

DOI: 10.11918/j.issn.0367-6234.201707123

一种基于全局-局部双向驱动的改进果蝇优化算法

王友卫¹, 凤丽洲², 朱建明¹, 柴艳妹¹, 吴越³

(1. 中央财经大学 信息学院, 北京 100081; 2. 天津财经大学 理工学院, 天津 300222;

3. 中央财经大学 保险学院, 北京 100081)

摘要:为解决传统果蝇优化算法过早收敛、结果不稳定等问题,提出一种基于全局-局部双向驱动的果蝇优化新算法。首先,为综合考虑果蝇群体的全局化驱动信息和果蝇个体的局部化驱动信息,引入先进群组 and 记忆空间的概念,即在每次迭代过程中,将果蝇种群中表现较好的若干只果蝇定义为先进群组,将每只果蝇经过的若干历史最优位置定义为该果蝇的记忆空间。然后,为避免过早收敛问题,考虑先进群组中所有个体的全局化驱动作用,通过顺序选择果蝇位置向量的各个维度实现果蝇位置更新。最后,为避免种群接近收敛时盲目地进行全局搜索,每只果蝇个体将考虑自身认知经验的局部化驱动作用,通过使用轮盘赌策略选择记忆空间中特定位置并向其靠近以跳出局部最优。针对典型测试函数及网络异常检测仿真的实验结果表明:基于全局-局部双向驱动的果蝇优化算法收敛精度高、稳定性好、收敛速度快,适用于处理网络异常检测中的高维、复杂的优化问题。

关键词: 果蝇算法; 局部最优; 轮盘赌; 异常检测; 多极值

中图分类号: TP301.6

文献标志码: A

文章编号: 0367-6234(2018)05-0093-09

An improved fruit fly optimization algorithm based on global-local bidirectional driving

WANG Youwei¹, FENG Lizhou², ZHU Jianming¹, CHAI Yanmei¹, WU Yue³

(1. School of information, Central University of Finance and Economics, Beijing 100081, China;

2. School of Science and Engineering, Tianjin University of Finance and Economics, Tianjin 300222, China;

3. School of Insurance, Central University of Finance and Economics, Beijing 100081, China)

Abstract: To solve the problems that the traditional fruit fly algorithms fall into convergence too early and the results are not stable, an improved fruit fly optimization algorithm based on global-local bidirectional driving is proposed. Firstly, in order to comprehensively consider the global driving information of fruit fly population and the local driving information of a fruit fly individual, the conceptions of advanced group and memory space are introduced. In each iteration, the fruit flies which have good performances are defined as the advanced group, and the historical best positions of a fruit fly are defined as the memory space of this fruit fly. Secondly, in order to avoid the premature convergence problem, the global driving effect of the fruit flies in the advanced group is considered, and the dimensional components of the fruit fly position vectors are updated sequentially in the position updating processes. Finally, in order to avoid the blind global searching when the population approaches convergence, each fruit fly will consider the local driving effect of its own cognitive experience, and the roulette strategy is used to select the positions in the memory space for jumping out the local optimum. The experimental results of typical test functions and the web anomaly detection simulation show that, the proposed fruit fly optimization algorithm based on global-local bidirectional driving has high searching accuracy, good stability, and high convergence speed, and is suitable for dealing with the complex problems with high dimensions in web anomaly detection.

Keywords: fruit fly algorithm; local optimum; roulette; anomaly detection; multiple extremes

目前,针对优化问题的研究在科学、工程和金融领域占有重要的地位,引起国内外学者的广泛关注。

群体智能算法凭借其自适应、自组织、协同性、强鲁棒性和良好的分布性并行性等优点,已经成为求解优化问题的主要方法^[1]。人工蜂群(Artificial Bee Colony, ABC)算法模拟蜂群的智能采蜜行为,根据蜜蜂的分工不同实现蜜源信息的共享和交流,继而找到问题的最优解。由于该算法具有结构简单、易于实现、参数较少等特点,因此相对于蚁群算法、遗传算法及粒子群优化算法而言具有一定优势^[2]。然

收稿日期: 2017-07-18

基金项目: 北京市自然科学基金(4174105); 国家自然科学基金重点支持项目(U1509214); 中央财经大学学科建设基金(2016XX01, 2016XX02); 全国统计科研计划重点项目(2017LZ05)

作者简介: 王友卫(1987—), 男, 博士, 讲师;
朱建明(1965—), 男, 教授, 博士生导师

通信作者: 王友卫, ywwang15@126.com.

而,标准 ABC 算法仍面临易陷入局部最优、进化后期收敛速度慢等问题,继而限制了该算法在实际中的应用^[2]。果蝇优化算法(Fruit Fly Optimization Algorithm, FOA)是一种通过模拟果蝇觅食行为获得全局最优解的新型群体智能优化算法^[3-4]。与传统群体智能算法相比,FOA 算法实现简单、执行速度较快、所需参数较少。但是,该算法极易陷入局部最优,导致后期收敛速度变慢,收敛精度降低,尤其是对于高维多极值的复杂优化问题。文献[5]提出动态双子群协同进化的果蝇优化算法(DDSCFOA)。该算法动态地将整个种群划分为先进子群和后进子群,较好地平衡了局部搜索能力和全局搜索能力。文献[6]提出一种基于随机扰动的改进果蝇优化算法(IFOA)。该算法通过计算果蝇适应值大小决定是向最优位置靠近还是进行全局搜索。算法全局寻优能力较强,但不足之处在于全局搜索随机性大,影响了算法的收敛速度及稳定性。文献[7]提出一种迭代步进值自适应调整的果蝇优化算法(FOAMR)。该算法引入果蝇群体速度进化因子和聚集度因子,同时定义自适应调整因子实现搜索距离大小的动态调整,但无法对负值参数进行寻优。文献[8]对 FOA 算法进行改进,使用果蝇位置向量直接计算适应度函数实现针对负值参数的寻优,但不足之处在于寻优结果对于搜索半径依赖较强。文献[9]考虑过去对现在的影响的启发,增加“历史认知”部分的改进策略,通过线性递增的动态变化系数调整在迭代寻优过程中历史位置对本次学习的价值,避免了因单纯聚集行为使得寻优过程迂回曲折导致错过全局最优解的问题。

现有的基于 FOA 的优化算法仍面临以下问题:

1) 在果蝇位置更新过程中只强调当前全局最优位置的驱动作用,忽略了实际最优解出现在其他表现较好的位置附近的可能性,因此丢失了必要的全局化驱动信息,易导致算法陷入局部最优;

