文章编号:1001 - 9081(2007)03 - 0630 - 03

求解旅行商问题的位置—次序编码差分演化算法

贺毅朝[1],寇应展[2],陈致明[2]
(1. 石家庄经济学院,信息工程学院,河北 石家庄 050031;
2. 军械工程学院,计算机工程系,河北 石家庄 050003)
(heyichao@ sjzue. edu. cn)

摘 要:首先利用"差异算子"和"选择算子"描述了差分演化算法(DE)的基本原理,然后提出了一种新的、通用的特殊编码方法:位置-次序编码法,并利用此编码方法,提出了求解著名旅行商问题的离散差分演化算法:基于位置-次序编码的差分演化算法(PODE)。对于 TSPLIB 中两个不同规模的旅行商问题实例的计算表明,PODE 算法具有极好的收敛性和稳定性。

关键词:差分演化算法;位置次序编码;旅行商问题
中图分类号: TP181 **文献标识码**:A

Differential evolution algorithm with position-order encoding for solving traveling salesman problem

HE Yi-chao[1], KOU Ying-zhan[2], CHEN Zhi-ming[2]
(1. *School of Information Engineering, Shijiazhuang University of Economics, Shijiazhuang Hebei* 050031, *China*;
2. *Computer Engineering Department, Ordnance Engineering College, Shijiazhuang Hebei* 050003, *China*)

Abstract: First, the differential evolution algorithm with differential operator and selection operator was introduced. Then, a new and universal encode method, position-order encoding method, was proposed. Based on the approach, a discrete differential evolution algorithm, position-order encoding based differential evolution algorithm (PODE), was proposed. The numerical results about two TSP instances in TSPLIB show that PODE has excellent convergence and stability.

Key words: differential evolution (DE); position-order encoding; traveling salesman problem (TSP)

0 引言

差分演化(Differential Evolution,DE)算法[1,2]是一种演化算法,非常适合于求解连续域最优化问题。目前,DE 已经被成功地用于解决函数优化、多目标优化、神经网络训练和系统控制等不同领域中的连续域数值最优化问题[3-5]。但是,对于解决离散域上的组合优化问题,特别是一类 NP 难问题的应用研究还是一片空白。旅行商问题(Traveling Saleman Problem,TSP)是著名的 NP-hard 问题,其描述为:给定 n 个城市和两两城市之间的距离,求一条访问各城市一次且仅一次的最短路径。TSP 是典型的组合优化问题,经常用来检验智能演化算法的有效性[6]。

本文首先利用"差异算子"和"选择算子"描述了 DE 的基本原理;然后基于排序的思想,采用个体分量的位置与其大小次序相结合所构成的自然数序列(即位置-次序编码)表示一条 TSP 路径,从而提出了一种新的、通用的编码方法,解决了基于实数运算的 DE 算法求解 TSP 的关键性难题:TSP 路径的自然数编码表示与实数运算的不兼容性。并利用这种编码方法,提出了求解 TSP 的离散差分演化算法:基于位置-次序编码的差分演化算法(简记 PODE)。通过对 TSP 实例的计算表明:PODE 算法既保持了实数编码 DE 的优点,又非常适于求解组合最优化问题,其全局收敛性能和稳定性能都极佳。

1 差分演化算法

差分演化算法是一种基于实数编码的演化算法,在每一代演化过程中,首先利用差异算子(Differential Operator, DO)得到一个由父子混合个体构成的中间种群,然后利用选择算子(Select Operator, SO)基于优胜劣汰的竞争机制重组新一代种群。由于 DE 算法的 DO 算子和 SO 算子的计算简单且高效,使得 DE 算法比粒子群优化算法和遗传算法的收敛速度更快,全局收敛性能更优[7]。目前,DE 算法存在十几种不同的模式,各模式除了差异算子有异以外,其余均基本相同。下面基于 DE/r/1/bin 模式,在严格形式化描述的基础上,以演化算法的一般实现框架阐明差分演化算法的原理。

首先定义基本个体与中间个体如下:

定义 1 在 DE 算法中,称用于表示问题潜在解的 d 维向量 $\boldsymbol{X}_i = (x_{i1}, x_{i2}, \cdots, x_{id})^{\mathrm{T}} \in S$ 为种群中的第 i 个基本个体。

