基于改进果蝇算法求解混合整数非线性规划问题

朱志同[1] 赵 阳[2] 李 炜[1,2] 郭 星[1,2]

[1](安徽大学计算智能与信号处理重点实验室 安徽 合肥 230039)

[2](安徽大学计算机科学与技术学院 安徽 合肥 230601)

摘 要 在科学及工程系统设计中存在许多混合整数非线性规划 MINLP(Mixed-Integer NonLinear Programming)问题,该类问题变量类型丰富且约束条件较多,难以求解,为此提出一种改进果蝇算法。该算法对不同类型变量的更新采取不同的策略,并采用周期性的步长函数指导果蝇的寻优,使其避免陷入局部最优。并通过与另外两种常用的算法在稳定性、收敛速度等方面进行了比较,实验结果表明该改进的果蝇算法效果较优,能有效地解决 MINLP 问题。

关键词 混合整数非线性规划 智能计算 果蝇算法

中图分类号 TP301.6 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2017.06.047

SOLVING MIXED INTEGER NONLINEAR PROGRAMMING BASED ON THE IMPROVED FRUIT FLIES ALGORITHM

Zhu Zhitong[1] Zhao Yang[2] Li Wei[1,2] Guo Xing[1,2]

[1](*Key Laboratory of ICSP, Ministry of Education, Anhui University, Hefei 230039, Anhui, China*)

[2](*School of Computer Science and Technology, Anhui University, Hefei 230601, Anhui, China*)

Abstract There are many MINLP problems in the design of science and engineering systems, which are rich in variables and have many constraints and are difficult to solve. Therefore, this paper proposes an improved fruit flies algorithm. The algorithm uses different strategies to update different types of variables, and uses the periodic step function to guide the optimization of FOA so as to avoid falling into local optimization. Compared with the other two commonly used algorithms in terms of stability, convergence speed and so on, experimental results show that the improved fruit flies algorithm can effectively solve the MINLP problems.

Keywords Mixed integer nonlinear programming Smart computing Fruit flies algorithm

0 引 言

在科学及工程系统设计中存在许多问题都是非线性规划问题,其中混合整数非线性规划 MINLP 又是非线性规划的一个重要分支[1]。混合整数非线性规划是同时包含离散性变量和连续性变量的优化问题,许多组合优化问题如:TSP 问题、0 - 1 背包问题等都可视为 MNLP 问题。对于 MNLP 问题,其目标函数和约束条件的非线性,使得结果往往存在局部最优解[2]。

解决 MINLP 主要有确定性方法和随机性方法[3]。确定性方法如分枝定界法(B&B)、外逼近法(OA)、广义 Bender 分解法(CBD),但这些方法局限性很大:一是计算量太大;二是不利于编程求解。如分枝定界法取消或放宽原问题的约束,并将改变后的问题分解成多个子问题进行求解,以此确定原问题的最优解或其最优解所在的区间,由此可以看出该方法不适宜用编程的方法去求解。近年来,越来越多的随机性方法应用于 MINLP,如遗传算法(GA)、差分进化算法(DE)、粒子群算法(PSO)等[4-5]。但这些方法也各有缺点,例如:遗传算法易于早熟且计算量大;差分进化算法过于复杂且计算耗时多;粒子群算法容易陷入局部

收稿日期:2016 - 06 - 24。国家科技支撑计划项目(2015BAK24B00)。**朱志同**,硕士生,主研领域:智能计算,信号处理。**赵阳**,本科生。**李炜**,教授。**郭星**,博士。

最优[6]。

潘文超[7]于 2011 年提出了果蝇优化算法 FOA (Fruit Fly Optimization Algorithm)。该算法也属于群智能算法的一种,作为新型的进化算法,与其他算法相比,FOA 在先天上就有很多优势,易于理解且实现简单,全局寻优能力强且收敛速度快,但缺点在于稳定性不足,易于陷入局部最优[8]。随着研究的深入,很多学者对果蝇算法进行了改进,在保持其优势的情况下,尽量提高其稳定性及跳出局部最优的能力。目前已有研究,大多是求解连续性问题,没有应用于解决具有离散变量的实际问题[9]。本文通过改进果蝇算法,使之能有效地解决混合整数非线性规划这一实际问题[10-11]。

