nanyang Technological University, Singapore 639798, Singapore)

Abstract: A multi-swarm Artificial Bee Colony(MABC) algorithm based on the segmentation of population was proposed in this paper. It was applied to function optimization to overcome the drawbacks of slow convergence and low computational accuracy of conventional ABC algorithm. In this algorithm, K-means clustering algorithm based on Euclidean distance was introduced to divide the bee colony. In the subpopulation, a method was introduced to update the location of nectar based on global communication to accelerate the convergence of the algorithm; and the fitness function based on local communication was introduced to expand the diversity of the solution. The simulation results of six standard functions showed that the MABC algorithm could attain significant improvement on convergence rate and solution accuracy, and show better performance in function optimization problems when compared with the ABC algorithm.

Key words: Artificial Bee Colony algorithm; segmentation of population; nectar location updating; fitness function; function optimization

---

(上接第 22 页)

Improved Whale Swarm Algorithm and its Application in Steelmaking Continuous Casting Scheduling

ZENG Bing, WANG Mengyu, GAO Liang, DONG Haozhen

(School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: A new swam intelligent optimization algorithm named whale swarm algorithm (WSA) was studied. The principle and essential procedures of WSA were introduced; and the characteristics of WSA were presented through comparison with other classical swam intelligent optimization algorithms. For multimodal optimization, the iteration rule of WSA was improved, two parameters namely stability threshold and fitness threshold were introduced, and thus WSA with iterative counter (WSA-IC) was developed. The experimental results demonstrated that WSA-IC showed good performance in terms of the number and quality of optimal solutions and convergence speed. Then WSA-IC was applied to the steelmaking continuous casting scheduling problem, and proved to have good optimization ability and strong stability through the experiments. Finally, with the above research results, it was summarized that WSA had much value in practice, and further research of WSA could be carried out in theoretical study and practical application.

Key words: swam intelligent optimization algorithm; whale swarm algorithm; multimodal optimization; steelmaking continuous casting scheduling