定义 2 在 DE 算法中,称由当前种群重组后所得到的中间种群中的个体为中间个体。对应于基本个体 $\boldsymbol{X}_i = (x_{i1}, x_{i2}, \cdots, x_{id})^{\mathrm{T}}$ 的中间个体用向量 $\boldsymbol{V}_i = (v_{i1}, v_{i2}, \cdots, v_{id})^{\mathrm{T}}$ 表示,$\boldsymbol{V}_i \in S$。

设所求解问题的目标函数 $f(\boldsymbol{X})$ 为最小化函数,其搜索空间 $S = \{\boldsymbol{X} \mid \boldsymbol{X} = (x_1, x_2, \cdots, x_d)^{\mathrm{T}} \wedge L_i \leqslant x_i \leqslant U_i, i = 1, 2, \cdots, d\} \subseteq \mathbf{R}^d$,$\mathbf{R}$ 为实数集,d 为解空间的维数。又设 t 表示 DE 算法

收稿日期:2006 - 06 - 28;**修订日期**:2006 - 09 - 03 **基金项目**:国家自然科学基金重点资助项目(60471022)

作者简介:贺毅朝 (1969 -),男,河北晋州人,讲师,硕士,主要研究方向:智能计算、计算机密码学、计算复杂性理论; 寇应展 (1962 -),男,陕西西安人,教授,主要研究方向:优化算法、软件工程、计算机网络安全; 陈致明 (1940 -),男,吉林洮南人,教授,博士生导师,主要研究方向:计算机网络与信息安全、算法理论、计算机高等体系结构.

的演化代数,s 为种群规模,$\boldsymbol{V}_i(t+1)=(v_{i1}(t+1),v_{i2}(t+1),\cdots,v_{id}(t+1))^{\mathrm{T}}$ 是第 $t+1$ 代中间种群中的第 i 个中间个体,$\boldsymbol{X}_i(t+1)$ 和 $\boldsymbol{X}_i(t)$ 分别为第 $t+1$、t 代种群中第 i 个基本个体,则差异算子的计算公式为:

$$v_{ij}(t+1)=\begin{cases}x_{p_1j}(t)+\alpha(x_{p_2j}(t)-x_{p_3j}(t), & \\ \qquad r\leqslant CR \text{ 或 } j=R(i) & (1)\\ x_{ij}(t), \quad r>CR \text{ 或 } j\neq R(i) & \end{cases}$$

其中 $1\leqslant i\leqslant s$,$1\leqslant j\leqslant d$,$p_1\neq p_2\neq p_3\neq i$;$\alpha<1$ 称为缩放因子,r 是一个随机数且 $r\sim(0,1)$;CR 称为交叉概率,并且 $CR\in[0,1]$,$R(i)$ 是 $[1,n]$ 上的一个随机正整数。

选择算子的计算公式为:

$$\boldsymbol{X}_i(t+1)=\begin{cases}\boldsymbol{V}_i(t+1), & f(\boldsymbol{V}_i(t+1))<f(\boldsymbol{X}_i(t))\\ \boldsymbol{X}_i(t), & f(\boldsymbol{V}_i(t+1))\geqslant f(\boldsymbol{X}_i(t))\end{cases}\quad(2)$$

令 DE 算法的演化迭代过程中的全局最优个体为 $\boldsymbol{X}_{best}$,则 DE 算法的形式化描述如下:

算法 1　DE Algorithm

Initialize population $\{\boldsymbol{X}_i\mid 1\leqslant i\leqslant PopulationSize\}$;
Do
　For $i=1$ to $PopulationSize$
　　For $j=1$ to $Dimension$
　　If $(r\leqslant CR\vee j=R(i))$
　　then $v_{ij}=x_{p1j}+\alpha(x_{p2j}-x_{p3j})$
　　else $v_{ij}=x_{ij}$;
　　Next j
　　If $(f(v_{i1},v_{i2},\cdots,v_{id})<f(x_{i1},x_{i2}\cdots,x_{id})$ then $X_i=V_i$;
　Next i
　Compute $(\boldsymbol{X}_{best})$;
　Until termination criterion is met;
Return $(\boldsymbol{X}_{best},f(\boldsymbol{X}_{best}))$.

