[文章编号] 1007－7405(2021)01－0056－10 DOI:10.19715/j.jmuzr.2021.01.09

采用群智能算法的多约束问题优化

郜怀通[1]，王荣杰[1,2]，曾广淼[1]，刘文霞[1]

（1. 集美大学轮机工程学院，福建 厦门 361021；2. 福建省船舶与海洋重点实验室，福建 厦门 361021）

[摘要] 为了解决约束优化问题，采用一种基于群智能算法优化的多约束问题优化方法。首先构造同时计及约束条件和优化适应度的目标函数，然后分别利用粒子群算法和人工蜂群算法优化其函数，从而获得约束条件下的优化解。仿真结果表明，该多约束问题优化方法是可行性的，人工蜂群算法比粒子群算法具有更好的搜索和收敛能力。

[关键词] 人工蜂群算法；粒子群算法；约束函数

[中图分类号] TP 301

A Method to Deal with Multi Constraint Problems

GAO Huaitong[1], WANG Rongjie[1,2], ZENG Guangmiao[1], LIU Wenxia[1]

(1. School of Marine Engineering, Jimei University, Xiamen 361021, China;
2. Fujian Province Key Laboratory of Naval Architecture and Marine Engineering, Xiamen 361021, China)

Abstract: In order to solve the problem of constraint optimization, a multi-constraint optimization method based on swarm intelligence optimization algorithm is adopted. This method first constructs an objective function that takes into account both constraint condition and optimization fitness, and then the function is optimized by particle swarm algorithm and artificial bee colony algorithm respectively to obtain the optimized solution under the constraint condition. The simulation results of the test function show the feasibility of the multi-constraint optimization method proposed in this paper. In addition, the artificial bee colony algorithm has better search and convergence ability than the particle swarm algorithm.

Keywords: artificial bee colony algorithm; particle swarm optimization algorithm; constraint function

0 引言

为了应对越来越复杂的优化问题，人们基于生命系统，研究出了诸多优化算法，比如基于“优胜劣汰，适者生存”的进化类算法，典型代表有：通过遗传和变异进行求解的遗传优化算法（genetic algorithm，GA）[1]、模仿生物群体的合作和竞争进行求解的差分进化方法（differential evolution，DE）[2]、模仿生物体内免疫系统的进化进行求解的免疫算法（immune algorithm，IA）[3]等。模仿生物种群觅食和迁徙行为群智能算法典型代表有：模仿鸟群迁徙行为求解的粒子群优化算法（particle

[收稿日期] 2020－03－19

[基金项目] 国家自然科学基金项目（51879118）；福建省科技拥军项目（B19101）；福建省高等学校新世纪优秀人才支持计划（B17159）；农业部渔业装备与工程技术重点实验室基金项目（2016002，2018001）；人工智能四川省重点实验室基金项目（2017RJY02）；江苏省输配电装备技术重点实验室项目（2017JSSPD01）

[作者简介] 郜怀通（1997—），男，硕士生，从事智能信息处理研究。通信作者：王荣杰（1981—），男，教授，博士，从事智能信息处理与电力系统故障诊断研究。E-mail:roger811207@163.com

swarm optimization，PSO）[4]、模仿蜂群觅食行为求解的人工蜂群优化算法（artificial bee colony，ABC）[5]、模仿蚁群觅食行为求解的蚁群优化算法（ant colony optimization，ACO）[6]等。

国内外主要将群智能优化算法应用于解决多约束问题，Kamil 等[7]提出了一种使用群智能算法来增强水下图像颜色的方法；Guan 等[8]将改进的群算法用于船舶内壳的参数设计与优化，在保证船舶安全性的同时，极大地提高了船舶的载运能力；Zhang 等[9]将群智能算法用于图像分割，可以更加精确地从图像中提取有意义的特征，提高了图像识别的精度。

ABC 算法和 PSO 算法，在无约束目标优化中展现出了较好的收敛性和搜索性。本文使用这两种算法对多个约束优化问题进行处理。文中对约束处理有如下的创新：

1）对算法第 k 次迭代得到的优化解，将其满足约束条件的个数 $q(k)$ 设定为权值，进行优化解替换时，优先考虑权值的大小，使更新解始终满足 $q(k) \geqslant q(k-1)$，$k \geqslant 2$。

2）改进的群智能算法将步长由确定值改为每次迭代都会变化的随机值，增强了算法的搜索能力。

1　约束问题

约束优化问题的形式如式（1）所示：

适应度函数：$f(x_1,\cdots,x_n)$。

约束条件：

$$\begin{cases} g_l(x_1,\cdots,x_i) \geqslant y_l; z_i \leqslant x_i \leqslant o_i; \\ y \in \mathbf{R}; z \in \mathbf{R}; o \in \mathbf{R}; \\ i = 1,\cdots,n; l = 1,\cdots,m。 \end{cases} \tag{1}$$

式（1）中，存在 n 个自变量 $x_1,\cdots,x_n$，定义域为 $[z_i,o_i]$，求得的向量 $\boldsymbol{x} = [x_1,\cdots,x_n]$ 在使得 $f(x_1,\cdots,x_n)$ 的解最优的同时，需要满足所有约束函数 $g_j(x_1,\cdots,x_i)$ 的值域。

求解过程中，一次需要求出多组优化解，分别储存在向量 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_K$ 中，设定 $k = 1,\cdots,K$，如果向量 $\boldsymbol{x}_k$ 满足约束条件的个数大于向量 $\boldsymbol{x}_{k+1}$ 满足约束条件的个数，则 $\boldsymbol{x}_k$ 优于 $\boldsymbol{x}_{k+1}$；如果 $\boldsymbol{x}_k$ 满足约束条件的个数等于 $\boldsymbol{x}_{k+1}$ 满足约束条件的个数，则比较 $\boldsymbol{x}_k$ 与 $\boldsymbol{x}_{k+1}$ 的适应度函数，确定最优值。即以 $\boldsymbol{x}_k$ 满足约束条件的个数（即等式/不等式 $g_j(x)$ 成立的个数）作为最高评价准则，其次比较 $\boldsymbol{x}_k$ 的适应度函数值 $f(\boldsymbol{x}_k)$。

