基于改进樽海鞘群算法的测试数据自动生成

徐 良[1] 田青云[1] 文 成[1] 张海波[1] 郭晶晶[1]
XU Liang TIAN Qingyun WEN Cheng ZHANG Haibo GUO Jingjing

摘 要 为了提高软件测试数据自动生成的可靠性，研究了一种改进的樽海鞘群算法自动生成测试数据。首先对樽海鞘群算法引入了人工鱼群算法中的随机行为，改善了樽海鞘个体容易陷入局部最优的问题，充分平衡迭代过程中的探索行为与开发行为；然后根据樽海鞘个体寻优结果引入末位淘汰机制，选择性舍弃适应度值最低的个体，并在搜索空间内随机生成一个新的个体进行种群补充；最后将改进后的樽海鞘群算法应用于基准程序的测试数据自动生成。实验结果表明，改进后的算法能够有效改善个体容易陷入局部最优的问题，正确搜寻到满足测试条件的数据，具有一定的优越性。

关键词 软件测试；樽海鞘群算法；随机行为；末位淘汰；测试数据自动生成

doi：10.3969/j.issn.1672-9528.2024.01.025

0 引言

软件测试保障了软件的可靠性与稳定性，随着软件功能的迭代，程序规模越来越庞大，传统依靠人工设计测试用例会消耗大量的时间与财力，并且存在效率低、难以进行全路径覆盖等问题。随着智能优化算法的发展，将其应用于软件测试数据的自动生成逐渐成为软件测试领域的热门研究。

针对测试数据自动生成问题，目前各种研究方法大体上可以分为随机法、静态法、动态法、基于搜索的方法[1]。其中基于搜索的方法主要使用启发式算法实现测试数据的自动生成。首先将测试数据自动生成问题转化成目标优化问题，再通过算法寻优生成测试数据是目前智能优化算法自动生成测试数据最主要的应用方式。XANTHAKIS 等人[2]早在 1992 年就将遗传算法应用于软件测试自动生成过程中；苗晓旭等人[3]对蚁群算法进行改进，并将其应用于测试数据自动生成中，改善了测试数据自动生成搜索效率低的问题；Kumar 等人[4]将人工蜂群算法、标准粒子群算法、遗传算法分别用于生成路径测试数据；王培崇等人[5]对人工鱼群算法进行改进，并应用于软件路径测试数据自动生成；廖伟志等人[6]研究了一种基于蚁群算法的多路径覆盖测试数据生成方法，有效提高了测试路径覆盖率；佘嘉纯等人[7]在人工鱼群算法中引入动态步长，改善了测试数据自动生成过程中迭代速度慢的问题。

樽海鞘群算法（SSA）是由 Seyedal 等人[8]在 2017 年提出的一种新型启发式算法，该算法基于自然界中生物的群体行为，其灵感来源于樽海鞘这种群体生物的行为。樽海鞘是一种海洋生物，它们的个体通常会聚集在一起形成群体，从而完成一些复杂的行为。群体中并非所有个体都向着食物浓度最大处移动，每个个体都会跟随着自己的前一个个体移动，个体的移动状态主要依赖于前后相邻个体的移动状态。SSA 的控制参数少，因此该算法运行效率较高[9]，但当某个体陷入局部最优时可能影响其相邻个体也陷入局部最优，最终可能导致整个群体止步不前。为了缓解此问题，对 SSA 引入人工鱼群算法中的随机行为，并执行“末位淘汰”机制，选择性地舍弃每次迭代后适应度值最低的个体，并在搜索范围内随机生成一个新的个体进行补充，增强种群活力。最后将改进后的 SSA（ISSA）应用于测试数据自动生成中，与改进前的算法运行结果进行对比，证明 ISSA 的有效性和优越性。

1 SSA 算法

1.1 算法描述

SSA 是受海洋中樽海鞘的成群行为启发，樽海鞘是一种与水母相似的透明生物，它们最主要的行为之一就是聚群。在深海中，为了通过快速协调变化和觅食实现更好的运动，樽海鞘群通常首尾相接形成一个称为樽海鞘链的群体。群体中对环境有最优判断的领导者在链条的最前端，其余个体均服从领导者的行动，当领导者由于前期搜索不充分陷入局部最优时，跟随它的个体也将无法到达全局最优位置。除了领导者之外，其余每个樽海鞘个体都只受自己前一个个体的影响，避免出现“越级管理”的情况。因此，领导者对于后面的个体影响程度层层递减，位置越靠后的个体受到领导者影响的程度越小，每个个体都能保持自身的多样性，由此在一