在算法 1 中,Initialize population 表示随机产生初始大小为 $PopulationSize$ 的种群,$Dimension=d$ 为解空间的维数;$f(v_{i1},v_{i2},\cdots,v_{id})$ 和 $f(x_{i1},x_{i2}\cdots,x_{id})$ 分别为 $\boldsymbol{V}_i(t+1)$ 和 $\boldsymbol{X}_i(t)$ 对应的函数值,$\alpha\sim(0,1)$ 是一个随机数。此外,由上述描述可以看出,DE 算法所涉及到的数据结构和运算非常简单,相比遗传算法和粒子群优化算法更易于编程实现,而且运算速度也比它们更快[2,7]。

2　旅行商问题

TSP 是图论、运筹学和组合优化中的经典 NP-hard 问题,对其常用的求解方法有近似算法和演化算法两大类。其中近似求解算法有最邻近法、最小生成树法和最小权匹配法等[8];演化求解算法有模拟退火算法、遗传算法、免疫算法和蚁群算法等[6,9]。从图论的角度而言,TSP 的数学描述为:

给定无向图 $G=<V,E>$,其中 V 为顶点集,E 为边集,且 $|V|=n$。$D=(d_{ij})_{n\times n}$ 为顶点距离矩阵,其元素 d_{ij} 表示城市 i 与 j 间的距离。又设 $\boldsymbol{\pi}=(\pi_1,\pi_2,\cdots,\pi_n)$ 表示 $1,2,\cdots,n$ 的一个排列,则 TSP 为约束组合优化问题:

$$\min f(\boldsymbol{\pi})=\min\sum_{i=1}^{n}d_{\pi i,\pi(i+1)},\text{假设 }\pi_{(i+1)}=\pi_1\quad(3)$$

$$\text{s.t.}\begin{cases}d_{ij}\geqslant 0,d_{ii}=0,d_{ij}=d_{ji},\text{其中 }1\leqslant i,j\leqslant n\\ \text{满足三角不等式:}d_{ij}+d_{jk}\geqslant d_{ik},\text{其中 }1\leqslant i,j,k\leqslant n\end{cases}\quad(4)$$

TSP 的描述很简单,但是却不容易求解,因为该问题的解空间基数为 $(n-1)!/2$,若用穷举搜索法求解,其算法复杂性是指数阶的,即随着 n 的增大,求解该问题所用时间呈指数增长,所以寻求和研究 TSP 的有效智能化算法是问题的关键。

3　基于位置-次序编码的差分演化算法

DE 算法虽然成功应用于连续域上的数值优化问题,但对于离散域上组合优化问题却不能直接使用,原因在于差异算子的计算公式(1)是基于实数域上的四则运算,而离散域上组合优化问题的解(个体)往往由自然数(向量)表示,差异算子的计算公式(1)对自然数不满足封闭性。所以,在保留差异算子高效计算的基础上,如何使自然数的四则运算封闭,是将 DE 算法用于求解组合优化问题的关键所在。通过实验和分析发现,在 DE 算法中,个体中的每一分量虽然是实数,但不同分量之间是有大小之分的,而且各分量的大小是不断变化的。那么将各分量按大小排序后,利用其对应的次序表示此分量所在位置上所对应的一个自然数,由此即构成一个自然数向量,可用于表示组合优化问题的解向量。这样,DE 算法中的每一个基于实数编码的个体,均对应于一个基于自然数编码的解向量。显然,利用这种方法即可表示离散域上组合优化问题的解,从而实现对该问题的求解。下面给出这种编码方法的定义。

定义 3　设 $\boldsymbol{X}_i(t)=(x_{i1}(t),x_{i2}(t),\cdots,x_{id}(t))^{\mathrm{T}}$ 为 DE 算法的第 t 代种群中第 i 个基本个体,令 $Ord(x_{ij}(t))$ 表示 $x_{i1}(t)$,$x_{i2}(t),\cdots,x_{id}(t)$ 按由小到大排序以后 $x_{ij}(t)$ 在排序序列中的次序,$1\leqslant j\leqslant d$,则称:

$$\boldsymbol{C}(t)=(Ord(x_{i1}(t)),Ord(x_{i2}(t)),\cdots,Ord(x_{id}(t)))^{\mathrm{T}}\quad(5)$$