这就要求在通过算法求解的过程中需要建立评价函数，将每组自变量满足条件的个数，储存在向量 $\boldsymbol{a}_{\text{evaluate}}$ 中，用于选择优质的自变量。

本文选择解决方案由式（2）所示。

$$F(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}), & g_l(x) \geqslant y_l; \forall\, l = 1,\cdots,m; \\ f(\boldsymbol{x})_{\max} + \sum_{j}^{m} |g_l(\boldsymbol{x})|, & g_l(\boldsymbol{x}) < 0。 \end{cases} \tag{2}$$

其中：$F(\boldsymbol{x})$ 用来存放适应度函数；$f(\boldsymbol{x})_{\max}$ 是满足所有约束的解中可行性最差的解。如果求得的解均不能满足所有的约束条件，则 $f(\boldsymbol{x})_{\max}$ 取值为 0。

2　群智能算法

2.1　ABC 算法

ABC 算法主要分为初始化、工蜂（EB）、观察蜂（OB）、侦查蜂（SB）四个阶段[10-11]。初始化阶段设定蜂群的规模为 sn，维度为 n，使用式（3）随机生成初始解 $\boldsymbol{a}_{i,j}$，

$$\boldsymbol{a}_{i,j} = a\_\min_j + \text{rand}[0\ 1] \cdot (a\_\max_j - a\_\min_j)。 \tag{3}$$

其中：rand[0 1] 表示在 0 ~ 1 产生一个随机数；$a\_\max_j$ 和 $a\_\min_j$ 分别为蜜源 $\boldsymbol{a}$ 第 j 维的解能取到的最大值和最小值，$j = 1,2,\cdots,n$，$i = 1,2,\cdots,s_n$。将 $\boldsymbol{a}$ 存入矩阵 $\boldsymbol{v}$。

EB、OB 和 SB 阶段蜜源的位置由式（4）进行更新，

$$\boldsymbol{a}(i,j) = \boldsymbol{v}(i,j) + 2\{\mathrm{rand}[0\ 1] - 0.5\}[\boldsymbol{v}(i,j) - \boldsymbol{v}(r,j)]。 \tag{4}$$

其中：r 是 [1 sn] 之间随机的一个整数，且不等于 i。

ABC 算法的运算流程如图 1 所示。

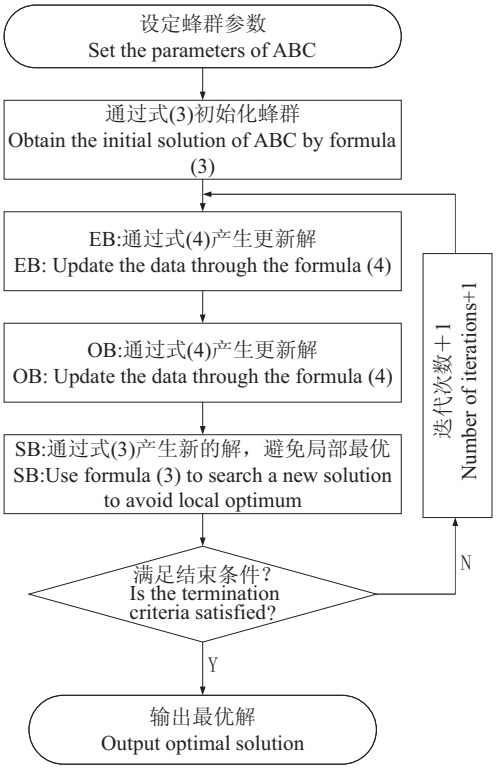


图 1 ABC 算法流程图

Fig.1 ABC algorithm flowchart

ABC 算法在实际应用中被发现了一些固有的缺点。例如，在处理单峰问题时，收敛速度比差分优化算法慢。除此之外，在解决复杂的多模态问题时，极容易陷入局部最优[13]。为了平衡 ABC 算法的探索和开发能力，众多研究人员开始对 ABC 算法进行改进。比如使用自适应方法改进 ABC 算法[13-14]，引入其他群智能算法的搜索方程改进 ABC 算法[15-16]。

本文中使用了文献［17］提出的 MABC 算法。MABC 算法参考了具有较强全局收敛能力和稳健性的 DE 算法[18]，引入了 DE 算法的搜索方程。MABC 算法的搜索方程如下所示。

EB 阶段：$\mathrm{eb}_{i,j} = a_{\min,j} + \mathrm{rand}[0\ 1](a_{\max,j} - a_{\min,j})$；

OB 阶段：$\mathrm{ob}_{i,j} = \mathrm{eb}_{i,j} + \mathrm{rand}[0\ 1](\mathrm{eb}_{r,j} - \mathrm{eb}_{i,j})$；

SB 阶段：$\mathrm{sb}_{i,j} = \mathrm{ob}_{i,j} + 2\{\mathrm{rand}[0\ 1] - 0.5\}(\mathrm{ob}_{i,j} - \mathrm{ob}_{k,j})$。

其中：$\mathrm{eb}_{i,j}$、$\mathrm{ob}_{i,j}$ $\mathrm{sb}_{i,j}$ 分别代表 EB、OB、SB 阶段的更新解；$a_{\min,j}$ 和 $a_{\max,j}$ 分别代表蜜源 $\boldsymbol{a}_{i,j}$ 第 j 维的最小值和最大值；r,l,k 皆为 [1 np] 之间一个随机的整数，且 $r \neq l \neq i$，$k \neq i$。