1. 信阳学院 河南信阳 464000

[基金项目] 信阳学院校级科研项目一般项目（2022-XJLYB-020）

定程度上避免了领导者陷入局部最优时其余个体也都停止移动而使得整体陷入局部最优。

1.2 数学模型

1.2.1 对应关系

为了建立樽海鞘链的数学模型，设置种群中最前端的樽海鞘为领导者，其余的樽海鞘为追随者。SSA 中的变量与优化问题对应关系如表 1 所示。

表 1 SSA 变量与优化问题对应关系

SSA	优化问题
樽海鞘	候选解
海洋范围	搜索空间
适应度值	解的质量
排在第一的樽海鞘	全局最优解

1.2.2 初始化种群

樽海鞘群中的所有个体存在于一定范围内，每个樽海鞘对应的位置信息可以看做问题的一个候选解，假设一个樽海鞘群共有 n 个个体，问题的变量数，即问题的搜索空间维度有 d 个，则所有樽海鞘的位置可以存储在一个二维矩阵 $\boldsymbol{X}$ 中，如式（1）所示。

$$\boldsymbol{X}=\begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & x_d^1 \\ \vdots & \vdots & \cdots & \vdots \\ x_1^n & x_2^n & \cdots & x_d^n \end{bmatrix} \tag{1}$$

$$x_j^i = rand \times \left(u_{bj} - l_{bj}\right) + l_{bj} \\ i \in \{1,2,...,n\}, j \in \{1,2,...,d\} \tag{2}$$

式中：$\boldsymbol{X}$ 是一个 $n \times d$ 的矩阵（n 代表种群规模，即有多少个候选解；d 代表问题的维度），u_b 表示问题解的上限，l_b 表示问题解的下限，rand 表示生成的一个 0~1 的随机数。

1.2.3 领导者和跟随者位置更新

初始化种群后，算法对每个樽海鞘进行适应度的评估，并按照适应度值进行排序，适应度最高的，排在第一位。算法会将樽海鞘群分为两组：领导者和追随者。领导者是链条前部的樽海鞘（排名靠前的樽海鞘），而其余的樽海鞘则是追随者 [10]。领导者更新公式为：

$$x_j^{(i)} = \begin{cases} F_j + r_1\left(\left(u_{bj} - l_{bj}\right) \times r_2 + l_{bj}\right) & if\ r_3 < 0.5 \\ F_j - r_1\left(\left(u_{bj} - l_{bj}\right) \times r_2 + l_{bj}\right) & if\ r_3 \geq 0.5 \end{cases} \tag{3}$$

$$r_1 = 2e^{-(\frac{4t}{T})^2} \tag{4}$$

式中：j 表示问题的维度，F_j 表示第 j 维度的全局最优位置，即食物的浓度最大的位置，$x_j^{(i)}$ 表示第 i 个樽海鞘的第 j 个维度，r_1 是 SSA 的收敛因子，随着算法迭代收敛因子逐渐减小，t 表示当前迭代次数，T 表示最大迭代次数。r_2 和 r_3 都是一个 0~1 的随机数，r_3 用于控制领导者在两种更新方式中进行选择。跟随者更新公式为：

$$x_j^i = \frac{1}{2}\left(x_j^i + x_j^{i-1}\right) \tag{5}$$

式（5）表示跟随者樽海鞘会向自己前一个樽海鞘移动，移动的距离是自己到对方距离的一半。

1.2.4 改进的 SSA 算法

在前面 SSA 基础上，除领导者之外，对种群中的所有个体引入人工鱼群算法中的随机行为，即个体经过 try_number 次数迭代后适应度值仍然没有改善，说明该个体陷入局部最优的可能性较大，则在距离前一个个体范围内随机搜寻一个新位置，并计算新位置适应度值，若新位置适应度值大于当前位置，则移动至新位置，移动规则公式为：

$$x_j^i = x_j^{i-1} + rand \times \left(x_j^i - x_j^{i-1}\right) \tag{6}$$

对种群引入“末位淘汰”制，首先计算出每次迭代结束后所有个体适应度值。然后在搜寻范围内随机生成一个新的个体，计算新个体适应度值，并与现有种群中适应度值最小的个体进行对比。若新个体适应度值大于种群当前最小适应度值，则舍弃掉种群中适应度值最小的个体，将新个体加入到种群中。最后根据适应度值将整个种群重新排序。