2) 文献[5-7]等算法通过混沌搜索、调整搜索半径等方法跳出局部最优,未能有效利用果蝇个体的历史认知经验,导致新位置产生随机性较大、算法收敛速度较慢、结果不稳定等问题。为此,本文对传统果蝇算法做出改进,引入先进群组、记忆空间等相关概念,实现了一种基于全局-局部双向驱动的果蝇优化新算法。

1 传统果蝇优化算法

与粒子群算法、遗传算法类似,传统果蝇算法以随机值作为初始位置,在迭代寻优过程中所有个体都飞向到上一代最优个体,通过在最优个体附近搜

索寻找全局最优解。算法主要执行步骤如下^[3-4]:

输入:果蝇种群 $X = \{x_i\}$ ($1 < i \leq N$, N 为种群中果蝇数量)、搜索范围 $[x_{\min}, x_{\max}]$ 、最大迭代次数 T_{\max} 。

输出:果蝇最优位置 (x_{b1}, x_{b2}) 、果蝇最优气味浓度 $Smell_{gb}$ 。

初始化参数。具体包括:果蝇数量 N 、最大迭代次数 T_{\max} 、果蝇最优气味浓度 φ_{gb} ($\varphi_{gb} = -\infty$) 等;

更新每只果蝇 x_i 的位置 (x_{i1}, x_{i2}) :

$$x_{i1} = x_{b1} + \text{Rand}(), x_{i2} = x_{b2} + \text{Rand}() \quad (1)$$

式(1)中 $\text{Rand}()$ 产生一个 $(0,1)$ 区间内的随机数。

求得 x_i 对应的气味浓度判定值 s_i , 如下式:

$$d_i = \sqrt{x_{i1}^2 + x_{i2}^2}, s_i = \frac{1}{d_i} \quad (2)$$

根据适应度函数 f 求得果蝇 x_i 的气味浓度 φ_i :

$$\varphi_i = f(s_i) \quad (3)$$

将气味浓度最大的果蝇对应的位置及气味浓度值分别记为 (x_{m1}, x_{m2}) 、 φ_m ;

若 $\varphi_m > \varphi_{gb}$, 则按照公式(4)更新果蝇最优位置 (x_{b1}, x_{b2}) :

$$x_{b1} = x_{m1}, x_{b2} = x_{m2} \quad (4)$$

循环执行上述步骤直到达到最大迭代次数。

2 基于全局-局部双向驱动的果蝇优化算法

2.1 算法描述

为提高搜索性能,FOABHC 算法^[9]融合了当前最优位置及果蝇历史位置对位置更新的驱动作用。但是,该算法只关注全局最优位置,忽略了表现较好的那些优秀果蝇的引导作用。并且,该算法只将果蝇当前位置作为历史位置,忽略了那些可能导致全局最优解的历史位置信息见图 1,假定 x_{cgb1} 为当前最优位置, x_{gb} 为实际最优位置, x_{cgb2} 、 x_{cgb3} 、 x_{cgb4} 为 3 个表现与 x_{cgb1} 相近但稍逊于 x_{cgb1} 的果蝇位置。由于现有算法仅考虑当前最优值的导向作用,因此果蝇 x 将直接向 x_{cgb1} 靠近。但实际上, x_{cgb2} (x_{cgb3}) 离实际最优位置 x_{gb} 更近,因此,在位置 x_{cgb2} (x_{cgb3}) 附近搜索将更有可能找到全局最优值 x_{gb} 。如图 2 所示,假定果蝇 x 的移动轨迹为: $x_{h1} \rightarrow x_{h2} \rightarrow x_{h3} \rightarrow x_{h4} \rightarrow x_{h5}$, x_{h5} 为 x 的当前位置, x_{cgb} 为当前最优位置, x_{gb} 为实际最优位置。由于 FOABHC 在果蝇位置更新过程中仅考虑当前最优位置及果蝇当前位置,因此若 x 在位置 x_{h3} 处错过实际最优位置 x_{gb} 而转向 x_{h4} 的话,其后续位置 x_{h5} 便由 x_{h4} 及当前最优位置 x_{cgb} 共同决定。由图 2 知, x 将由 x_{h4} 转向一个靠近 x_{cgb} 而远离 x_{gb} 的位置 x_{h5} , 因此 x 将极有可能陷入局部最优状态。

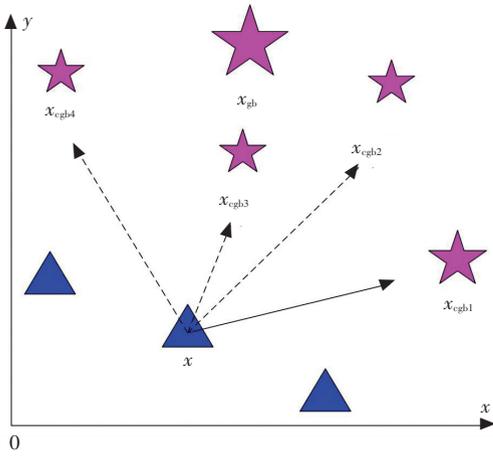


图1 现有优化算法面临的问题

Fig.1 Problems of existing optimization algorithms

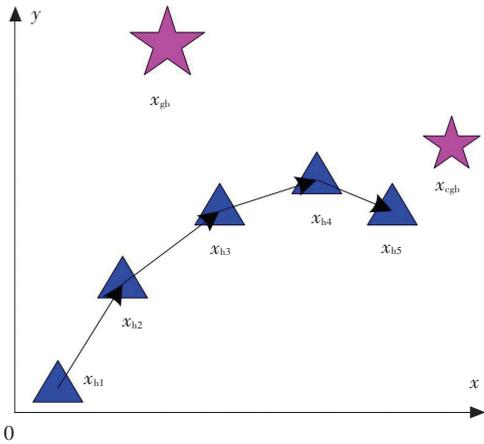


图2 FOABHC算法面临的问题

Fig.2 Problems of FOABHC algorithm

由此可见,FOABHC 算法虽然融合了优秀果蝇个体及果蝇自身认知经验在果蝇位置更新的驱动作用,但缺乏对基于多个最优位置的全局化信息及基于个体自身认知经验的局部化信息的综合运用. 为此,本文引入两个概念:先进群组 and 记忆空间,并以最小化问题为例给出相关定义如下:

定义1 先进群组

给定果蝇种群 $X = \{x_i\} (1 \leq i \leq N)$, 将第 $t (t \leq T_{max})$ 次迭代情况下适应值最小的前 n 只果蝇集合称为该迭代次数下的先进群组, 记为

$$G_t = \{x_g^1, x_g^2, \dots, x_g^u, x_g^v, \dots, x_g^n\}. \quad (5)$$

式中 $1 \leq u \leq v \leq n, f(x_g^u) \leq f(x_g^v)$.