2.2 PSO 算法

PSO 算法的初始化方程如式（5）所示：

$$\begin{cases} \boldsymbol{a}_{i,j} = a_{\min,j} + \mathrm{rand}[0\ 1](a_{\max,j} - a_{\min,j}), \\ \boldsymbol{v}_{i,j} = v_{\min,j} + \mathrm{rand}[0\ 1](v_{\max,j} - v_{\min,j})。 \end{cases} \tag{5}$$

其中：$\boldsymbol{a}_{i,j}$ 是粒子群的初始位置；$\boldsymbol{v}_{i,j}$ 是粒子群的初始速度；$a_{\min,j}$ 和 $a_{\max,j}$ 分别是位于矩阵 $\boldsymbol{a}$ 第 j 维的最小值和最大值；$v_{\min,j}$ 和 $v_{\max,j}$ 分别是速度 $\boldsymbol{v}$ 第 j 维的最小值和最大值。

进行一次搜索之后，粒子群的位置和速度根据式（6）进行更新：

$$\boldsymbol{V}_{i,j} = w - \boldsymbol{v}_{i,j} + c_1\mathrm{rand}[0\ 1](\boldsymbol{p}_j - \boldsymbol{a}_{i,j}) + c_2\mathrm{rand}[0\ 1](\boldsymbol{g}_j - \boldsymbol{a}_{i,j}),\ \boldsymbol{X}_{i,j} = \boldsymbol{a}_{i,j} + \boldsymbol{V}_{i,j}。 \tag{6}$$

其中：$\boldsymbol{p}$ 是粒子群的个体最优位置；$\boldsymbol{g}$ 是粒子群的全局最优位置；w 是粒子的惯性权重，代表粒子有维持自己先前速度的趋势，用来平衡算法的全局收敛能力和局部收敛能力，一般取值为 [0.8,1.2]；c_1、c_2 是粒子群的学习因子，用来调节粒子向 $\boldsymbol{p}$ 和 $\boldsymbol{g}$ 方向的搜索长度，一般都取值为 1.5。

PSO 算法优化流程如图 2 所示。

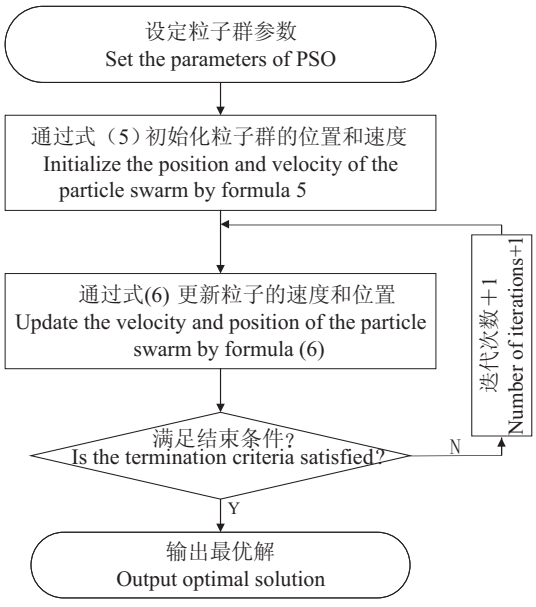


图 2 PSO 算法流程图

Fig.2 PSO algorithm flowchar

3　算法的多约束问题优化

3.1　ABC 算法的实现

使用 ABC 算法进行优化时，步骤如下。

步骤 1：设定蜂群的数量为 $s_n, i = 1,2,\cdots,s_n$，蜂群维度为 $n, j = 1,2,\cdots,n$。最大迭代次数设为 K；迭代次数计数器为 $k, k = 0,1,2,\cdots$。允许连续得不到更优解的次数设为 K_{limit}，连续得不到更优解的计数器为 $t, t = 0,1,2,\cdots$；创建存储矩阵 $\boldsymbol{G}$。

步骤 2：蜂群初始化。

1）设定第 j 维的优化解能取到的最大值为 $a\_\max_j$，最小值为 $a\_\min_j$；$k = 0$。

2）初始化蜂群，通过式（3）创建初始解 $\boldsymbol{a}$。将初始解 $\boldsymbol{a}$ 存入矩阵 $\boldsymbol{v}$，矩阵 $\boldsymbol{a}$、$\boldsymbol{v}$ 第 i 行数据分别表示为 $\boldsymbol{a}_i$ 和 $\boldsymbol{v}_i$。

3）设定适应度函数 $f(x)$，约束函数 $g_d(x)$，$d = 1,\cdots,D$，D 为约束函数的个数。设定第 i 组解满足约束函数个数的计数器为 dd_i，$dd_i = 1,\cdots,D$。满足的约束条件越多，适应度函数的值越靠近目标值，寻到的解越好。

4）计算矩阵 $\boldsymbol{v}$ 第 i 组解的 $f(x_i)$ 和 $dd_i, i = 1,2,\cdots,sn$。

步骤 3：EB 阶段。

1）若 $t \leqslant K_{\text{limit}}$，使用式（4）创建更新解 $\boldsymbol{a}$；若 $t \geqslant K_{\text{limit}}$，使用式（3）创建更新解 $\boldsymbol{a}$ 且令 $t = 0$。

2）如若出现 $\boldsymbol{a}_{i,j} > a\_\max_j$ 或 $\boldsymbol{a}_{i,j} < a\_\min_j$ 的情况，使用式（3）更新 $\boldsymbol{a}_{i,j}$。

3）计算更新解 $\boldsymbol{a}$ 第 i 组解的适应度函数值 $f\_eb(x_i)$ 和满足约束的个数 dd_eb_i，$i = 1,2,\cdots,sn$。