1.3 算法运行步骤

ISSA 具体运行步骤如下。

（1）初始化种群。根据解空间的大小，设置每个维度解的上下限，使用式（2）初始化一个规模为 $n \times d$ 的樽海鞘群位置矩阵。

（2）计算初始适应度值。计算每个樽海鞘所在位置的初始适应度值，即初始食物浓度。

（3）选定最佳位置与领导者。根据樽海鞘所在位置的食物浓度将樽海鞘按照浓度从高到低的顺序依次排序，排在首位的樽海鞘处于当前食物浓度最大的位置。

（4）选定领导者。将处在樽海鞘链第一位的个体设置为群体领导者，其余个体设置为追随者。

（5）位置更新。根据式（3）和式（4）更新领导者位置，再根据式（5）更新每个追随者位置。

（6）计算适应度值。计算更新位置后的每个个体所处位置的食物浓度，如某个个体更新后位置的食物浓度高于更新前位置，则将该个体移动至更新后的位置。若某个个体经过 try_number 次迭代后适应度值仍然没有改善，则按照式（6）执行随机行为。在搜寻范围内随机生成一个新个体，计算新个体适应度值，对比新个体与现有种群中适应度值最低的个体，选择是否执行“末位淘汰”。所有个体完成更新后再根据食物浓度的高低对整个种群重新排序，然后重复执行步骤

（4）~（6），直到迭代次数达到设定的最大次数，或适应度值达到设定值，输出达到终止条件后适应度值最佳的位置数据作为本次 ISSA 迭代的最优解。

ISSA 具体执行流程图如图 1 所示。

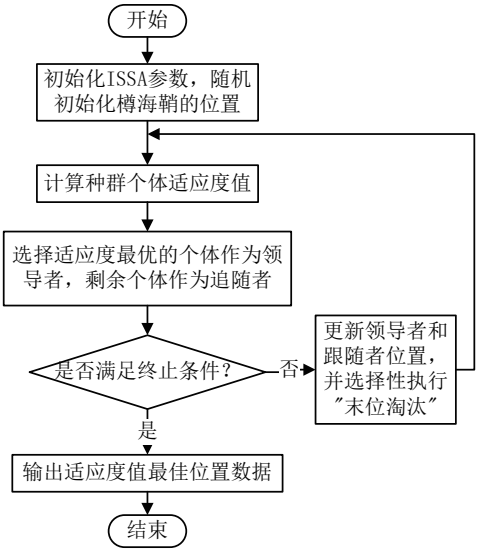


图 1 ISSA 执行流程图

2 仿真条件分析

2.1 构造适应度函数

根据分支函数的构造方式，通过路径覆盖和分支覆盖来设计适应度函数。首先分析被测程序，然后对被测程序的各个分支点进行函数插桩，插入的分支函数为 $f(1), f(2), \cdots, f(n)$，n 表示分支数，从而得到插桩程序 [11]。在程序执行时，根据计算出的分支函数的值就可以判断测试用例是否执行了所选逻辑路径。根据 KOREL 分支函数理论 [12]，分支函数的值越小，表明执行所选的分支越多，若某个分支函数的值为 0，说明该分支点被执行。因此为了得出测试用例的分支覆盖情况，本文采用“分支叠加法”适应度函数用于评价樽海鞘个体所处位置的优劣。

程序中的每个分支都可以用分支谓词 [13] 来表示，如判断语句“if (x > y){…}”中的分支谓词就是 $x>y$，可以把分支谓词抽象为：

$$E_1 \ \text{op} \ E_2 \tag{7}$$

式中：E_1、E_2 为算术表达式，op 为 {<，≤，>，≥，=，≠ } 中的一个。分支函数可以将分支谓词转换成实值，即分支函数值，用以衡量实际执行路径和逻辑执行路径的偏差。常见分支谓词的分支函数创建方法如文献 [13] 所示，通过文献 [13] 可知，当某个路径上的分支谓词为真且分支函数之和为 0 时，说明测试数据完全覆盖了该路径上的指定分支。