定义2 记忆空间

给定果蝇个体 x_i , 假定 x_i 在第 t 迭代前所经历的历史位置为 $hp_i = \{hp_i^1, hp_i^2, \dots, hp_i^t\}$, 则将 hp_i 中适应值最小的前 $m (m \leq t)$ 个位置称为 x_i 在第 t 迭代时对应的记忆空间, 记为

$$M_i = \{h_i^1, h_i^2, \dots, h_i^{u'}, h_i^{v'}, \dots, h_i^m\}. \quad (6)$$

式中 $1 \leq u' < v' \leq m, f(h_i^{u'}) \leq f(h_i^{v'})$.

在此基础上,将给出基于全局-局部双向驱动的果蝇位置更新过程,具体包括:

1) 基于全局化信息驱动的果蝇位置更新:

如图3所示,为充分利用先进群组中的全局化驱动信息,针对果蝇 x_i , 本文将结合轮盘赌策略^[10]从先进群组 G_t 中选择某只果蝇 x_g 并向其靠近. 此外,IFFO算法^[8]通过随机修改位置向量的某个维度实现果蝇位置的更新,虽然有效降低了算法时间复杂度,但所选维度随机性较大,易造成不稳定的收敛结果. 为此,本文将在迭代过程中顺序选择位置向量的各个维度进行更新,以保证每个维度都能被公平选择. 具体而言,选择 x_g 的过程如下:

步骤1 按照式(7)计算 x_i 选择先进群组 G_t 中每只果蝇 x_g^j 的概率 p_j :

$$p_j = 1 - f_g^j / \sum_{k=1}^n f_g^k. \quad (7)$$

其中 f_g^j 为 x_g^j 对应的适应值.

步骤2 根据文献[11]知,随机数产生函数 Chebyshev 表现优于 Sine、Logistic、Circle 等函数,因此将使用该函数(简记为 cbs())产生随机数 r . 根据轮盘赌策略,按照式(8)选择一只果蝇个体 x_s :

$$x_s = \begin{cases} x_g^1, & \text{if } r \in (0, p_1]; \\ x_g^{j+1}, & \text{if } r \in (\sum_{k=1}^j p_k, \sum_{k=1}^{j+1} p_k] (1 \leq j \leq n-1). \end{cases} \quad (8)$$

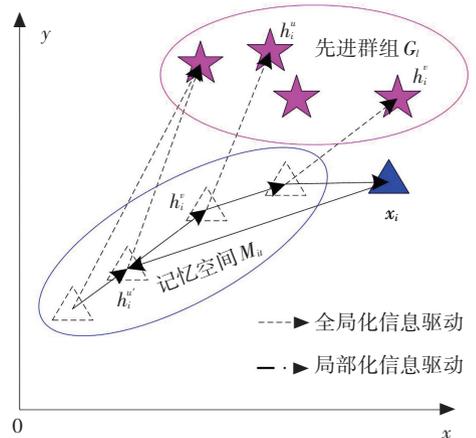


图3 基于全局-局部双向驱动的果蝇位置更新过程

Fig.3 Global-local bidirectional driving based position updating process of fruit flies

2) 基于局部化信息驱动的果蝇位置更新:

为避免陷入局部最优,本文将在果蝇种群接近收敛时执行全局化搜索. 与文献[5-7]等算法不同的是,本文将在全局化搜索过程中结合果蝇个体的历史认知经验降低盲目搜索的可能性,在保证全局优化效果的同时加快算法的收敛速度见图3,当全

局最优位置 x_b 在连续 L 次迭代后仍未改变时,果蝇个体 x_i 将从其记忆空间 M_i 中选择一个特定位置 h_i^s 并在该位置附近搜索以跳出局部最优状态. 具体而言,选择 h_i^s 的过程如下:

步骤1 式(9)计算 x_i 选择记忆空间 M_i 中位置 h_i^j 的概率 p_i^j :

$$p_i^j = 1 - f_i^j / \sum_{k=1}^m f_i^k. \quad (9)$$

式中 f_i^j 为果蝇 x_i 历史位置 h_i^j 对应的适应值.

步骤2 令 $r = \text{cbs}()$, 根据轮盘赌策略,按照公式(10)选择一个最优位置 h_i^s :

$$h_i^s = \begin{cases} h_i^1, & \text{if } r \in (0, p_i^1]; \\ h_i^{j+1}, & \text{if } r \in \left(\sum_{k=1}^j p_i^k, \sum_{k=1}^{j+1} p_i^k \right] \quad (1 \leq j \leq m-1). \end{cases} \quad (10)$$

在此基础上,以最小化问题为例,给出本文算法的执行过程如下:

输入 果蝇种群 X (数量为 N), 最大迭代次数 T_{\max} , 当前迭代次数 t , 位置向量维数 D , 搜索半径 r 最大值 r_{\max} , 搜索半径 r 最小值 r_{\min} , 搜索范围 $[x_{\min}, x_{\max}]$, 迭代次数阈值 L .

输出 全局最优果蝇 x_b .

1) for ($i \leftarrow 1; x_i \in X; i++$)

2) 按照下面公式随机生成果蝇 x_i 的位置:

$$x_{ij} \leftarrow x_{\min} + (x_{\max} - x_{\min}) \times \text{cbs}(), 1 \leq j \leq D. \quad (11)$$

式(11)中 x_{ij} 为果蝇 x_i 的第 j 维数值.

3) 令果蝇 x_i 的记忆空间 $M_i \leftarrow \text{null}$.

4) end for

5) for ($i \leftarrow 1; x_i \in X; i++$)

$$M_i \leftarrow M_i \cup \{x_i\}.$$

6) end for

$$7) x_b \leftarrow \arg(\max_{x_i \in X} f(x_i)). \quad (12)$$

8) 构造先进群组 $G_t = \{x_g^j\} (1 \leq j \leq n)$:

$$X' \leftarrow \text{rank}(X), G_t \leftarrow \text{sel}(X', n). \quad (13)$$

式(13)中 rank 函数实现对果蝇种群 X 按照适应值从小到大排序, $\text{sel}(X', m)$ 表示排序后种群 X' 中的前 m 只果蝇.