为对应于第 t 代种群中基本个体 $\boldsymbol{X}_i(t)$ 的位置-次序编码向量(Position-Order Encoding Vector, POEV)。

例如,若基本个体 $\boldsymbol{X}(t)=(2.0337,-1.9720,4.9901,-4.0015)^{\mathrm{T}}$,则由定义 3 可知,$\boldsymbol{X}(t)$ 对应的位置-次序编码向量为 $\boldsymbol{C}(t)=(2,3,1,4)^{\mathrm{T}}$。

显然,$\boldsymbol{C}(t)$ 是一个基于自然数编码的 d 维向量,而且 $Ord(x_{i1}(t)),Ord(x_{i2}(t)),\cdots,Ord(x_{id}(t))$ 对应于 $1,2,\cdots,d$ 的一个排列,故可用于表示离散域上组合优化问题的解向量编码。由于这种编码方法是由个体分量的位置和排序后的次序所决定的,因此称这种编码方法为位置-次序编码方法(Position-Order Encoding Method, POEM)。

由定义 3 可以看出,POEM 是求解组合优化问题的一种通用编码方法,该方法可用于任意基于实数编码的演化算法,例如粒子群优化算法和思维进化算法等,使这些算法也适用于求解组合优化问题。下面将 POEM 与 DE 算法相结合,提出一种求解 TSP 的基于位置-次序编码的差分演化算法(PODE),首次将差分演化算法用于求解组合优化问题中的著名 TSP。

设算法的种群规模为 s,t 表示当前的迭代次数,d 为解空间维数。第 t 代基本个体的编码由有序对 $(\boldsymbol{X}_i(t),\boldsymbol{C}_i(t))$ 表示,其中 $\boldsymbol{X}_i(t)=(x_{i1}(t),x_{i2}(t),\cdots,x_{id}(t))^{\mathrm{T}}\in\mathbf{R}^d$ 为实数向量,$\boldsymbol{C}_i(t)=(c_{i1}(t),c_{i2}(t),\cdots,c_{id}(t))^{\mathrm{T}}$ 为对应于 $\boldsymbol{X}_i(t)$ 的位置-次序编码向量,$c_{ij}(t)=Ord(x_{ij}(t))$;第 $t+1$ 代中间个体的编码由 $(\boldsymbol{V}_i(t+1),\boldsymbol{CC}_i(t+1))$ 表示,其中 $\boldsymbol{V}_i(t+1)=(v_{i1}(t+1),v_{i2}(t+1),\cdots,v_{id}(t+1))^{\mathrm{T}}\in\mathbf{R}^d$,$\boldsymbol{CC}_i(t+1)=(cc_{i1}(t+1),cc_{i2}(t+1),\cdots,cc_{id}(t+1))^{\mathrm{T}}$ 为对应于 $\boldsymbol{V}_i(t+1)$ 的位置-次序编码向量,$cc_{ij}(t+1)=Ord(v_{ij}(t+1))$。

定义 4 新差异算子(New Differential Operator, NDO)的计算公式定义如下:

$$v_{ij}(t+1)=\begin{cases}x_{p_1j}(t)+\alpha(x_{p_2j}(t)-x_{p_3j}(t)),\\ \qquad r\leqslant CR \text{ 或 } j=R(i)\\ x_{ij}(t),\ r>CR \text{ 或 } j\neq R(i)\end{cases}\tag{6}$$

$$cc_{ij}(t+1)=Ord(v_{ij}(t+1))\tag{7}$$

其中 $1\leqslant i\leqslant s$,$1\leqslant j\leqslant d$,$p_1\neq p_2\neq p_3\neq i$;$\alpha<1$ 称为缩放因子,r 是一个随机数且 $r\sim(0,1)$,CR 称为交叉概率,并且 $CR\in[0,1]$,$R(i)$ 是 $[1,n]$ 上的一个随机正整数。

定义 5 新选择算子(New Select Operator, NSO)的计算公式定义如下:

$$\boldsymbol{X}_i(t+1)=\begin{cases}\boldsymbol{V}_i(t+1), & f(\boldsymbol{CC}_i(t+1))<f(\boldsymbol{C}_i(t))\\ \boldsymbol{X}_i(t), & f(\boldsymbol{CC}_i(t+1))\geqslant f(\boldsymbol{C}_i(t))\end{cases}\tag{8}$$