计算适应度函数之前首先对被测程序进行静态分析，确定要覆盖的目标路径；然后将分支函数以插桩的形式插入到目标路径的所有分支结点；最后将这些结点的分支函数相加即可得到适应度函数的值。假设目标路径中包含 m 个分支，n 个输入参数 $(x_1, x_2, \cdots, x_n)$，则分支函数为：

$$\begin{aligned} f_1 &= f_1(x_1, x_2, \ldots, x_n) \\ f_2 &= f_2(x_1, x_2, \ldots, x_n) \\ &\cdots \\ f_m &= f_m(x_1, x_2, \ldots, x_n) \end{aligned} \tag{8}$$

适应度函数值为各分支函数相加的结果，因此可得适应度函数 F 如式（9）所示。

$$F(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{m} f(x_1, x_2, \cdots, x_n) \tag{9}$$

2.2 模型实验分析

实验采用三角形分类程序（TRI）中的直角三角形判定程序进行仿真测试，TRI 程序结构简单、逻辑清晰并且输出参数少（3 个），只需要少量的参数组合就能覆盖某些特定分支。因此 TRI 常被采用作为测试数据自动生成的基准程序。

对直角三角形判定程序进行插桩，并使用分支函数叠加的方式构造适应度函数，可得如下 C 语言程序：

```
int TRI(int x, int y, int z, int lb, int ub) {
  int f1 =10, f2=20, f3=30,f4=40,f5=50,F;
  f1 = max((lb -x), (x- ub));
  if((x > lb) && (x < ub)) {
    f2 = max((lb -y), (y- ub));
    if((y > lb) && (y < ub)) {
      f3 = max((lb -z), (z- ub));
      if((z > lb) && (z < ub)) {
        f4 = max((z-x-y), (x-z-y), (y-z-x));
        if((x + y) > z && (y+z) > x && (x + z) > y) {
          if(max(x,y,z) ==x)
          f5= abs(y*y+z*z-x*x);
          else if(max(x,y,z) ==y)
            f5= abs(z*z+x*x-y*y);
            else if(max(x,y,z) ==z)
                f5= abs(x*x +y*y-z*z);
            }}}}
  f1 = f1 < 0 ? 0 : f1+1;  f2 = f2 < 0 ? 0 : f2+1;
  f3 = f3 < 0 ? 0 : f3+1;  f4 = f4 < 0 ? 0 : f4+1;
  f5 = f5 < 0 ? 0 : f5+1;  F = f1 + f2 + f3 + f4 + f5;
return F;}
```

需要生成的测试数据为 3 个指定范围内能组成直角三角形的数字 (x, y, z)，即 3 个数字需要满足勾股定理。由上面插桩后的程序可以看出，适应度函数 F 越小时，表示个体适应度值越大，实际执行路径越接近直角三角形判定的目标路径，当 F=0 时则完全覆盖目标路径，因此目标优化问题转化为找

3 个数字使得 F=0。

3 实验设计及结果分析

SSA 参数的上界 ub 设置为 [50, 50, 50]，下界 lb 设置为 [1, 1, 1]，参数边界构成了边长为 50 的立方体空间，实验中只考虑边长为整数的情况。由计算可知在设定范围内的（x, y, z）共有 125 000 种组合，满足条件的组合有 120 种，占比约为千分之一，如随机生成测试数据则找到满足条件的数据概率较低。ISSA 的 try_number 设置为 5，其他参数与 SSA 参数一致，两种算法最大迭代次数均设置为 100，种群数量设置为 30，分别运行 10 次，对寻优结果进行对比分析。

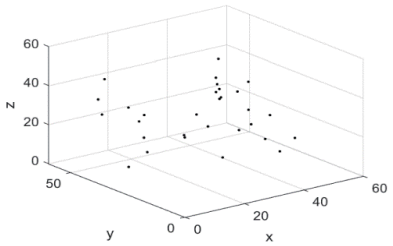
表 2 为 SSA 执行 10 次，(x, y, z) 为每次实验执行完成后得到的测试数据，即三角形 3 条边长，最佳适应度值为本次实验最后一次迭代结束后领导者的适应度值，迭代次数为本次实验算法运行结束时迭代总次数。