4）对比 dd_i 和 dd_eb_i，如果 $dd_i < dd\_eb_i$，将 $\boldsymbol{v}_i$ 替换为 $\boldsymbol{a}_i$；若 $dd_i > dd\_eb_i$ 保留 $\boldsymbol{v}_i$；若 $dd_i = dd\_eb_i$，对比 $f(x_i)$ 和 $f\_eb(x_i)$，若 $f\_eb(x_i)$ 比 $f(x_i)$ 更靠近目标值，将 $\boldsymbol{v}_i$ 替换为 $\boldsymbol{a}_i$，否则保留 $\boldsymbol{v}_i$。$i = 1,2,\cdots,s_n$。

5）若适应度函数最优值没有得到更新，$t = t + 1$。

6）若 $t > K_{\text{limit}}$，则跳至步骤 3 1）。

步骤 4：OB 阶段。

1）使用式（4）创建更新解 $\boldsymbol{a}$。

2）如若出现 $\boldsymbol{a}_{i,j} > a\_\max_j$ 或 $\boldsymbol{a}_{i,j} < a\_\min_j$ 的情况，使用式（3）更新 $\boldsymbol{a}_{i,j}$。

3）计算更新解 $\boldsymbol{a}$ 第 i 组解的适应度函数值 $f\_ob(x_i)$ 和满足约束的个数 dd_ob_i，$i = 1,2,\cdots,sn$。

4）对比 dd_i 和 dd_ob_i，如果 $dd_i < dd\_ob_i$，将 $\boldsymbol{v}_i$ 替换为 $\boldsymbol{a}_i$；若 $dd_i > dd\_ob_i$，保留 $\boldsymbol{v}_i$；若 $dd_i = dd\_ob_i$，对比 $f(x_i)$ 和 $f\_ob(x_i)$，若 $f\_ob(x_i)$ 比 $f(x_i)$ 更靠近目标值，将 $\boldsymbol{v}_i$ 替换为 $\boldsymbol{a}_i$，否则保留 $\boldsymbol{v}_i$。$i = 1,2,\cdots,sn$。

5）若适应度函数最优值没有得到更新，$t = t + 1$。

6）若 $t > K_{\text{limit}}$，则跳至步骤 3 1）。

步骤 5：SB 阶段。

1）使用式（4）创建更新解 $\boldsymbol{a}$。

2）如若出现 $\boldsymbol{a}_{i,j} > a\_\max_j$ 或 $\boldsymbol{a}_{i,j} < a\_\min_j$ 的情况，使用式（3）更新 $\boldsymbol{a}_{i,j}$。

3）计算更新解 $\boldsymbol{a}$ 第 i 组解的适应度函数值 $f\_sb(x_i)$ 和满足约束的个数 dd_sb_i，$i = 1,2,\cdots,sn$。

4）对比 dd_i 和 dd_sb_i，如果 $dd_i < dd\_sb_i$，将 $\boldsymbol{v}_i$ 替换为 $\boldsymbol{a}_i$；若 $dd_i > dd\_sb_i$，保留 $\boldsymbol{v}_i$；若 $dd_i = dd\_sb_i$，对比 $f(x_i)$ 和 $f\_sb(x_i)$，若 $f\_sb(x_i)$ 比 $f(x_i)$ 更靠近目标值，将 $\boldsymbol{v}_i$ 替换为 $\boldsymbol{a}_i$，否则保留 $\boldsymbol{v}_i$。$i = 1,2,\cdots,sn$。

5）若适应度函数最优值没有得到更新，$t = t + 1$。

6）若 $t > K_{\text{limit}}$，则跳至步骤 3 1）。

7）令 $k = k + 1$。

8）$G_{k,1\sim n}$ 储存第 k 次迭代中的最优解，$\boldsymbol{G}_{k,n+1}$ 储存对应的适应度函数值。

9）若 $k \geqslant K$，进行步骤 f；不然跳至步骤 3 1）。

10）输出最优解储存矩阵 $\boldsymbol{G}$。

3.2 PSO 算法的实现

使用 PSO 算法进行优化时，步骤如下。

步骤 1：设定粒子群的数量为 sn，$i=1$，2，…，sn，粒子群的维度为 n，$j=1,2,\cdots,n$。最大迭代次数设为 K；迭代次数计数器为 k，$k=0,1,2,\cdots$；惯性权重设为 ω；学习因子设为 c_1 和 c_2；创建存储矩阵 $\boldsymbol{G}$，储存个体最优位置的向量 $\boldsymbol{P}$，储存全局最优位置的向量 $\boldsymbol{g}$。

步骤 2：粒子群初始化。

1）设定第 j 维的优化解的能取到的最大值为 $a_{\max,j}$，最小值为 $a_{\min,j}$，最大速度为 $v_{\max,j}$，最小速度为 $v_{\min,j}$；$k=0$。

2）初始化粒子群，通过式（5）创建粒子群初始位置 $\boldsymbol{a}$，粒子群初始速度 $\boldsymbol{v}$，矩阵 $\boldsymbol{a}$ 和 $\boldsymbol{v}$ 的第 i 行数据分别表示为 $\boldsymbol{a}_i$ 和 $\boldsymbol{v}_i$。

3）设定适应度函数 $f(x)$，约束函数 $g_d(x)$，$d=1,\cdots,D$，D 为约束函数的个数。设定第 i 组解满足约束函数的个数的计数器为 dd_i，$dd_i=1,\cdots,D$。满足的约束条件越多，适应度函数的值越靠近目标值，寻到的解越好。

4）计算初始解 $\boldsymbol{a}$ 第 i 组解的 $f(x_i)$ 和 dd_i，$i=1,2,\cdots,sn$。

步骤 3：更新粒子群。

1）使用式（6）更新粒子群的速度 $\boldsymbol{V}$ 和位置 $\boldsymbol{A}$，矩阵 $\boldsymbol{A}$、$\boldsymbol{V}$ 的第 i 行数据分别表示为 $\boldsymbol{A}_i$ 和 $\boldsymbol{V}_i$。