1　果蝇算法

1.1　基本果蝇算法

果蝇算法是基于动物群体觅食行为演化出的一种全局寻优新方法。果蝇在视觉和嗅觉上优于其他物种,其发现食物行为的特点包括:① 果蝇个体使用嗅觉器官分辨食物的气味来源并朝这个方向飞行;② 果蝇在飞行过程中使用视觉去寻找食物[12]。

其具体过程如下:

① 初始化参数,果蝇种群数量 ($mSize$),迭代次数 ($pSize$),飞行范围也即自变量的定义域(dom)。

② 随机初始化果蝇群位置 (x_{is},y_{is}):

$$x_{is} = rand(dom) \tag{1}$$

$$y_{is} = rand(dom) \tag{2}$$

③ 果蝇个体先使用嗅觉分别食物气味的方向并立即飞向该方向,再使用视觉发现该食物,$RandomValue$ 为[-1,1]上的随机值,其值的正负表示果蝇飞行的方向,大小表示果蝇飞行的距离:

$$x_i = x_{is} + RandomValue \tag{3}$$

$$y_i = y_{is} + RandomValue \tag{4}$$

④ 计算每只果蝇当前位置与原点的距离 ($dist_i$) 及食物的气味浓度值(s_i):

$$dist_i = \sqrt{x_i^2 + y_i^2} \tag{5}$$

$$s_i = \frac{1}{dist_i} \tag{6}$$

⑤ 将 s_i 代入到目标函数中,求出该位置上食物的气味浓度值($smell_i$):

$$smell_i = function(s_i) \tag{7}$$

⑥ 在果蝇群中找出气味浓度值最大的果蝇个体:

$$[bestsmell\ bestindex] = max(smell_i) \tag{8}$$

⑦ 保留最大气味浓度值,所有果蝇向该位置聚集,并将该位置更新为果蝇群的新位置:

$$Bestsmell = bestsmell \tag{9}$$

$$x_{is} = x[bestindex] \tag{10}$$

$$y_{is} = y[bestindex] \tag{11}$$

⑧ 判断算法是否达到结束标志,若是则结束,否则重复执行步骤③ - 步骤⑥,判断当前果蝇群中最优的气味浓度值是否优于历史最优气味浓度值,若是则执行步骤⑦。

1.2　改进的果蝇算法

由 1.1 节基本果蝇算法的流程可以看出,该算法在果蝇个体的更新方式及计算上存在不足,具体如下:

1) 式(3)、式(4)在连续域上采用随机数方式更新果蝇个体位置,使得果蝇寻优的效率较为低下;

2) 在连续域上取值不能适用于离散域,适用范围较窄;

3) 式(5)、式(6)是计算当前位置与原点间的距离,不能适用于最优点不是原点的情况。

为此,我们要用其解决混合整数非线性规划问题,必须对其进行改造。针对基本果蝇算法的不足,改进主要有以下三个方面:

1) 对不同类型变量采取不同的更新方式,如式(14)、式(15),解决了变量类型与更新方式不匹配的问题。

2) 采取步长函数指导果蝇位置的更新,如式(16)、式(17),使果蝇在寻找最优解的过程不再具有随机性,可以加快收敛,参数 α 为连续型变量的指导参数,取值范围为 0.5 ~ 0.95,β 为离散型变量的参数,取值范围为 0.2 ~ 0.5。并且该函数具有周期性,可以提高果蝇跳出局部最优解的能力,更好地找出全局最优解,参数 T 为其周期,取值范围为 20 ~ 40,T 值越小,函数震荡越强,T 值越大,函数精度越高。p 为当前的迭代次数。

3) 去除式(5)和式(6),将变量直接带入所求函数中,如式(18),简化了计算。

下面是改进的果蝇算法的详细步骤:

① 初始化参数,果蝇种群数量 ($mSize$),迭代次数 ($pSize$),步长函数的参数 α、β、p、T,果蝇在变量 j 的飞行范围($dom_j \in [dom_{L,j}, dom_{u,j}]$)。

② 随机生成果蝇群的初始位置,其中,$x_{axi,j}$ 为连续型变量,$y_{axi,j}$ 为离散型变量:

$$x_{axi,j} = rand(dom_j) \tag{12}$$

$$y_{axi,j} = \lfloor rand(dom_j) \rfloor \tag{13}$$

③ 果蝇个体中各种变量的更新规则,即其使用嗅觉及视觉发现食物,如 $x_{i,j}$ 代表第 i 只果蝇中的第 j 个变

量，$randomvalue \in [-1,1]$：

$$x_{i,j} = x_{axi,j} + \alpha_j \times randomvalue \tag{14}$$

$$y_{i,j} = \lfloor y_{axi,j} + \beta_j \times randomvalue \rfloor \tag{15}$$

$$\alpha_j = | dom_{U,j} - dom_{L,j} | \times \alpha^{p\%T} \tag{16}$$

$$\beta_j = | dom_{U,j} - dom_{L,j} | \times \beta^{p\%T} \tag{17}$$

④ 将 $x_{i,j}$、$y_{i,j}$ 代入目标函数中，求出该果蝇此时位置上的气味浓度（$smell_i$）：

$$smell_i = Function(x_{i,j}, y_{i,j}) \tag{18}$$

⑤ 找出果蝇群中气味浓度最优的果蝇个体：

$$[bestsmell\ bestindex] = max(smell_i) \tag{19}$$

⑥ 保留最优气味浓度值及该果蝇的位置，所有果蝇都飞向该位置：

$$bestSmell = bsetsmell \tag{20}$$

$$x_{axi,j} = x[bestindex] \tag{21}$$

$$y_{axi,j} = y[bestindex] \tag{22}$$

⑦ 判断算法是否到达结束标志，若是则结束算法，否则重复步骤③－步骤⑤，并判断当前果蝇群中最优气味浓度是否优于历史值，若是则还回步骤⑥。

对比基本算法与改进的算法可知，本文从基本算法的不足与所研究问题的特点出发，有针对性地提出了周期性震荡函数指导果蝇飞行，以及分离不同类型变量的更新方式，从而能有效地解决所研究的问题。

2 数值实验与分析

本文选取四个混合整数非线性规划问题进行仿真[13-14]，该类函数是由实际的工程问题抽象而成。并且选取的函数具有很强的检验性，第一个测试用例连续性的变量多于离散性的变量，而第二个测试用例则相反，这样可以综合的检验出改进算法的性能。并将改进算法与粒子群算法[15]（PSO）及差分进化算法[16]（DE）进行了比较。PSO 算法中惯性约束系数 $W = 0.5$，加速系数 $c1 = 2$、$c2 = 2$；DE 算法中变异算子 $F = 0.5$，交叉算子 $CR = 0.15$。PSO 及 DE 的种群数与迭代数均和果蝇算法一致。果蝇群数量 $mSize = 100$，迭代次数 $pSize = 1\,000$，步长参数 $\alpha = 0.65$、$\beta = 0.35$、$T = 20$，p 为当前的迭代次数。该实验的计算机配置与环境为：英特尔 CPU、3.5 GHz，8 GB 内存，Win7 64 位操作系统。本文主要从算法的稳定性、收敛速度及最优解的精度等方面与粒子群算法（PSO）及差分进化算法（DE）进行了综合的比较。本文算法在下文图中用 OURS 表示。

2.1 混合整数非线性规划问题

混合整数非线性规划其一般形式为：

$$\min f(x,y)$$

$$\text{s.t.}\begin{cases} g_i(x,y) \leqslant 0 \quad i = 1,2,\cdots,m \\ h_j(x,y) \leqslant 0 \quad j = 1,2,\cdots,n \\ x_L \leqslant x \leqslant x_U \\ y_L \leqslant y \leqslant y_U \end{cases} \tag{23}$$