9) $t \leftarrow 1, d \leftarrow 0$.

10) while ($t < T_{\max}$)

11) for ($i \leftarrow 1; x_i \in X; i++$)

$$12) j = d\%D + 1. \quad (14)$$

13) if x_b 在 L 次迭代后仍未改变 then

式(10)从 x_i 对应的记忆空间 M_i 中选择位置 h_i^s , 令 $x_s \leftarrow h_i^s$.

按照式(15)更新 x_i 在第 j 维上的值 x_{ij} :

$$x_{ij} = x_{ij} + (x_{bj} - x_{ij}) \times (\text{cbs}() - 0.5) \times 2. \quad (15)$$

14) else

式(8)从 G_t 中选择个体 x_s .

式(16)更新 x_{ij} :

$$x_{ij} = x_{sj} + r \times \text{cbs}(). \quad (16)$$

15) end if

16) if $x_{ij} > x_{\max}$ then $x_{ij} = x_{\max}$ endif

17) if $x_{ij} < x_{\min}$ then $x_{ij} = x_{\min}$ endif

18) $d++$.

19) end for

20) 更新全局最优值 x_b 、先进群组 G_t 及半径 r .

21) end while

本文算法输出步骤13中,迭代次数阈值 L 根据经验取 $L = 500$. 式(16)中,参数 r 为动态搜索半径,随着迭代次数增加, r 将逐渐减小,具体更新过程如下:

$$r = r_{\max} - (r_{\max} - r_{\min}) \times \frac{t}{T_{\max}}. \quad (17)$$

参考文献[8],设置 $r_{\min} = 0.001, r_{\max} = (x_{\max} - x_{\min})/2$. 可见,改进后的果蝇优化算法相对于传统算法(FOA^[3-4], IFOA^[6], IFFO^[8]等)而言具有以下优势:1)引入先进群组的概念,利用种群中表现较好的个体构建先进群组(本文算法输出步骤8);综合考虑先进群组中的优秀个体在果蝇位置更新过程中的驱动作用(本文算法输出步骤14),避免传统算法单纯依赖全局最优位置而可能造成的局部收敛问题;2)引入记忆空间的概念,将果蝇个体的历史最优位置定义为该果蝇的记忆空间(本文算法输出步骤5);每只果蝇在种群接近收敛时利用自身历史认知经验的局部化驱动作用从记忆空间中选择特定位置并向其靠近(本文算法输出步骤13),以此在避免局部最优的同时提高算法的收敛速度.

2.2 算法时间复杂度分析

1) 初始化过程对应本文算法描述中步骤1~9. 其中,构造先进群组过程(本文算法输出步骤8)使用 quicksort 算法^[11]对果蝇种群进行排序,因此其时间复杂度为

$$T_1 = O(D \times N + N \times \log_2 N). \quad (18)$$

2) 果蝇位置更新过程对应本文算法中步骤10~19,对应的时间复杂度为

$$T_2 = O(T_{\max} \times N \times (p \times m + q \times n)). \quad (19)$$

式中: m, n 分别为果蝇记忆空间大小与先进群组中果蝇数量, p, q 分别为本文算法执行步骤13、14的权重系数,满足 $0 < p, q < 1$ 且 $p + q = 1$.

3) 参数更新过程对应本文算法中步骤20. 该过程时间复杂度为

$$T_3 = O(T_{\max} \times N \times \log_2 N). \quad (20)$$

本文算法的时间复杂度为

$$T = T_1 + T_2 + T_3 = O(N \times (D + \log_2 N + T_{\max} \times (p \times m + q \times n + \log_2 N))). \quad (21)$$

由于 m, n 均为固定值,因此去掉式(21)中的常数项,可得

$$T = O(N \times (D + T_{\max} \times \log_2 N)). \quad (22)$$

表1分析了几种典型算法的时间复杂度.其中, N_e 为 FOAMR 算法在计算果蝇聚集度因子时所采用的迭代次数阈值.由表知,IFFO 及 ABC 算法对应的的时间复杂度最小.由公式(22)知,当存在: $\log_2 N < D \times \left(1 - \frac{1}{T_{\max}}\right)$ 时,本文时间复杂度小于除 IFFO、ABC 以外的其他算法.

表1 不同算法的时间复杂度对比

Tab.1 Comparison of time complexities of different algorithms

算法	时间复杂度
FOA/FOABHC/IFOA	$O(N \times D \times T)$
IFFO/ABC	$O(N \times (D + T))$
FOAMR	$O(T \times (N \times D + N_e))$
本文	$O(N \times (D + T \times \log_2 N))$

3 实验结果与分析

3.1 标准函数测试实验

3.1.1 实验设置

实验所采用的平台为: Win7 64 位操作系统, Matlab 2013. 见表2, 分别选取3个单峰标准函数 ($F_1 \sim F_3$)^[6,9,12]、3个多峰标准函数 ($F_4 \sim F_6$)^[9] 进行测试. 实验选用的对比算法包括: FOA^[3-4]、IFOA^[6]、FOAMR^[7]、IFFO^[8]、FOABHC^[9] 及 ABC^[1-2]. 公平起见, 设置果蝇数量 $N = 50$, 先进群组中个体数量 $n = 5$, 记忆空间大小 $m = 5$, 最大迭代次数 $T_{\max} = 5000$.

3.1.2 衡量指标

为衡量优化算法的性能, 将从不同角度测试算法在不同测试函数上的表现. 分别用指标 A、S、C 表示算法在某测试函数上的收敛精度、收敛稳定性及达到收敛状态时的迭代次数. A、S、C 定义为

$$A = \frac{1}{n_e} \sum_{j=1}^{n_e} (f_j - f), \quad S = \sqrt{\frac{1}{n_e} \sum_{j=1}^{n_e} (f_j - f - A)^2},$$

$$C = \frac{1}{n_e} \sum_{j=1}^{n_e} n_j. \quad (23)$$

式中: n_e 为算法针对测试函数进行的实验次数, f 为测试函数对应的理论最优值, $f_j (1 \leq j \leq n_e)$ 为算法针对测试函数进行第 j 次实验所得的实际最优值, n_j

为算法在测试函数上的第 j 次实验达到收敛状态时对应的迭代次数. 为便于比较, 使用指标 A' 衡量算法的收敛精度, 对应公式如下:

$$A' = -\lg A. \quad (24)$$

可见, 指标 A (A') 反映函数最优值理论最优值与实际最优值之间的平均差异, 该值越小 (大), 说明算法收敛精度越高; 指标 S 反映算法收敛结果的稳定性, 该值越小, 说明算法表现越稳定; 指标 C 反映算法的收敛速度, 该值越小, 说明算法达到收敛状态所需迭代次数越少, 算法收敛得越快.