2）如若出现 $V_{i,j}>v\_\max_j$，$V_{i,j}<v\_\min_j$，$A_{i,j}>a\_\max_j$ 或 $A_{i,j}<a\_\min_j$ 的情况，使用式（6）更新 $V_{i,j}$ 和 $A_{i,j}$。

3）计算更新解 $\boldsymbol{A}$ 第 i 组解的适应度函数值 $f\_A(x_i)$ 和满足约束的个数 dd_A_i，$i=1,2,\cdots,sn$。

4）对比 dd_i 和 dd_A_i，如果 $dd_i<dd\_A_i$，将 $\boldsymbol{a}_i$ 替换为 $\boldsymbol{A}_i$，$\boldsymbol{v}_i$ 替换为 $\boldsymbol{V}_i$；若 $dd_i>dd\_A_i$ 保留 $\boldsymbol{a}_i$ 和 $\boldsymbol{v}_i$；若 $dd_i=dd\_A_i$，对比 $f(x_i)$ 和 $f\_A(x_i)$，若 $f\_A(x_i)$ 比 $f(x_i)$ 更靠近目标值，将 $\boldsymbol{a}_i$ 替换为 $\boldsymbol{A}_i$，$\boldsymbol{v}_i$ 替换为 $\boldsymbol{V}_i$，否则保留保留 $\boldsymbol{a}_i$ 和 $\boldsymbol{v}_i$，$i=1,2,\cdots,sn$，更新向量 $\boldsymbol{p}$ 和 $\boldsymbol{g}$。

5）令 $k=k+1$。

6）$\boldsymbol{G}_{k,1\sim n}$ 储存第 k 次迭代中的最优解，$\boldsymbol{G}_{k,n+1}$ 储存对应的适应度函数值。

7）若 $k\geqslant K$，进行步骤 d；不然跳至步骤 3 1）。

步骤 4：输出最优解储存矩阵 $\boldsymbol{G}$。

4 仿真实验分析

本节中，使用 6 个测试函数，分别用三个算法进行计算，所有的问题中，都从不同的初始自变量开始运行 30 次，ABC 算法和 MABC 算法中的蜂群个数 sn 都设置为 20，PSO 中的粒子群个数 sn 设置为 100。

本文使用的 6 个测试函数如下所示。

测试函数 1。

适应度函数：$f_1(x)=(x_1^2+x_2-11)^2+(x_1+x_2^2-7)^2$。

约束函数：$g_1(x)=4.84-(x_1-0.05)^2-(x_2-2.5)^2\geqslant 0$；

$$g_2(x)=x_1^2+(x_2-2.5)^2-(x_2-2.5)^2\geqslant 0,\ 0\leqslant x_1\leqslant 6,\ 0\leqslant x_2\leqslant 6。$$

测试函数 2。

适应度函数：$f_2(x)=5\sum_{i=1}^{4}x_i-5\sum_{i=1}^{4}x_i^2-5\sum_{i=5}^{13}x_i$。

约束函数：$g_1(x) = 10 - (2x_1 + 2x_2 + x_{10} + x_{11}) \geqslant 0$；

$g_2(x) = 10 - (2x_1 + 2x_3 + x_{10} + x_{12}) \geqslant 0$；

$g_3(x) = 10 - (2x_1 + 2x_3 + x_{10} + x_{12}) \geqslant 0$；

$g_4(x) = -(-8x_1 + x_{10}) \geqslant 0$；

$g_5(x) = -(-8x_2 + x_{11}) \geqslant 0$；

$g_6(x) = -(-8x_3 + x_{12}) \geqslant 0$；

$g_7(x) = -(-2x_4 - x_5 + x_{10}) \geqslant 0$；

$g_8(x) = -(-2x_6 - x_7 + x_{11}) \geqslant 0$；

$g_9(x) = -(-2x_8 - x_9 + x_{12}) \geqslant 0$；

$0 \leqslant x_i \leqslant 1, i = 1, \cdots, 9, 0 \leqslant x_i \leqslant 100, i = 10, 11, 12, 0 \leqslant x_i \leqslant 1, i = 13$。

测试函数 3。

适应度函数：

$f_3(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$。

约束函数：$g_1(x) = 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geqslant 0$；

$g_2(x) = 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geqslant 0$；

$g_3(x) = 196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geqslant 0$；

$g_4(x) = -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geqslant 0$，

$-10 \leqslant x_i \leqslant 10, i = 1, 2, \cdots, 7$。

测试函数 4。

适应度函数：$f_4(x) = 5.357\,854\,7x_3^2 + 0.835\,689\,1x_1x_5 + 37.283\,239x_1 - 40\,792.141$。

约束函数：$g_1(x) = 85.334\,407 + 0.005\,685\,8x_2x_5 + 0.000\,626\,2x_1x_4 - 0.002\,205\,3x_3x_5 \geqslant 0$；

$g_2(x) = 92 - (85.334\,407 + 0.005\,685\,8x_2x_5 + 0.000\,626\,2x_1x_4 - 0.002\,205\,3x_3x_5) \geqslant 0$；

$g_3(x) = 80.512\,49 + 0.071\,317x_2x_5 + 0.002\,995\,5x_1x_2 + 0.002\,181\,3x_3 - 90 \geqslant 0$；

$g_4(x) = 110 - (80.512\,49 + 0.0713\,17x_2x_5 + 0.002\,995\,5x_1x_2 + 0.002\,181\,3x_3) \geqslant 0$；

$g_5(x) = 9.300\,961 + 0.004\,702\,6x_3x_5 + 0.001\,254\,7x_1x_3 + 0.001\,908\,5x_3x_4 - 20 \geqslant 0$；

$g_6(x) = 25 - (9.300\,961 + 0.004\,702\,6x_3x_5 + 0.001\,254\,7x_1x_3 + 0.001\,908\,5x_3x_4) \geqslant 0$，