其中，m、n 分别表示不等式约束或等式约束的个数，x 表示 n_e 维连续变量，y 表示 n_c 维整数或离散变量。

（1）测试用例 1：

$$\min f_1(x,y) = -y + 2x - In(x/2)$$

$$\text{s.t.}\begin{cases} -x - In(x/2) + y \leqslant 0 \\ 0.5 \leqslant x \leqslant 1.5 \\ y \in \{0,1\} \end{cases} \tag{24}$$

最优解为 $(x.y) = (1.375,1)$，最优值为 2.214。

（2）测试用例 2：

$$min\ f_2(x,y) = -0.7y + 5(x_1 - 0.5)^2 + 0.8$$

$$\text{s.t.}\begin{cases} -e^{(x_1-0.2)} - x_2 \leqslant 0 \\ x_2 + 1.1y + 1 \leqslant 0 \\ x_1 - 1.2y - 0.2 \leqslant 0 \\ 0.2 \leqslant x_1 \leqslant 1 \\ -2.225\,54 \leqslant x_2 \leqslant -1 \\ y \in \{0,1\} \end{cases} \tag{25}$$

最优解为 $(x_1,x_2,y) = (0.941\,94, -2,1)$，最优值为 1.076 54。

（3）测试用例 3：

$$\begin{aligned} \min f_4(x,y) = {} & 0.622\,4(0.062\,5y_1)x_1x_2 + \\ & 1.778\,1(0.062\,5y_2)x_1^2 + \\ & 3.166\,1(0.062\,5y_1)^2x_2 + \\ & 19.84(0.062\,5y_1)^2x_1 \end{aligned}$$

$$\text{s.t.}\begin{cases} 0.019\,3x_1 - 0.062\,5y_1 \leqslant 0 \\ 0.009\,54x_1 - 0.062\,5y_2 \leqslant 0 \\ 750 \times 1\,728 - x_1^2x_2 - 4/3\pi x_1^3 \leqslant 0 \\ x_2 - 240 \leqslant 0 \\ x \in [10,240] \\ y \in [1,15] \end{cases} \tag{26}$$

已知最优解为 $(x_1,x_2,y_1,y_2) = (38.860\,1, 221.365,12,6)$，最优值为 5 850.38。

（4）测试用例 4：

$$\begin{aligned} \min f_3(x,y) = {} & (y_1 - 1)^2 + (y_2 - 1)^2 + \\ & (y_3 - 1)^2 - In(1 + y_4) + \\ & (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 \end{aligned}$$

$$\text{s.t.}\begin{cases} y_1+y_2+y_3+x_1+x_2+x_3-5\leqslant 0\\ y_3^2+x_1^2+x_2^2+x_3^2-5.5\leqslant 0\\ y_1+x_1-1.2\leqslant 0\\ y_2+x_2-1.8\leqslant 0\\ y_3+x_3-2.5\leqslant 0\\ y_4+x_1-1.2\leqslant 0\\ y_2^2+x_2^2-1.64\leqslant 0\\ y_3^2+x_3^2-4.25\leqslant 0\\ y_2^2+x_3^2-4.64\leqslant 0\\ x\geqslant 0\\ y\in\{0,1\} \end{cases} \tag{27}$$

已知最优解为:$(x_1,x_2,x_3,y_1,y_2,y_3,y_4)=(0.2,1.280\,624,1.954\,483,1,0,0,1)$,最优值为3.557 463。

2.2　实验结果与分析

本文从以下几个方面来对比各种算法在解决上述两个测试用例时的性能。图 1 – 图 4 表示三种算法在迭代次数为 1 000 时算法的寻优情况。其横坐标表示为 1 000 次迭代数据每 20 次取样所形成的 50 个样本数,纵坐标为每个取样样本上的函数最优值。