表2 函数公式、最优值及最优位置分量

Tab.2 Function formulas, optimal values and optimal position components

函数名称	公式	最优值	最优位置分量
F_1	$\sum_{i=1}^D x_i + 1 $	0	-1
F_2	$\sum_{i=1}^D (100(x_i - x_i^2)^2 + (x_i - 1)^2)$	0	1
F_3	$\sum_{i=1}^D x_i + \prod_{i=1}^D x_i $	0	0
F_4	$\sum_{i=1}^D (x_i^2 - 10\cos(2\pi x_i) + 10)$	0	0
F_5	$\sum_{i=1}^{D-1} \frac{\sin^2 \sqrt{x_i^2 + x_{i+1}^2} - 0.5}{(1 + 0.001(x_i^2 + x_{i+1}^2))^2} + 0.5$	0	0
F_6	$1 - \cos\left(2\pi\sqrt{\sum_{i=1}^D x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^D x_i^2}$	0	0

3.1.3 收敛精度及稳定性比较

设置每个测试函数的维数 D 分别取 2、4、6、8、10, 针对每种算法在每个函数上实验 100 次, 统计每种算法对应的 A 值及 S 值, 结果见表3、4. 由表知, 当处理单峰函数 F_1 时, FOA、FOABHC、IFOA 及 FOAMR 算法对应 A 值均普遍大于 6, 原因在于它们直接使用味道浓度作为选择全局最优果蝇的判定标准, 因此无法实现针对负值参数的寻优. 当处理 F_2 时, 本文表现最好; 当处理 F_3 时, 本文表现虽稍逊于 ABC 算法, 但对应的 A 值仍低于其他算法. 进一步发现, 本文虽在多峰函数 F_4 上所得结果较 ABC 算法偏高, 但在 F_5 、 F_6 上均获得最小 A 值、 S 值, 这说明: 1) 鉴于多峰函数的复杂性, 传统算法存在易陷入局部最优、全局搜索随机性较大等问题, 因此难以有效协调算法的收敛精度及稳定性; 2) 本文基于全局 - 局部双向驱动的果蝇位置更新策略能有效利用果蝇群体的最优位置信息及果蝇个体的认知经验, 在避免算法局部收敛、提高全局搜索稳定性方面起到一定作用.

3.1.4 收敛速度比较

令每个测试函数的维数 D 分别取 2、4、6、8、10,

针对每种参数优化算法在每个测试函数上实验 100 次,统计每种算法达到收敛状态时对应的 C 均值 (C_a) 及 C_{aa} 均值 (C_{aa}),结果见表 5. 由表知,FOA 收敛最快,对应的 C_{aa} 值为 805,ABC 收敛最慢,对应的 C_{aa} 值为 1518;IFFO 及 FOABHC 表现相近,对应 C_{aa} 值分别为 1 045、1 048,明显小于 IFOA、FOAMR 及 ABC 算法. 但是,FOA、IFFO 及 FOABHC 没有在局

部收敛后进行全局搜索,因此由表 3 结果知,这些算法将面临过早陷入局部极值的风险. 与 IFOA、FOAMR 及 ABC 算法相比,本文对应的 C_a 值及 C_{aa} 值均明显偏小,这主要是因为 IFOA、FOAMR 及 ABC 算法在局部收敛后随机产生果蝇新位置,而本文利用个体历史认知信息能有效降低全局搜索的盲目性、提高算法收敛速度.

表 3 不同算法对应的 A 值比较

Tab.3 Comparison of A values of different algorithms

算法	F_1	F_2	F_3	F_4	F_5	F_6
FOA	6.210	1.25×10^{-2}	8.14×10^{-4}	5.36×10^{-4}	2.64×10^{-1}	6.25×10^{-2}
IFFO	5.34×10^{-2}	2.82×10^{-3}	2.32×10^{-5}	1.05×10^{-5}	9.21×10^{-2}	3.53×10^{-4}
FOABHC	6.362	2.19×10^{-3}	5.21×10^{-5}	2.37×10^{-5}	7.12×10^{-4}	3.90×10^{-3}
IFOA	6.118	7.18×10^{-2}	5.36×10^{-4}	2.85×10^{-5}	6.75×10^{-3}	4.38×10^{-4}
FOAMR	6.326	2.74×10^{-3}	5.57×10^{-5}	3.16×10^{-4}	6.76×10^{-4}	9.34×10^{-4}
ABC	6.45×10^{-6}	2.18×10^{-2}	4.37×10^{-7}	5.42×10^{-7}	5.63×10^{-2}	2.61×10^{-2}
本文	4.32×10^{-4}	4.77×10^{-5}	7.83×10^{-6}	2.54×10^{-6}	1.91×10^{-4}	2.39×10^{-4}

表 4 不同算法对应的 S 值比较

Tab.4 Comparison of S values of different algorithms

算法	F_1	F_2	F_3	F_4	F_5	F_6
FOA	6.49×10^{-2}	3.44×10^{-2}	6.25×10^{-5}	6.14×10^{-5}	1.03×10^{-5}	2.47×10^{-4}
IFFO	2.05×10^{-6}	1.11×10^{-4}	2.46×10^{-5}	6.35×10^{-6}	4.61×10^{-4}	3.18×10^{-6}
FOABHC	3.65×10^{-2}	3.25×10^{-2}	4.21×10^{-6}	5.38×10^{-6}	9.16×10^{-5}	1.66×10^{-5}
IFOA	4.26×10^{-1}	8.32×10^{-1}	9.04×10^{-4}	3.22×10^{-6}	6.52×10^{-5}	4.17×10^{-6}
FOAMR	5.38×10^{-1}	6.74×10^{-3}	4.08×10^{-5}	2.17×10^{-5}	4.33×10^{-5}	5.52×10^{-6}
ABC	2.31×10^{-4}	3.15×10^{-3}	1.15×10^{-6}	1.81×10^{-6}	2.69×10^{-4}	3.38×10^{-4}
本文	8.24×10^{-5}	3.57×10^{-5}	5.69×10^{-5}	3.22×10^{-5}	4.37×10^{-6}	2.32×10^{-6}