$78 \leqslant x_1 \leqslant 102$，$33 \leqslant x_2 \leqslant 45$，$27 \leqslant x_i \leqslant 45, i = 3, 4, 5$。

测试函数 5。

适应度函数：$f_5(x) = -0.5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7)$。

约束函数：$g_1(x) = -(x_3^2 + x_4^2 - 1) \geqslant 0$；

$g_2(x) = -(x_9^2 - 1) \geqslant 0$；

$g_3(x) = -(x_5^2 + x_6^2 - 1) \geqslant 0$；

$g_4(x) = -(x_1^2 + (x_2 - x_9)^2 - 1) \geqslant 0$；

$g_5(x) = -((x_1 - x_5)^2 + (x_2 - x_6)^2 - 1) \geqslant 0$；

$g_6(x) = -((x_1 - x_7)^2 + (x_2 - x_8)^2 - 1) \geqslant 0$；

$g_7(x) = -((x_3 - x_5)^2 + (x_4 - x_6)^2 - 1) \geqslant 0$；

$g_8(x) = -((x_3 - x_7)^2 + (x_4 - x_8)^2 - 1) \geqslant 0$；

$g_9(x) = -(x_7^2 + (x_8 - x_9)^2 - 1) \geqslant 0$；

$g_{10}(x) = -(x_2x_3 - x_1x_4) \geqslant 0$；

$g_{11}(x) = x_3x_9 \geqslant 0$；

$g_{12}(x) = -x_5x_9 \geqslant 0$；

$g_{13}(x) = -(x_6x_7 - x_5x_8) \geqslant 0$；

$-10 \leqslant x_i \leqslant 10, i = 1, \cdots, 8, 0 \leqslant x_9 \leqslant 20$。

测试函数 6。

适应度函数：$f_6(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 + x_3)^2 + 2(x_6 - 1)^2 + 5x_7 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$。

约束函数：$g_1(x) = 105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geqslant 0$；

$g_2(x) = -10x_1 + 8x_2 + 17x_7 - 2x_8 \geqslant 0$；

$g_3(x) = 8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geqslant 0$；

$g_4(x) = -3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geqslant 0$；

$g_5(x) = -5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geqslant 0$；

$g_6(x) = -x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + x_6 \geqslant 0$；

$g_7(x) = -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geqslant 0$；

$g_8(x) = 3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geqslant 0$，

$10 \leqslant x_i \leqslant 10, i = 1, \cdots, 10$。

使用 MABC、ABC、PSO 算法分别对测试函数进行优化计算，每个函数计算 30 次，对这 30 次计算的最优值、最差值、中值、平均值进行统计，结果如表 1 所示。