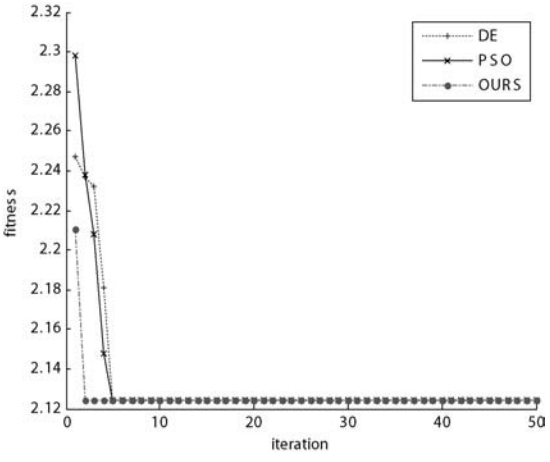

图 1　三种算法解决测试用例 1 的性能对比图

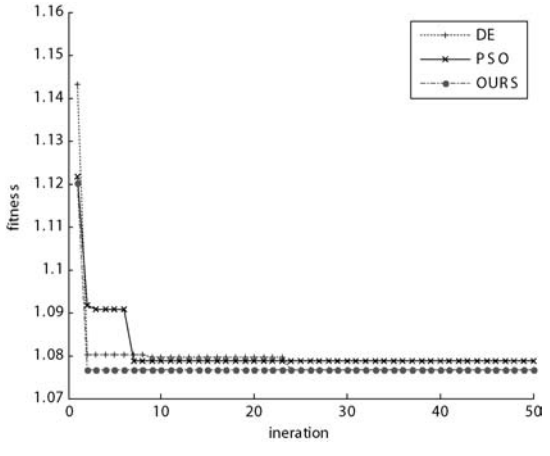

图 2　三种算法解决测试用例 2 的性能对比图

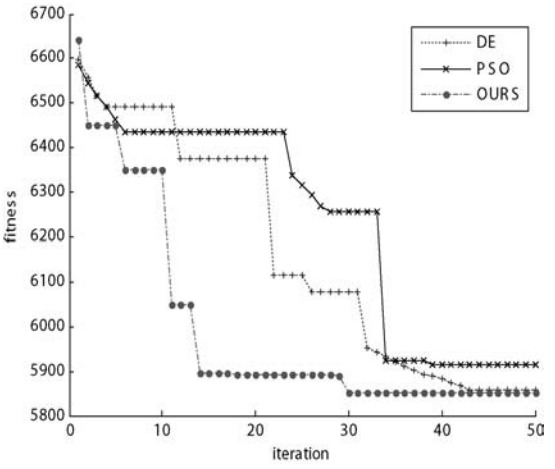

图 3　三种算法解决测试用例 3 的性能对比图

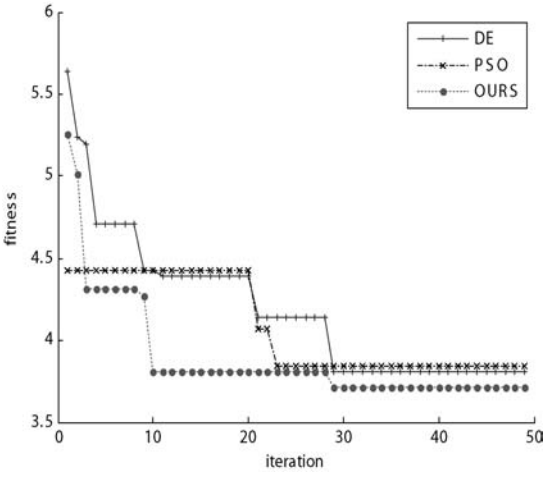

图 4　三种算法解决测试用例 4 的性能对比图

从图 1 和图 2 中可以看出,由于函数 f_1 及函数 f_2 的变量较少,离散性变量不多于连续性变量,并且变量的域间较小,所以从图 1 和图 2 上可以看出三种算法在对此类型的问题求解的效率较高、效果也较好。但是,本文算法(图中的 OURS)从收敛速度来看,性能明显比其他两种要好。其原因一是函数的特点造成的,二是改进的果蝇算法中加入了步长函数可以指导果蝇的飞行,加快果蝇寻找到最优值。图 3 中函数 f_3 的变量类型的个数相同,但是,变量的域间距较大。从图 3 中可以看出本文算法优于 DE 及 PSO 算法。图 4 中函数 f_4 的离散性变量多于连续性变量,从图中可以看出三种算法都在某一段时间内陷入了局部最优,但改进的果蝇算法最后能找到最优值,而其他算法在面对多离散性变量问题时效果就不是那么好了。