表 5 不同算法对应的 C_a 值及 C_{aa} 值比较

Tab. 5 Comparison of C_a and C_{aa} values of different algorithms

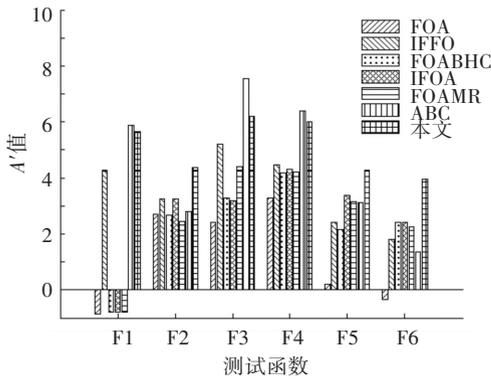
算法	C_a						C_{aa}
	F_1	F_2	F_3	F_4	F_5	F_6	
FOA	821	725	773	905	730	875	805
IFFO	1 087	936	1 056	1 174	1 009	1 008	1 045
FOABHC	1 092	968	1 066	1 139	995	1 029	1 048
IFOA	1 356	1 271	1 316	1 558	1 789	1 615	1 484
FOAMR	1 329	1 474	1 262	1 501	1 579	1 496	1 440
ABC	1 168	1 496	1 274	1 332	2 055	1 786	1 518
本文	1 132	1 224	1 208	1 216	1 458	1 272	1 251

3.1.5 维数变化的影响

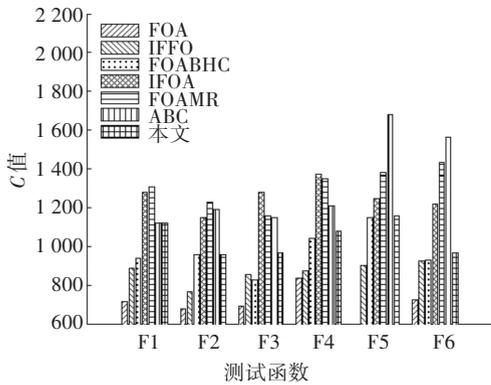
当维数 D 分别取 4、8 时,针对每种参数优化算法在每个测试函数上实验 100 次,统计不同算法对应的 A' 值及 C 值,结果见图 4. 可见,随着维数的增加,所有算法在收敛精度及收敛速度等方面的表现均变差. 当处理单峰函数时,本文在 F_2 上精度明显

高于其他算法,虽然在 F_1 、 F_3 上精度不及 ABC,但对应 A' 值仍明显高于 IFFO、IFOA 等算法. 当处理多峰函数 F_4 时,ABC 算法表现最好;当处理多峰函数 F_5 、 F_6 时,本文所得 A' 值均明显高于其他方法,并且在收敛速度上表现优于 IFOA、FOAMR 及 ABC 算法. 可见,随着维数增加,本文性能虽有所下降,但在收

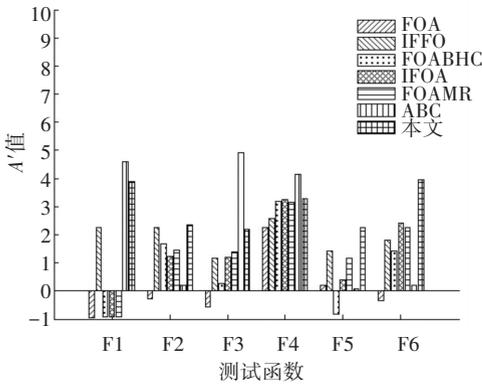
敛精度及收敛速度方面相对于其他算法仍具有一定优势。



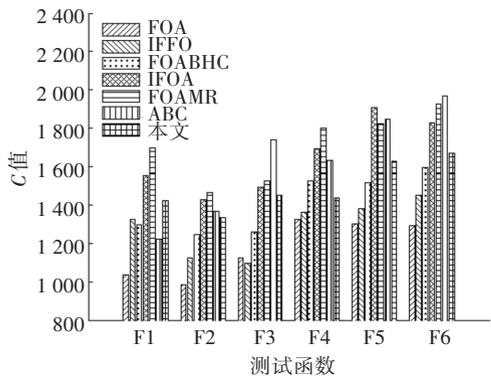
(a) 当 $D = 4$ 时不同算法对应的收敛精度 A'



(b) 当 $D = 4$ 时不同算法对应的收敛迭代次数 C



(c) 当 $D = 8$ 时不同算法对应的收敛精度 A'



(d) 当 $D = 8$ 时不同算法对应的收敛迭代次数 C

图4 维数变化对算法性能的影响

Fig.4 Effect of dimension change on algorithm performances

3.2 网络异常检测仿真实验

基于支持向量机 (Support Vector Machine, SVM) 的网络异常检测技术具有精度高、泛化能力强等优势,因此在网络异常检测中得到广泛的应用^[13]. 将从该领域中涉及的参数优化、特征选择及运行时间比较等方面对不同优化算法进行比较。

3.2.1 异常检测中的参数优化实验

在基于径向基函数 (Radial Basis Function, RBF) 的 SVM 异常检测中, 惩罚系数 c 及 RBF 核函数中的 δ 系数是影响分类性能的重要参数^[14]. 由文献^[14]知, 针对样本向量 x 的类别决策函数为

$$f(x) = \text{sgn} \left(\sum_{i=1}^{n_a} \alpha_i^* y_i \text{RBF}(x, x_i) + b^* \right). \quad (25)$$

式中: $b^* = y_j - \sum_{i=1}^{n_a} \alpha_i^* y_i \text{RBF}(x, x_j), 0 < \alpha_j < c,$

$\text{RBF}(x, x_j) = \exp \left(-\frac{\|x - x_j\|^2}{2\delta^2} \right),$ sgn 为求符号函数,

$\text{RBF}(x, x_i)$ 为核函数, α_i^* 为样本 x_i 对应的拉格朗日乘子, y_i 为样本 x_i 的类别, n_a 为样本数量, c 为惩罚因子, δ 为 RBF 核函数宽度参数. 为验证不同算法的参数优化效果, 将 LibSVM 作为训练和测试工具, 采用 CMFS 特征选择方法^[15] 从 KDDcup99 标准数据集^[16] 41 个特征中分别选择 2、4、6、8、10 个特征用于 SVM 训练和分类, 使用微平均 F 值 (F_{ma}) 为果蝇适应值函数:

$$F_{ma} = \frac{2 \times \rho \times \tau}{\rho + \tau}. \quad (26)$$

式中 ρ, τ 分别为异常检测准确率及召回率^[15]. 可见, F_{ma} 值越大, c, δ 值越准确, 算法优化能力越强. 公平起见, 设置相关参数如下: c, δ 最小值 $x_{\min} = 0.0001,$ c, δ 最大值 $x_{\max} = 5,$ 果蝇数量 $N = 10,$ 最大迭代次数 $T_{\max} = 200.$ 在此基础上, 统计各算法在特征数目 (n_s) 不同情况下达到收敛状态时对应的全局最优适应值 (F_a), 结果见图 5.

