基于离散数字编码的蚁群连续优化算法*)

吴广潮[1,2] 黄 翰[2]

(华南理工大学数学科学学院 广州 510640)[1] (华南理工大学计算机科学与工程学院 广州 510640)[2]

摘 要 本文提出了一种基于离散编码的蚁群连续优化算法(CACO-DE),用于求解连续优化问题。以往蚁群算法(ACO)的研究,以求解离散优化问题为主,较少涉及连续优化问题。与经典的ACO算法不同,CACO-DE将有限精度的实数转化为一个数字串,数字串的每位取0到9之间的数字,从而实现了用离散编码描述实数的效果。CACO-DE延用了经典ACO算法的框架,并加入了特殊的选择机制、信息素更新方式和局部搜索策略。测试实验结果表明:CACO-DE比以往同类算法求解速度更快且精度更高。

关键词 蚁群算法,连续优化,离散数字编码

Ant Colony Continuous Optimization Based on Discrete Numerical Encoding

WU Guang-Chao[1,2] HUANG Han[2]

(School of Mathematical Sciences, South China University of Technology, Guangzhou 510640)[1]

(School of Computer Science and Engineering, South China University of Technology, Guangzhou 510640)[2]

Abstract The presented paper proposes an ant colony algorithm for continuous optimization (CACO-DE). ACO algorithms are always used for discrete optimization problems, but rarely for continuous optimization. CACO-DE is designed based on the numerical encoding in which each real number is changed into a string made up of characters {0,…, 9}. The length of encoding depends on the accuracy and dimension of the solution. Artificial ants construct solutions being guided by a high dimension pheromone vector. The framework of the proposed algorithm is similar to the classical ACO except for the updating rule and local search strategy. Some preliminary results obtained on benchmark problems show that the new method can solve continuous optimization problems faster than other ant and non-ant methods.

Keywords Ant colony algorithm, Continuous optimization, Discrete numerical encoding

1 引言

蚁群算法(ACO)[1]是由M. Dorigo及其同伴在上世纪90年代提出的一种仿生算法,用于求解如旅行商问题[2]之类的组合优化问题。目前,ACO算法的应用已经扩展到解决多种优化问题,如:Vehicle Routing[3]、Quadratic assignment[3]、QoS[4]、Job shop[5]等,但这些问题几乎都是离散优化问题。

与遗传算法、粒子群算法和进化规划算法不同,ACO算法求解连续优化问题的设计研究较少。第一种求解连续函数优化问题的蚁群算法为Continuous ACO(CACO)算法[6],其主要思想是将连续区间分段,离散化后区间段视为TSP问题中的城市。CACO算法虽然实现了ACO算法求解连续优化问题0的突破,但是求解效果并不理想。后期又对CACO算法作了些改进[7,8],提高了求解的精度,但是改进的程度有限。后来相应又有API[9]和CIAC[10]。另外两种算法出现,取得了一定的改进效果,但这些算法加入了遗传算法等其他计算工具的策略,只是用了ACO算法的框架而已。最新的算法还有基于正态分布的ACO算法[11],然而这种算法也需要将区间分段离散化,从而会出现两个缺点:1. 算法求解精度有限;2. 算法求解的计算复杂度较高,需要花费较多的函数评估次数。

作为改进,本文在文[12]基础上提出了一种新型的求解连续优化问题的ACO算法:基于离散编码的蚁群算法(CACO-DE)。实验结果表明:CACO-DE比以往其他ACO算法求解的效果更好,而且速度更快。

2 ACO算法基本思想介绍

自然界蚂蚁在其经过的路径上会留下某种生物信息物质(信息素),该物质会吸引蚁群中的其它成员再次选择该段路径。食物与巢穴之前较短的路径容易积累较多的信息素,因而使得更多的蚂蚁选择走该段路径,最终几乎所有的蚂蚁都集中在最短路径上完成食物的搬运。M. Dorigo等从此现象中抽象出路径选择和信息素积累的数学模型,作为蚁群算法的核心;并通过对蚂蚁寻找最短路径的计算机模拟,实现了对TSP问题的求解[2]。

按M. Dorigo的设计[3],蚁群算法的基本框架如图1所示。

```
Procedure ACOMetaheuristic
    ScheduleActivities
        ConstructAntsSolutions
        UpdatePheromones
        DaemonActions
    End-ScheduleActivities
End-procedure
```

图1 蚁群算法(ACO)的基本框架

一般情况下,ACO算法可以分为三个部分:生成解(ConstructAntsSolutions),更新信息素(UpdatePheromones)和附加策略(DaemonActions)。