表 2 SSA 执行 10 次结果

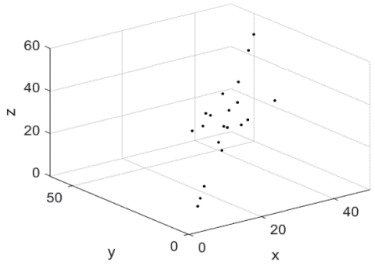
实验次数	(x, y, z)	最佳适应度值	迭代次数
1	(32,24,40)	0	9
2	(23,16,28)	9	100
3	(30,34,16)	0	3
4	(37,17,33)	33	100
5	(16,12,20)	0	3
6	(20,15,25)	0	7
7	(29,20,21)	9	100
8	(16,34,30)	0	2
9	(14,31,34)	9	100
10	(28,35,21)	0	4

从表 2 可以看出，10 次执行中有 6 次成功找到了符合条件的测试数据，成功率为 60%，另外 4 次在达到最大迭代次数时最佳适应度值仍未达到 0。

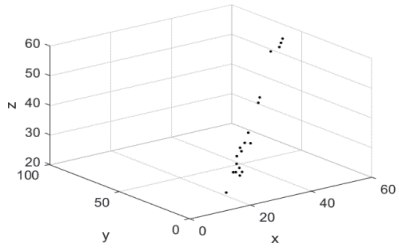
图 2 为其中一次到达最大迭代次数时仍未找到满足条件的测试数据的种群个体分布情况，（a）、（b）、（c）、（d）分别为初始随机生成的、迭代 3 次、迭代 10 次、迭代 20 次的种群个体分布。由图 3（d）可以看出，由于个体陷入局部最优使得前后关联个体被动陷入局部最优，最终导致整个樽海鞘链陷入停滞，到达最大迭代次数后最佳适应度值仍未改善。



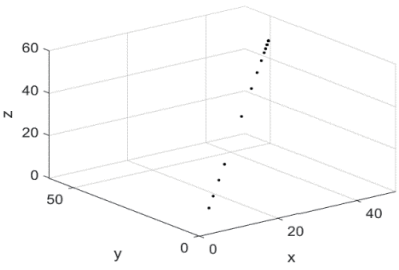
（a）初始随机生成



（b）迭代 3 次



（c）迭代 10 次



（d）迭代 20 次

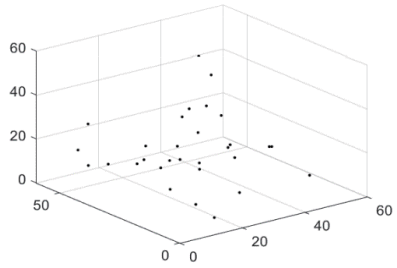
图 2 SSA 运行中不同迭代次数的种群个体分布情况

表 3 为 ISSA 执行 10 次的结果，均在设定最大迭代次数内找到满足测试条件的数据，成功率为 100%，平均迭代次数为 29.4 次。

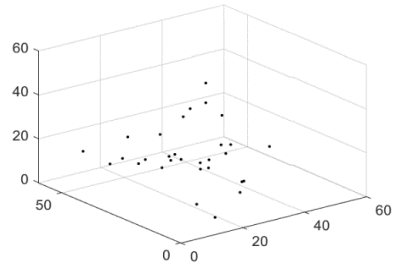
表 3 ISSA 执行 10 次结果

实验次数	(x, y, z)	最佳适应度值	迭代次数
1	(20,29,21)	0	36
2	(5,13,12)	0	21
3	(26,24,10)	0	33
4	(16,30,34)	0	38
5	(15,25,20)	0	32
6	(8,17,15)	0	32
7	(29,20,21)	0	32
8	(35,37,12)	0	33
9	(29,21,20)	0	3
10	(12,15,9)	0	34

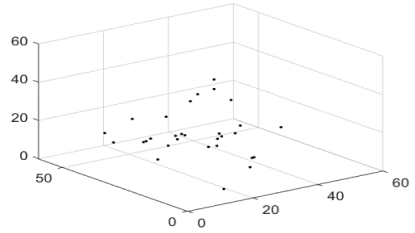
图 3 为其中一次迭代过程中种群个体分布情况，（a）、（b）、（c）、（d）分别为初始随机生成的、迭代 3 次、迭代 10 次、迭代结束时的种群个体分布情况，可见由于随机行为和“末位淘汰”机制的加入。随着迭代的进行，群体未出现持续陷入局部最优的情况，并且都能在较少的迭代次数内找到满足测试条件的数据。相比 SSA 具有更高的可靠性。