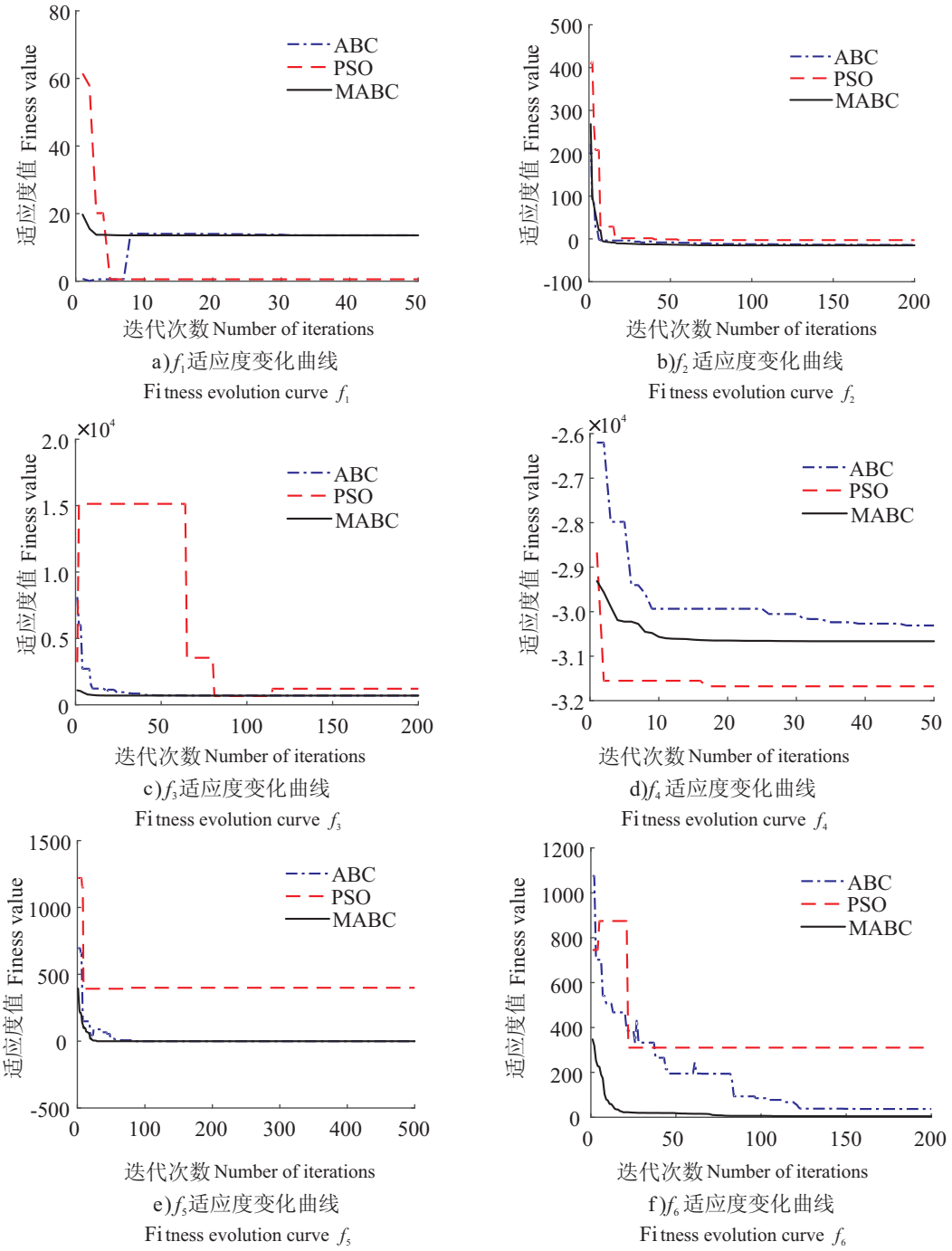
表 1 优化结果

Tab. 1 Optimized results

测试函数 Test function		ABC		MABC		PSO	
		数值 Numerical value	权值 Weight	数值 Numerical value	权值 Weight	数值 Numerical value	权值 Weight
f_1	最优 Optimal value	13.591 2	2.000 0	13.590 8	2.000 0	0.836 5	1.000 0
	最差 Worst value	49.715 7	2.000 0	13.590 8	2.000 0	1 001.879 5	1.000 0
	中值 Median value	13.656 2	2.000 0	13.590 8	2.000 0	167.074 3	1.000 0
	平均 Average value	17.479 4	2.000 0	13.590 8	2.000 0	275.957 1	1.000 0
f_2	最优 Optimal value	-14.600 2	9.000 0	-15.000 0	9.000 0	-5.250 8	9.000 0
	最差 Worst value	-13.762 3	9.000 0	-7.185 7	9.000 0	-2.672 1	9.000 0
	中值 Median value	-14.145 2	9.000 0	-14.453 0	9.000 0	-3.764 5	9.000 0
	平均 Average value	-14.156 2	9.000 0	-13.691 5	9.000 0	-3.844 4	9.000 0
f_3	最优 Optimal value	697.983 4	4.000 0	697.742 7	4.000 0	1 434.953 9	4.000 0
	最差 Worst value	700.524 9	4.000 0	724.302 2	4.000 0	4 962 416.201 8	4.000 0
	中值 Median value	698.630 3	4.000 0	697.845 2	4.000 0	7 061.146 1	4.000 0
	平均 Average value	698.750 5	4.000 0	698.911 1	4.000 0	251 496.118 7	4.000 0
f_4	最优 Optimal value	-30 330.618 8	6.000 0	-30 665.531 7	6.000 0	-29 949.572 5	6.000 0
	最差 Worst value	-30 004.107 6	6.000 0	-30 663.411 9	6.000 0	-30 767.540 3	4.000 0
	中值 Median value	-30 276.791 6	6.000 0	-30 665.473 5	6.000 0	-25 641.293 8	6.000 0
	平均 Average value	-30 247.241 4	6.000 0	-30 665.232 5	6.000 0	-27 706.051 3	5.600 0
f_5	最优 Optimal value	-0.857 5	13.000 0	-0.866 0	13.000 0	-33.607 5	6.000 0
	最差 Worst value	-0.614 8	13.000 0	-0.033 8	13.000 0	37.830 9	5.000 0
	中值 Median value	-0.839 0	13.000 0	-0.860 8	13.000 0	-21.036 7	5.000 0
	平均 Average value	-0.820 8	13.000 0	-0.744 2	13.000 0	-21.035 4	5.033 3
f_6	最优 Optimal value	29.826 1	8.000 0	24.559 7	8.000 0	877.146 4	6.000 0
	最差 Worst value	300.754 6	7.000 0	33.586 1	8.000 0	812.887 1	1.000 0
	中值 Median value	35.769 6	8.000 0	25.570 6	8.000 0	4 006.045	4.000 0
	平均 Average value	54.022 8	7.966 7	26.185 5	8.000 0	1 236.804	3.500 0

由表 1 可得，MABC 算法和 ABC 算法对约束问题的处理能力强于 PSO 算法。同样的迭代次数，MABC 算法得到的解优于 ABC 算法得到的解。由约束函数 5 可知，PSO 算法会陷入局部最优，导致结果会出现较大的波动。

将约束函数每次迭代得到的最优值进行统计，绘制出每个约束函数的适应度进化曲线，用来评价函数的收敛能力，如图 3 所示。



a)f_1适应度变化曲线　Fitness evolution curve f_1

b)f_2适应度变化曲线　Fitness evolution curve f_2

c)f_3适应度变化曲线　Fitness evolution curve f_3

d)f_4适应度变化曲线　Fitness evolution curve f_4

e)f_5适应度变化曲线　Fitness evolution curve f_5

f)f_6适应度变化曲线　Fitness evolution curve f_6

图 3　适应度进化曲线

Fig.3　Fitness evolution curve

如图 3a 和图 3e 所示，f_1 和 f_5 中，ABC 算法和 MABC 算法均寻到了满足全部约束的解，MABC 算法的收敛能力优于 ABC 算法的收敛能力。PSO 算法陷入了局部最优，寻不到满足所有约束的解。

如图 3b 所示，约束函数 2 中，三种算法都寻到了满足全部约束条件的解，PSO 算法得到的解相较于其他两种算法差。MABC 算法对最优值的搜索能力优于 ABC 算法。

如图 3c 所示，约束函数 3 中，三个算法寻找到的最优解皆能满足全部约束条件。搜索过程中因为以满足约束条件作为第一标准，PSO 算法出现了大程度的波动。三个算法中最优值仍然由 MABC 算法寻到。

如图 3d 和图 3f 所示，约束函数 4 和约束函数 6 中，MABC 算法和 ABC 算法都搜索到了满足所有约束函数的解，但 MABC 算法搜索到的解明显优于 ABC 算法搜索到的解。PSO 算法陷入局部最优，没能搜索到满足全部约束的解。