利用定义 4,5 定义的 NDO 算子和 NSO 算子,并令 X_{best} 表示算法演化过程中的全局最优个体,则 PODE 算法可形式化描述为:

算法 2 PODE Algorithm

```
Initialize population: {(X_i, C_i) | 1 ≤ i ≤ PopulationSize};
Do
  For i = 1 to PopulationSize
    For j = 1 to Dimension
      If (r ≤ CR ∨ j = R(i))
        then v_ij = x_p1j + α(x_p2j − x_p3j)
        else v_ij = x_ij;
    Next j
    CC_i = Ord(V_i);                    //可利用快速排序算法实现
    If (f(cc_i1, cc_i2, ..., cc_id) < f(c_i1, c_i2 ... c_id)
      then (X_i, C_i) = (V_i, CC_i);
  Next i
Compute (X_best);
Until termination criterion is met;
Return (X_best, f(X_best))
```

算法 2 中的参数说明同算法 1(略)。由以上论述可以看出:在 PODE 算法中,NDO 的本质是在 DO 的基础上增加了一个 $Ord(\cdot)$ 运算,算法的基本演化特征并没有改变,而 NSO 与 SO 的实质相同,因此,PODE 算法在保持 DE 算法的基本演化特征的前提下,借助于个体的位置到次序的转化,成为适用于求解组合优化问题的演化算法。由于 $Ord(\cdot)$ 运算所需的排序算法(如快速排序算法)的时间复杂性是 $O(n\log n)$,因此 $Ord(\cdot)$ 运算在一定程度上增加了算法的时间开销;但由于 $Ord(\cdot)$ 运算具有高效性,而且 PODE 算法继承了 DE 算法的快速全局收敛性优点,使求得最优解所需的迭代次数大大减少,故可保持较快的求解速度。

4 实验结果

为了验证 PODE 算法的有效性和实用性,选用 TSPLIB[10] 中两个不同规模的 TSP 标准测试实例:ulysses16-TSP 和 att48-TSP,在 DELL Pentium(R)4-CPU 1.69GHz 型微计算机上利用 VC++6.0 编程求解。在求解 ulysses16-TSP 时,算法的种群规模取为 50;在求解 att48-TSP 时,种群规模为 100。此外,设定算法的缩放因子 $\alpha=0.5$,交叉概率 $CR=0.3$,每个 TSP 标准测试实例均连续求解 10 次。PODE 算法求解这两个 TSP 实例的结果见表 1。

ulysses16-TSP 中 16 个城市的序号和坐标 (x, y) 分别为:

1 (38.24,20.42),2 (39.57,26.15),3 (40.56,25.32),4 (36.26,23.12),5 (33.48,10.54),6 (37.56,12.19),7 (38.42,13.11),8 (37.52,20.44),9 (41.23,9.10),10 (41.17,13.05),11 (36.08,−5.21),12 (38.47,15.13),13 (38.15,15.35),14 (37.51,15.17),15 (35.49,14.32),16 (39.36,19.56)

att48-TSP 中 48 个城市的序号和坐标 (x, y) 分别为:

1 (6734,1453), 2 (2233,10), 3 (5530,1424), 4 (401,841), 5 (3082,1644), 6 (7608,4458), 7 (7573,3716), 8 (7265,1268), 9 (6898,1885), 10 (1112,2049), 11 (5468,2606), 12 (5989,2873), 13 (4706,2674), 14 (4612,2035), 15 (6347,2683), 16 (6107,669), 17 (7611,5184),18 (7462,3590), 19 (7732,4723), 20 (5900,3561), 21 (4483,3369), 22 (6101,1110), 23 (5199,2182), 24 (1633,2809), 25 (4307,2322), 26 (675,1006), 27 (7555,4819), 28 (7541,3981), 29 (3177,756), 30 (7352,4506), 31 (7545,2801), 32 (3245,3305), 33 (6426,173), 34 (4608,1198), 35 (23,2216), 36 (7248,3779), 37 (7762,4595), 38 (7392,2244), 39 (3484,2829), 40 (6271,2135), 41 (4985,140), 42 (1916,1569), 43 (7280,4899), 44 (7509,3239), 45 (10,2676), 46 (6807,2993), 47 (5185,3258), 48 (3023,1942)