图 5 – 图 8 表示了三种算法迭代次数分别为 100、500 和 1 000 次,独立运行 50 次时,得到的 50 个最优值中达到所规定的误差精度的成功次数占总次数的比例。其中达到函数 f_1、f_2 的规定最优值的误差精度为 1%,而函数 f_2、f_4 的误差精度为 5%。

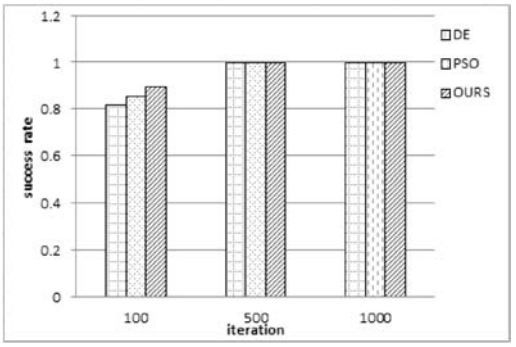

图 5 三种算法解决测试用例 1 的性能对比图

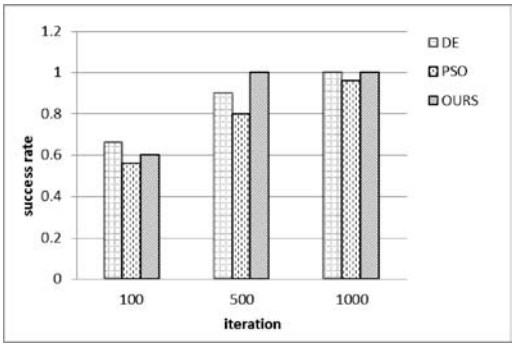

图 6 三种算法解决测试用例 2 的性能对比图

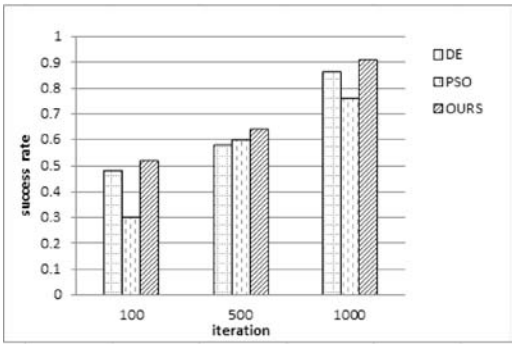

图 7 三种算法解决测试用例 3 的性能对比图

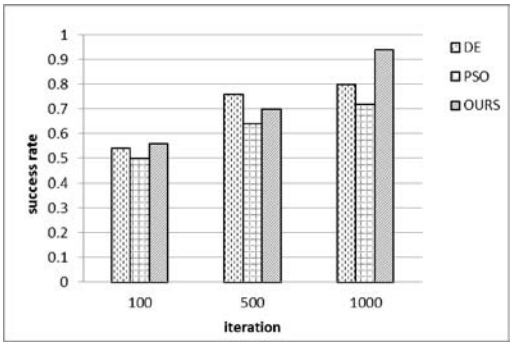

图 8 三种算法解决测试用例 4 的性能对比图

从图 5 – 图 8 的直方图可以得出这三种算法在解决不同复杂度的函数时,它们的效率及成功率也不相同,对于较复杂的离散变量,改进的果蝇算法有较高的成功率,表明算法的稳定性较强。