由图 5 可知, 随着所选特征数目 n_s 的增加, 使用不同算法优化后的 c, δ 值对应的 F_a 值均呈现递增趋势. FOA 算法表现最差, 且当 $n_s = 2$ 时获得最小 F_a 值 0.926; FOABCH 对应结果明显低于除 FOA 以外的其他算法, 原因在于该算法仅结合全局最优位置及当前位置确定果蝇新位置, 因此所得结果受果蝇初始位置影响较大, 导致算法极易陷入局部最优. 与其他算法相比, 本文在 n_s 分别取 4、6、10 时获得最大值 0.951、0.955、0.964, 可见本文在大多数情况下能获得更合适的 c, δ 值, 说明该算法在获取异常检测分类器重要参数的最佳取值方面起到一定作用。

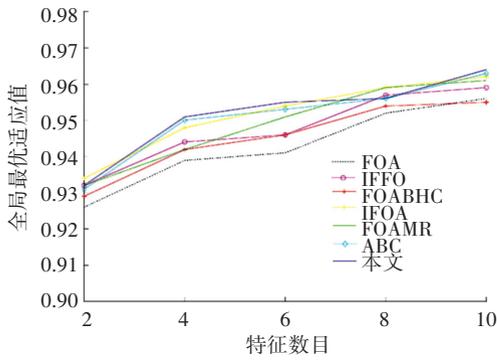


图 5 网络异常检测中不同算法对应的最优适应值比较

Fig.5 Comparison of best fitness values of different algorithms in web anomaly detection

3.2.2 异常检测中的特征选择实验

如何优化特征质量、提高检测速度和精度,是网络异常检测要解决的关键问题^[16]. 将使用不同算法从 KDDcup99 数据集中选择最优特征子集,以测试算法在高维环境中的优化性能. 不同于连续型参数优化算法,在果蝇位置更新过程中,需要对更新后位置 x_{ij} 进行离散化处理,即: 如果 $x_{ij} \in [x_{\min}, \frac{x_{\min} + x_{\max}}{2}]$, 则 $x_{ij} = 0$, 否则, $x_{ij} = 1$. 可见,每只果蝇

位置可表示为 $x_i = \{x_{ij} | (x_{ij} \in \{0, 1\}, 1 \leq j \leq 41)\}$. 其中, $x_{ij} = 1$ 表示第 j 个特征被选择; $x_{ij} = 0$, 表示该特征被丢弃. 进一步地, 设置 $x_{\min} = 0, x_{\max} = 1, r_{\min} = (x_{\max} - x_{\min}) / 2, r_{\max} = x_{\max} - x_{\min}$. 在此基础上, 将使用不同优化算法所得的特征集用于 SVM 分类, 统计 10 次实验后各算法对应的 F_{\max} 值均值 (记为 F_v), 结果见图 6. 可见, 本文对应的 F_v 值 (0.965) 明显高于其他算法, 说明基于全局 - 局部双向驱动的果蝇优化算法能较好地避免局部最优, 相对于其他方法而言更适合解决异常检测中的最优特征选择问题.

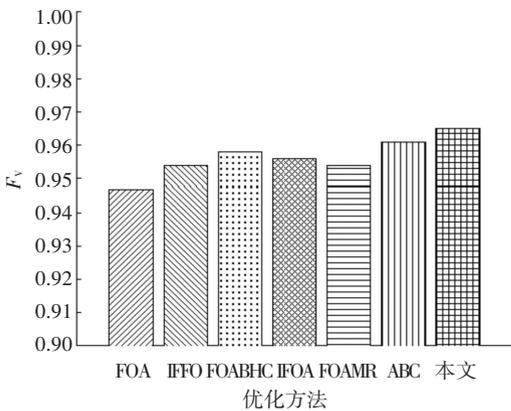


图 6 网络异常检测中不同算法对应的 F_v 值比较

Fig.6 Comparison of F_v values of different algorithms in web anomaly detection

3.2.3 不同优化算法的运行时间比较

为测试算法在不同维数 D 下的执行效率, 分别

统计网络异常检测中参数优化 ($D = 2$) 及特征选择 ($D = 41$) 过程不同算法对应的运行时间 t_R , 结果见表 6. 由表知, 在参数优化实验中, 本文在效率上优势并不明显; 但当处理维数较高的特征选择问题时, 本文及 IFFO 对应的运行时间明显低于 FOA、IFOA 及 ABC 等算法. 可见, 本文通过在迭代中顺序更新特定维度能有效降低维数变化的影响, 使算法能较好地处理高维环境中的复杂优化问题.

表 6 参数优化及特征选择中不同算法的运行时间比较

Tab.6 Comparison of running time of different algorithms in parameter optimization and feature selection

优化算法	参数优化 (t_R / s)	特征选择 (t_R / s)
FOA	8.765×10^3	1.042×10^5
IFFO	8.753×10^3	9.661×10^4
FOABHC	8.772×10^3	1.168×10^5
IFOA	8.776×10^3	1.292×10^5
FOAMR	8.782×10^3	1.431×10^5
ABC	8.774×10^3	1.636×10^5
本文	8.768×10^3	9.934×10^4

4 结 论

提出一种基于全局-局部双向驱动的果蝇优化新算法, 主要贡献如下: 1) 为防止算法早熟收敛, 引入先进群组的概念, 在果蝇位置更新过程中结合先进群组的全局化驱动作用, 使用轮盘赌策略进行果蝇位置更新; 2) 为避免算法在陷入局部最优后盲目地进行全局搜索, 引入记忆空间的概念, 在全局化搜索过程中结合果蝇个体历史认知经验的局部化驱动作用提高种群的收敛速度; 3) 将改进后的果蝇算法应用于基于 SVM 的网络异常检测, 并从参数优化、特征选择及运行时间等方面对几种典型的优化算法进行比较. 针对 6 种典型函数的实验结果表明, 本文在收敛精度、收敛速度、稳定性等方面表现较好, 适合处理高维、多极值的优化问题; 针对网络异常检测的仿真结果表明, 本文能较准确地获得 SVM 重要参数, 并且在特征选择、运行速度等方面相对于传统算法具有一定优势.