表 1 PODE 算法求解两个 TSP 标准测试实例的结果

TSP 标准实例	最优解个体的编码	最优解	最小迭代次数	平均时间/s	成功率(%)
ulysses16	1,14,13,12,7,6,15,5,11,9,10,16,3,2,4,8	193.12	1227	6.91	100
att48	1,8,38,31,44,18,7,28,6,37,19,27,17,43,30,36,46,33,20,47,21,32,39,48,5,42,24,10,45,35,4,26,2,29,34,41,16,22,3,23,14,25,13,11,12,15,40,9	15349.471	3051	15.38	90

对于 ulysses16-TSP,PODE 算法只需要搜索解空间的 $(1227\times50)/\{(16-1)!/2\}=0.000009.4\%$;而对于 att48-TSP,PODE 算法只需要搜索解空间的 $(3051\times100)/\{(48-1)!/2\}=(2.3594\times10^{-51})\%$。

由表 1 可以看出,无论是 PODE 算法求得最优解所需的最小迭代数,还是求得最优解所用的平均时间,都是较低的,这说明 PODE 算法求解这两个 TSP 标准实例的效率是非常高的,速度也是非常快的;同时,由表 1 还可以看出,PODE 算法求得的这两个 TSP 标准实例的最优解的概率也都很高。总之,虽然仅对两个 TSP 标准实例进行了求解测试,但从结果可以看出,利用位置-次序编码方法的 PODE 算法具有极好的收敛性和稳定性,在保持 DE 算法优势的基础上,可有效地求解 TSP 等组合优化问题;此外,由 PODE 算法的原理还可以看出:在算法中的 $\boldsymbol{CC}_i=Ord(\boldsymbol{V}_i)$ 步骤之后,还可以尝试通过引入求解 TSP 常用的"倒位算子[6]"等运算,进一步探讨对 PODE 算法的加强与改进,从而更好地提高其求解 TSP 的求解效率。

(下转第 641 页)

图 5是 MMAS 训练过程中适应度值的进化曲线,作为评价指标,它随 Epochs 的增加而逐步递减。

表 1　参数比较

算法	参数	序号 1	2	3	4	5	MSE (10^{-3})	TMSE (10^{-3})
遗传	w	0.3059	0.4337	0.4273	0.4713	0.4930	9.66970	
	u	0.7779	0.7635	0.7679	0.7378	0.7936	13.1040	10.53684
	τ	0.2954	0.3372	0.3109	0.3568	0.3139	8.83665	
蚁群	w	0.2223	0.5062	0.2731	0.3798	0.5947	0.33912	
	u	0.6860	0.8136	0.6457	0.8440	0.7105	2.36121	1.78528
	τ	0.2796	0.4419	0.3009	0.4814	0.4240	2.65551	

为比较算法性能,笔者对同一个 FPN 模型用另一全局搜索能力出众的标准遗传算法(GA)来实现。算法中 GA 种群数为 50,训练样本数 50,变异率从 0.05 到 0.1。将 GA 与 ACA 共同迭代 300 次后的结果进行对比,如表 1 所示。表中数据是 10 次运行程序结果的平均值。显然,ACA 无论在哪组参数的精确度上都要优于常用的 GA 全局优化算法,经 ACA 改进型算法 MMAS 优化的效果十分明显。另外,在两种算法的运行速度上,选取有代表性的迭代 200 和 300 次的时间进行对比,10 次实验的平均用时值如图 6 所示。可见,在两种情况下,ACA 的用时都要少于 GA,ACA 的运行效率明显高于 GA。

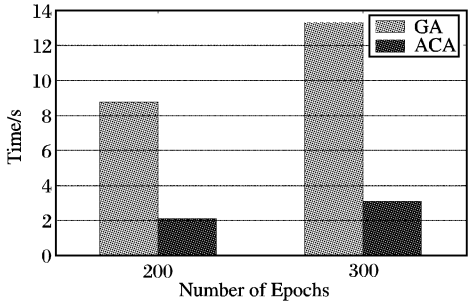

图 6　GA 与 MMAS 运行速度比较

最后,为验证其泛化功能,任取 10 组非样本中的输入数据,对按 MMAS 训练后的 FPN 模型进行模糊推理。推理结果如表 2 中的实际输出栏所示,它们反映的是结果命题 $M(p_8)$ 的隶属度值,输出端均方误差和为 9.4681×10^{-4},结果令人满意。