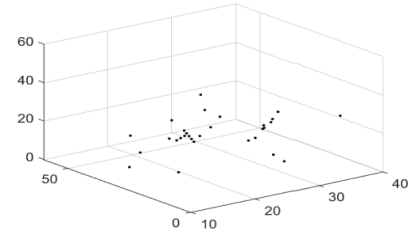
（a）初始随机生成



（b）迭代 3 次



（c）迭代 10 次



（d）迭代结束

图 3 ISSA 运行中不同迭代次数的种群个体分布情况

4 结语

通过对 SSA 引入人工鱼群算法的随机行为和“末位淘汰”机制，对算法进行改进并应用于直角三角形判定程序测试数据自动生成问题中。实验结果表明，改进后的 SSA 改善了改进前出现的种群容易陷入局部最优的问题，在较少的迭代次数下能有效得到满足基准程序测试条件的测试数据，具有更高的可靠性。

参考文献：

[1] 陈琳玲 . 基于简化粒子群算法的测试数据自动生成方法研究 [D]. 重庆 : 西南大学 , 2010.

[2] XANTHAKIS S，ELLIS C，SKOURLAS C，et al．Application of genetic algorithms to softw are testing[EB/OL]. (2009-11-01)[2023-07-06].https://www.researchgate.net/publication/228847410_Application_of_Genetic_Algorithm_in_Software_Testing.

[3] 苗晓旭 , 胡玉露 , 徐豪 , 等 . 基于改进的蚁群算法的测试数据自动生成方法 [J]. 电子技术与软件工程 ,2019(13):180-181.

[4] CHHABRA J K,KUMAR S,DAHIYA S. Automated test data generation using swarm intelligence approaches[J]. The institution of engineers(India) electronics and telecommunication engineering journal，2010,90(18):3-13.

[5] 王培崇 , 钱旭 . 基于改进鱼群算法的路径测试数据生成 [J]. 计算机应用 ,2013,33(4):1139-1141.

[6] 廖伟志 , 夏小云 , 贾小军 . 基于蚁群算法的多路径覆盖测试数据生成 [J]. 电子学报 ,2020,48(7):1330-1342.

[7] 佘嘉纯 , 刘涛 , 梅偌玮 , 等 . 基于改进的人工鱼群算法的软件测试数据自动生成算法 [J]. 成都信息工程大学学报 , 2019,34(6):595-599.

[8] SEYEDALI M, AMIR H G, SEYEDEH Z M, et al. Salp swarm algorithm: a bio-inspired optimizer for engineering design problems[J]. Advances in engineering software, 2017, 114(1): 163-191.

[9] 陈涛 , 王梦馨 , 黄湘松 . 基于樽海鞘群算法的无源时差定位 [J]. 电子与信息学报 ,2018,40(7):1591-1597.

[10] 李玲玲 , 陈文泉 , 冯欢 . 基于改进的樽海鞘群算法的电力负荷经济调度策略 [J]. 天津工业大学学报 ,2022,41(2):67-73.

[11] 高雪笛 , 周丽娟 , 张树东 , 等 . 基于改进遗传算法的测试数据自动生成的研究 [J]. 计算机科学 ,2017,44(3):209-214.

[12] KOREL B. Automated software test data generation[J]. IEEE trans on software engineering,1990,16(8):870-879.

[13] 安新 , 何明祥 . 一种改进粒子群算法的测试数据自动生成方法 [J]. 软件导刊 ,2016,15(9):46-48.

【作者简介】

徐良（1993—），男，河南信阳人，硕士，助教，研究方向：智能优化算法。

田青云（1996—），男，河南信阳人，硕士，助教，研究方向：数据挖掘。

文成（1994—），男，河南信阳人，硕士，助教，研究方向：数据采集。

张海波（1993—），男，河南信阳人，硕士，助教，研究方向：模式识别与人工智能。

郭晶晶（1989—），女，河南信阳人，硕士，讲师，研究方向：智能优化算法。

（收稿日期：2023-10-19）