在群智能算法中种群数量的大小对算法的收敛能力和搜索能力有着较大的影响，上文使用的 MABC 算法的蜂群数量都为 20，为了进一步分析种群数量对 MABC 算法性能的影响，将蜂群数量分别更改为 5、10、15、25、30、50，并统计了这 7 种情况取的最优解所需的迭代次数，如表 2 所示。

由表 2 可得：在相同的迭代次数的情况下，蜂群数量为 5 和 10 时，收敛和搜索能力，相较于上文选择的 20 都有明显的下降，容易出现不能满足全部约束的情况，容易出现局部最优的情况；蜂群数量为 15，在进行足够迭代次数的情况下，可以取得最优解；蜂群数量为 25 和 30 时，对仿真结果的收敛速度有一定的影响，但均能得到让人满意的最优解；蜂群数量为 50 时，算法会在较少迭代次数的情况下得到最优解，收敛曲线会在开始就出现断崖式下跌，不利于比较算法的优化能力。最后的仿真结果发现群智能算法中的种群数量对算法收敛和搜索的能力有着极大的影响，优化过程中应通过多次尝试，尽可能地选择合适的种群数量。

表 2 种群数量对算法优化能力的影响

Tab. 2 Influence of population number on optimization ability of algorithm

种群数量 Population size	f_1	f_2	f_3	f_4	f_5	f_6
5	–	–	–	–	–	–
10	44	642	–	94	–	–
15	20	265	241	58	474	855
20	17	172	147	33	210	149
25	13	125	129	34	266	176
30	11	139	140	29	246	133
50	5	72	47	14	93	83
最优解 Optimal value	13.5908	–15.0000	697.7427	–30665.5317	–0.8660	24.5597

5 结论

本文提出利用群智能算法处理多约束优化问题。首先构造同时计算约束条件和优化适应度的目标函数，然后建立约束问题的数学模型，进行 ABC 算法和 PSO 算法的仿真实验。仿真结果可知，MABC 算法的收敛和搜索能力强于 ABC 算法和 PSO 算法；PSO 算法在约束问题的处理上，容易陷入局部最优，普适性弱于 ABC 和 MABC 算法。本文基于群智能算法的多约束问题优化方法可以广泛应用于油船内壳结构优化，规划无人机的航行轨迹等工程领域。

[参 考 文 献]

[1] VLAŠIĆ I, DURASEVIĆ M, JAKOBOVIĆ D. Improving genetic algorithm performance by population initialisation with dispatching rules [J]. Computers & Industrial Engineering, 2019,137: 106030. 1-106030. 15.

[2] LI E. An adaptive surrogate assisted differential evolutionary algorithm for high dimensional constrained problems [J]. Applied Soft Computing, 2019,85: 105282. DOI:10.1016/j.knosys.2019.105282.

[3] LIN Q, MA Y, CHEN J, et al. An adaptive immune-inspired multi-objective algorithm with multiple differential evolution

strategies [J]. Information Sciences, 2017,430: 46-64. DOI:10.1016/j.ins.2017.11.030.
[4] TSAI H C, TYAN YY, WU Y W. Isolated particle swarm optimization with particle migration and global best adoption [J]. Engineering Optimization, 2012, 44(12): 1405-1424.
[5] KONGD , CHANG T , DAI W. An improved artificial bee colony algorithm based on elite group guidance and combined breadth-depth search strategy [J]. Information Sciences, 2018, 5(442/443): 54-57.
[6] PANIRI M, DOWLATSHAHI M B, NEZAMABADI-POUR H. MLACO: A multi-label feature selection algorithm based on ant colony optimization [J]. Knowledge-Based Systems, 2020, 192(15): 105285.
[7] KAMIL Z M A, AHMAD S A G. Natural-based underwater image color enhancement through fusion of swarm-intelligence algorithm [J]. Applied Soft Computing, 2019(85): 105810.
[8] GUAN G, QU Y. A new method for parametric design and optimization of ship inner shell based on the improved particle swarm optimization algorithm. [J]. Ocean Engineering, 2018(169): 551-566.
[9] ZHANG X, WANG D H. Application of artificial intelligence algorithms in image processing [J]. Journal of Visual Communication and Image Representation, 2019(61): 42-49.
[10] MEI Z, YING-TONG T, JIN-HUI Z. Modeling and simulation of improved artificial bee colony algorithm with data-driven optimization [J]. Simulation Modelling Practice and Theory, 2018(93): 305-321.
[11] KONG D, CHANG, DAI W. An improved artificial bee colony algorithm based on elite group guidance and combined breadth-depth search strategy [J]. Information Sciences, 2018(442/443): 54-71.
[12] SONG X, ZHAO M, YAN Q. A high-efficiency adaptive artificial bee colony algorithm using two strategies for continuous optimization [J]. Swarm and Evolutionary Computation, 2019, 11(50): 1-23
[13] LIANG Z, HU K, ZHU Q, et al. An enhanced artificial bee colony algorithm with adaptive differential operators [J]. Applied Soft Computing, 2017(58): 480-494.
[14] SONG X, YAN Q, ZHAO M. An adaptive artificial bee colony algorithm based on objective function value information [J]. Applied Soft Computing, 2017(55): 384-401.
[15] CUI L, LI G, LIN Q. A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation [J]. Information Sciences, 2016(367/368): 1012-1044.
[16] IMANIAN N, SHIRI M E, MORADI P. Velocity based artificial bee colony algorithm for high dimensional continuous optimization problems [J]. Engineering Applications of Artificial Intelligence, 2014(36): 148-163.
[17] GAO W F, LIU S Y. A modified artificial bee colony algorithm [J]. Computers and Operations Research, 2012, 39(3): 687-697.
[18] AL-DABBAGH R D, NERI F, IDRIS N. Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy [J]. Swarm and Evolutionary Computation, 2018(43): 284-311.

(责任编辑　陈　敏　英文审校　郑青榕)