表 2　期望输出和实际输出

序号	期望输出	实际输出	序号	期望输出	实际输出
1	0.652400	0.685034	6	0.409192	0.411359
2	0.590800	0.620175	7	2.223200	2.244070
3	0.876400	0.904015	8	0.471184	0.475376
4	1.103200	1.143881	9	0.756560	0.791225
5	0.989520	1.045914	10	1.612800	1.629620

4　结语

该文运用蚁群算法中的 MMAS 对 FPN 模型的各参数进行寻优,使 FPN 具有神经网络一样的学习能力。经寻优后的 FPN 参数精度较高,且泛化能力较强。搜索过程中,采用了线程技术,实现多组参数的同时查找,有效减少了程序运行时间。但由于蚁群算法属全局搜索算法,面对大空间复杂系统的优化问题时,仍不可避免地存在用时多精度不够的状况,今后的研究还需配合局部搜索能力出众的 BP 算法,结合激活函数的优化,来进一步提高 FPN 模型的寻优精确度。

表 2　期望输出和实际输出

序号	期望输出	实际输出	序号	期望输出	实际输出
1	0.652400	0.685034	6	0.409192	0.411359
2	0.590800	0.620175	7	2.223200	2.244070
3	0.876400	0.904015	8	0.471184	0.475376
4	1.103200	1.143881	9	0.756560	0.791225
5	0.989520	1.045914	10	1.612800	1.629620

参考文献:

[1] 鲍培明. 基于 BP 网络的模糊 Petri 网的学习能力[J]. 计算机学报, 2004, 27(5).

[2] 李士勇. 蚁群算法及其应用[M]. 哈尔滨: 哈尔滨工业大学出版社, 2004. 17 – 96.

[3] LI X-O. Dynamic knowledge inference and learning under adaptive fuzzy Petri net frame work[J]. IEEE Transactions on Systems, Man and Cybernetic – Part C: Application and Reviews, 2000, 30(4): 442 – 449.

[4] 周良忠. C ++ 面向对象多线程编程[M]. 北京: 人民邮电出版社, 2003. 76 – 120.

[5] 丛爽. 神经网络、模糊系统及其在运动控制中的应用[M]. 合肥: 中国科学技术大学出版社, 2001. 208 – 218.

(上接第 632 页)

参考文献:

[1] STORN R, PRICE K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of Global Optimization, 1997, 11(4): 341 – 359.

[2] PRICE K. Differential evolution: A fast and simple numerical optimizer[A]. Proceedings NAFIPS'96[C]. 1996. 524 – 525.

[3] BABU BV, MUNAWAR SA. Different Evolution for the Optimal Desig of Heat Exchangers[A]. Proceedings of All-India seminar on Chemical Engineering Progress on Resource Development: A Vision 2010 and Beyond, IE(I)[C]. 2000.

[4] CRUZ ILL, VAN WILLIGENBURG LG, VAN STRATEN G. Efficient differential evolution algorithms for multimodal/ optimal control problem[J]. Application Soft Computing, 2003, 3(2): 97 – 122.

[5] ABBASS H. Self-adaptive pareto differential evolution[A]. Proceedings of the IEEE 2002 Congress on Evolutionary Computation[C]. Honolulu, Hawaii, IEEE Press, 2002.

[6] 刘勇, 康立山, 陈毓屏. 非数值并行算法(第二册)——遗传算法[M]. 北京: 科学出版社, 2003.

[7] VESTERSTROM J, THOMSEN R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithm on numerical Benchmark problems[A]. Evolutionary Computation, CEC2004[C]. 2004, 2. 1980 – 1987.

[8] 耿素云, 屈婉玲, 王捍贫. 离散数学教程[M]. 北京: 北京大学出版社, 2003.

[9] 黄席樾, 张著洪, 何传江, 等. 现代智能算法理论及应用[M]. 北京: 科学出版社, 2005.

[10] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/[EB/OL], 2006.