参考文献

[1] 徐晓飞, 刘志中, 王忠杰, 等. S-ABC--面向服务领域的人工蜂群算法范型 [J]. 计算机学报, 2015(11):2301-2317. DOI: 10.11897/SP.J.1016.2015.02301.
 XU Xiaofei, LIU Zhizhong, WANG Zhongjie, et al. S-ABC-service domain-oriented artificial bee colony algorithm paradigm [J]. Chinese Journal of Computers, 2015(11):2301-2317. DOI: 10.11897/SP.J.1016.2015.02301.
 [2] 高卫峰, 刘三阳, 黄玲玲. 受启发的人工蜂群算法在全局优化

- 问题中的应用[J]. 电子学报, 2012, 40(12):2396-2403. DOI: 10.3969/j.issn.0372-2112.2012.12.007.
- GAO Weifeng, LIU Sanyang, HUANG Lingling. Inspired artificial bee colony algorithm for global optimization problems [J]. Acta Electronica Sinica, 2012, 40(12):2396-2403. DOI: 10.3969/j.issn.0372-2112.2012.12.007.
- [3] 潘文超. 应用果蝇优化算法优化广义回归神经网络进行企业经营绩效评估[J]. 太原理工大学学报(社会科学版), 2011, 29(4): 1-5. DOI: 10.3969/j.issn.1009-5837.2011.04.002.
- PAN Wenchao. Using fruit fly optimization algorithm optimized general regression neural network to construct the operating performance of enterprises model [J]. Journal of Taiyuan University of Technology (Social Sciences Edition), 2011, 29(4): 1-5. DOI: 10.3969/j.issn.1009-5837.2011.04.002.
- [4] PAN W T. A new fruit fly optimization algorithm: taking the financial distress model as an example [J]. Knowledge-Based Systems, 2012, 26: 69-74. DOI: 10.1016/j.knosys.2011.07.001.
- [5] 韩俊英, 刘成忠, 王联国. 动态双子群协同进化果蝇优化算法[J]. 模式识别与人工智能, 2013, 26(11):1057-1067. DOI: 10.3969/j.issn.1003-6059.2013.11.009.
- HAN Junying, LIU Chengzhong, WANG Lianguo. Dynamic double subgroups cooperative fruit fly optimization algorithm [J]. Pattern Recognition & Artificial Intelligence, 2013, 26(11):1057-1067. DOI: 10.3969/j.issn.1003-6059.2013.11.009.
- [6] WANG Lin, SHI Yuanlong, LIU Shan. An improved fruit fly optimization algorithm and its application to joint replenishment problems [J]. Expert Systems with Applications, 2015, 42(9): 4310-4323. DOI: 10.1016/j.eswa.2015.01.048.
- [7] 常鹏, 李树荣, 葛玉磊, 等. 迭代步进值自适应调整的果蝇优化算法[J]. 计算机工程与应用, 2016, 52(3):32-36. DOI: 10.3778/j.issn.1002-8331.1405-0147.
- CHANG Peng, LI Shurong, GE Yulei, et al. Fruit fly optimization algorithm with self-adapting adjustment of iteration step value [J]. Computer Engineering & Applications, 2016, 52(3):32-36. DOI: 10.3778/j.issn.1002-8331.1405-0147.
- [8] PAN Q K, SANG H Y, DUAN J H, et al. An improved fruit fly optimization algorithm for continuous function optimization problems [J]. Knowledge-Based Systems, 2014, 62: 69-83. DOI: 10.1016/j.knosys.2014.02.021.
- [9] 韩俊英, 刘成忠. 基于历史认知的果蝇优化算法[J]. 计算机科学与探索, 2014, 8(3):368-375. DOI: 10.3778/j.issn.1673-9418.1311028.
- HAN Junying, LIU Chengzhong. Fruit fly optimization algorithm based on history cognition [J]. Journal of Frontiers of Computer Science & Technology, 2014, 8(3):368-375. DOI: 10.3778/j.issn.1673-9418.1311028.
- [10] KUMAR R, JYOTISHREE. Blending roulette wheel selection & rank selection in genetic algorithms [J]. International Journal of Machine Learning and Computing, 2012, 2(4): 365-370. DOI: 10.7763/IJMLC.2012.V2.146.
- [11] MITIC M, VUKOVIC N, PETROVIC M, et al. Chaotic fruit fly optimization algorithm [J]. Knowledge-Based Systems, 2015, 89(C):446-458. DOI: 10.1016/j.knosys.2015.08.010.
- [12] SEDGEWICK R. Implementing quicksort programs [J]. Communications of the ACM, 1978, 21(10):847-857.
- [13] 刘敬, 谷利泽, 钮心忻, 等. 基于单分类支持向量机和主动学习的网络异常检测研究[J]. 通信学报, 2015, 36(11):136-146. DOI: 10.11959/j.issn.1000-436x.2015252.
- LIU Jing, GU Lize, NIU Xinxin, et al. Research on network anomaly detection based on one-class SVM and active learning [J]. Journal on Communications, 2015, 36(11):136-146. DOI: 10.11959/j.issn.1000-436x.2015252.
- [14] CHANG C C, LIN C J. LIBSVM: a library for support vector machines [J]. ACM Transactions on Intelligent Systems & Technology, 2007, 2(3):1-27.
- [15] YANGJ M, LIU Y N, ZHU X D, et al. A new feature selection based on comprehensive measurement both in inter-category and intra-category for text classification [J]. Inform. Process. Manage. 2012, 48(4): 741-754. DOI: 10.1016/j.ipm.2011.12.005.
- [16] 武小年, 彭小金, 杨宇洋, 等. 入侵检测中基于SVM的两级特征选择方法[J]. 通信学报, 2015, 36(4):19-26. DOI: 10.11959/j.issn.1000-436x.2015127.
- WU Xiaonian, PENG Xiaojin, YANG Yuyang, et al. Two-level feature selection method based on SVM for intrusion detection [J]. Journal on Communications, 2015, 36(4): 19-26. DOI: 10.11959/j.issn.1000-436x.2015127.

(编辑 苗秀芝)