表 1 表示了该三种算法它们独自运行 50 次,每次迭代 1 000 次时的实验结果,从表中可以看出,本算法比其他两种算法稳定性较好,且精度要高。

表 1 三种算法最优值与最差值对比表

算法	函数	平均值	最优值	最差值
DE	f_1	2.124 467 585	2.124 468 714	2.124 657 813
	f_2	1.079 565	1.076 562	1.086 631
	f_3	5 988.61 729	5 859.201	6 053.638 889
	f_4	3.962 152	3.574 362	4.23 848
PSO	f_1	2.124 469 255	2.124 469 019	2.124 737 585
	f_2	1.080 817	1.076 584	1.096 838
	f_3	6 064.1 474 305	5 909.924 721	6 165.373 525
	f_4	4.154 035	3.640 347	5.16 593
本文算法	f_1	2.124 461 531	2.124 467 585	2.124 487 585
	f_2	1.076 523	1.076 543	1.076 554
	f_3	5 852.189 288	5 850.389 165	5 869.564 243
	f_4	3.748 585	3.557 476	4.1 862 714

3 结 语

本文提出了一种求解混合整数非线性规划问题的改进果蝇算法,该算法通过分离不同类型的变量,且对变量的更新方式设置不同的步长约束函数,使得该改进的算法能有效地提高稳定性、避免陷入局部最优。实验结果表明,改进的果蝇算法比其他算法在求解此类非线性规划问题上的收敛速度快、精度高。然而,该改进的果蝇算法的适应性较低,虽对所求解问题的效果较好,但对其他规划问题的求解还有待深入的研究。

参 考 文 献

[1] Zhang X S. Introduction to mathematical programming[J]. Introduction to mathematical programming,2000,46(4071):273 – 298.

[2] 张莉,冯大政,李宏. 求解整数非线性规划结合正交杂交的离散 PSO 算法[J]. 控制与决策,2012,27(9):1387 – 1387.

[3] 刘明明,崔春风,童小娇,等. 混合整数非线性规划的算法软件及最新进展[J]. 中国科学:数学,2016(1):001.

[4] 刘振泽,许洋,王峰明. 改进差分进化算法在非线性模型预测控制中的应用[J]. 北京工业大学学报,2015,41(5):680 – 685.

[5] Guo X, Li X. A Genetic Algorithm for a Class of Fractional Bilevel Programming Problems with Interval Coefficients[J]. Advances in Applied Mathematics,2015,4(1):63 – 69.

[6] 吴小文,李擎. 果蝇算法和 5 种群智能算法的寻优性能研究[J]. 火力与指挥控制,2013,38(4):17 – 20.

[7] 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.

(下转第 308 页)

文提出的 DTJ-ALOHA 算法有一定的实用价值,可应用在 RFID 系统大规模标签待识别环境中。

参 考 文 献

[1] 孙其博,刘杰,黎羴,等. 物联网:概念、架构与关键技术研究综述[J]. 北京邮电大学学报,2010,33(3):2-3.

[2] 黄玉兰,夏璞,夏岩,等. 物联网射频识别(RFID)核心技术详解[M]. 北京:人民邮电出版社,2012:22-25.

[3] 石封查,崔琛,余剑. 基于标签运动的一种新型 RFID 防碰撞算法[J]. 计算机科学,2013,40(6):76-79.

[4] Finken Zeller K. 射频识别(RFID)技术—无线电感应的应答器和非接触 IC 卡原理与应用[M]. 2 版. 北京:电子工业出版社,2001:98-101.

[5] EPC Global. EPC Radio-Frequency Identity Protocols Gengernation-1[S]. American:UCC,2013-10-31.

[6] 肖海慧,王红明. 一种基于 DFSA 防碰撞协议的 FBF 改进算法研究[J]. 计算机应用与软件,2013,30(7):305-308.

[7] Cha J R, Kim J H. Dynamic framed slotted ALOHA algorithms using fast tag estimation method for RFID system [C]//Consumer Communications and NETWORKING Conference,2006. Ccnc. 2006:768-772.

[8] EPC Global. EPC Radio-Frequency Identity Protocols Gengernation-2 UHF RFID Protocol for Communication at 860MHZ ~ 960MHZ Version[S]. California: EPC Global,2008.

[9] 徐圆圆,曾隽芳,刘禹. 基于 Aloha 算法的帧长及分组数改进研究[J]. 计算机工程,2008,3(12):57-59.

[10] 吴淼,刘德盟,张钊锋. 基于 EPC Gen2 防碰撞机制的研究与优化[J]. 微电子学与计算机,2013,30(5):101-104.

(上接第 235 页)

[7] Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation [C]//Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. New York, NY, USA: ACM Press, 2000: 1-12.

[8] Han J, Pei J, Mortazavi-Asl B, et al. FreeSpan: frequent pattern-projected sequential pattern mining[C]//Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM Press, 2000: 355-359.

[9] Pei J, Han J, Mortazavi-Asl B, et al. PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth [C]//Proceedings of the 17th International Conference on Data Engineering. IEEE Computer Society, 2001: 215-224.

[10] 张坤,朱扬勇. 无重复投影数据库扫描的序列模式挖掘算法[J]. 计算机研究与发展,2007,44(1):126-132.

[11] 张利军,李战怀,王淼. 基于位置信息的序列模式挖掘算法[J]. 计算机应用研究,2009,26(2):529-531.

[12] 刘栋,尉永清,薛文娟. 基于 Map Reduce 的序列模式挖掘算法[J]. 计算机工程,2012,38(15):43-45.

[13] 吴信东,谢飞,黄咏明,等. 带通配符和 One-Off 条件的序列模式挖掘[J]. 软件学报,2013,24(8):1804-1815.

[14] 刘端阳,冯建,李晓粉. 一种基于逻辑的频繁序列模式挖掘算法[J]. 计算机科学,2015,42(5):260-264.

[15] 朱扬勇,熊赟. DNA 序列数据挖掘技术[J]. 软件学报,2007,18(11):2766-2781.

[16] 熊赟,陈越,朱扬勇. DnaReSM:一个基于多支持度的 DNA 重复序列挖掘算法[J]. 计算机科学,2007,34(2):211-212,封四.

[17] 周溜溜,业宁,徐昇,等. 基于频繁子树挖掘的 DNA 重复序列识别方法[J]. 微电子学与计算机,2011,28(9):193-196,201.

[18] 姜华,孟志青,周克江. DNA 序列频繁近似模式挖掘[J]. 生物信息学,2013,11(1):11-15.

[19] 毛国君,杨静欣. 一种基于关联矩阵的高效 DNA 序列挖掘算法[J]. 计算机科学与应用,2015,5(8):271-277.

(上接第 266 页)

[8] Yuan X, Dai X, Zhao J, et al. On a novel multi-swarm fruit fly optimization algorithm and its application[J]. Applied Mathematics & Computation, 2014, 233(3): 260-271.

[9] Pan W T. A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example[J]. Knowledge-Based Systems, 2012, 26(2): 69-74.

[10] 郑晓龙,王凌,王圣尧. 求解置换流水线调度问题的混合离散果蝇算法[J]. 控制理论与应用,2014,31(2):159-164.

[11] 杨书,舒勤,何川. 改进的果蝇算法及其在 PPI 网络中的应用[J]. 计算机应用与软件,2014,31(12):291-294.

[12] Shan D, Cao G H, Dong H J. LGMS-FOA: An Improved Fruit Fly Optimization Algorithm for Solving Optimization Problems [J]. Mathematical Problems in Engineering, 2013, 2013(7): 1256-1271.

[13] Lin Y C, Hwang K S, Wang F S. A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems[J]. Computers & Mathematics with Applications, 2004, 47(8-9): 1295-1307.

[14] Deep K, Singh K P, Kansal M L, et al. A real coded genetic algorithm for solving integer and mixed integer optimization problems[J]. Applied Mathematics & Computation, 2009, 212(2): 505-518.

[15] 唐俊. PSO 算法原理及应用[J]. 计算机技术与发展,2010,20(2):213-216.

[16] 梅觅,薛惠锋,谷雨. 旅行商问题的改进差分进化方法[J]. 信息技术,2011(2):20-23.