*)国家自然科学基金项目(10471045)、广东省自然科学基金(04020079)、华南理工大学自然科学基金(B13-E5050190)。吴广潮 讲师,博士研究生,研究领域为算法设计与分析,数据库与信息处理;黄 翰 博士研究生,研究领域为进化计算方法的理论基础,进化计算方法的优化设计及其应用。

3 实数的离散编码表示

在CACO-DE算法提出之前，需要给出用离散数字编码表示实数的方法。首先，给出 M. Dorigo 对实数连续优化问题的定义[13]。

定义 1 定义 $P=(S,C,f)$ 为连续优化问题，其中 S 为有限实数集搜索空间，C 是约束集合，$f: S\to R^+$ 为最大或者最小目标函数，R^+ 为正实数集。

下面给出一个定义，实现将有限长度实数转化为离散编码。

定义 2 定义 $f_1: R_m\to D$ 为将有限长度实数转化为离散编码，其中 $\forall x\in R_m$ 是一个精度为 10^{-m} 的实数，且 $D=\{d\mid d=(d_0,d_1,\cdots,d_L),d_i\in\{0,1,\cdots,9\},i=1,\cdots,L;d_0\in\{0,1\}\}$，$d_i=\left\lfloor(|x|-\sum_{j=1}^{i-1}d_j\times10^{L-m-j})/10^{L-m-i}\right\rfloor$ 和 $d_0=0(x\geqslant0)$ 或 $d_0=1(x<0)$。

同理，可以给出定义 2 的逆映射。

定义 3 定义 $f_1^{-1}: D\to R_m$ 将离散编码转化为有限长度实数，其中 $D=\{d\mid d=(d_0,d_1,\cdots,d_L),d_i\in\{0,1,\cdots,9\},i=1,\cdots,L;d_0\in\{0,1\}\}$，$\forall x\in R_m$ 是一个精度为 10^{-m} 的实数，$x=\sum_{i=1}^{L}d_i\times10^{-(i-L+m)}(d_0=0)$ 或 $x=(-1)\times\sum_{i=1}^{L}d_i\times10^{-(i-L+m)}(d_0=1)$。

例如，假设 $d=(1,1,6,8,3,3,6)$ 和 $m=2$，$f_1^{-1}(d)=x=-1683.36$，$L=6$。因此，一个连续优化问题的解可以编码为一个数字串，即实现了编码的离散化。这样就可以用 ACO 算法的选择机制等算法策略。CACO-DE 就是用定义 2 进行编码，而用定义 3 进行解码。

4 CACO-DE 算法的设计

CACO-DE 的流程满足 ACO 算法的基本框架(图 1)，如图 2 所示。

算法 CACO-DE
输入: 满足定义 1 的连续优化问题 $P=(S,C,f)$.
初始化解和信息素，按定义 2 进行编码
$S_{best}\leftarrow NULL$
while 停机条件未满足 **do**
  for j=1,…,n_k **do** { n_k 是虚拟人工蚂蚁的数量}
    $s_i\leftarrow ConstructSolution(t)$ {具体设计见 4.1 节}
    执行局部更新策略 {具体设计见 4.2 节}
    if ($f(s_i)>f(s_{best})$) or ($s_{best}=NULL$) **then**
      $s_{best}\leftarrow s_i$
    endif
  endfor
  $s_{best}\leftarrow$ **LocalSearch**(s_{best}). {具体设计见 4.3 节}
  执行全局更新策略 {具体设计见 4.2 节}
endwhile
将当前最优解 s_{best} 根据定义 3 解码
输出: s_{best}

图 2 CACO-DE 算法的流程

4.1 生成解的选择机制

ACO 算法的核心部分是基于信息素的选择机制[1]，CACO-DE 算法的信息素设计取决于问题变量的数量和精度。CACO-DE 是用一个高维向量 τ 刻画信息素的，其中：$\tau(i,j)$ 是 10×10 的实数矩阵，$i=1,\cdots,n$ 和 $j=2,\cdots,L(L>m)$；$\tau(i,1)$ 是一个 2×10 的实数矩阵，$\tau(i,0)=(\tau(i,0,0),\tau(i,0,1))$ 是一个 2 维向量。

给定一个连续最优化问题 $P=(S,C,f)$，$\forall s\in S$ 是一个向量 $s=(x^{(1)},\cdots,x^{(i)},\cdots,x^{(n)})$，其中 $x^{(i)}\in R$ 是精度为 $10^{-m(i)}$ 的第 i 个变量。根据定义 2，s 可以编码为 $d=(d^{(1)},\cdots,d^{(i)},\cdots,d^{(n)})=(d_1,\cdots,d_p,\cdots,d_{(L+1\times n)})$。

CACO-DE 第 p 个位置的选择机制根据 p 值($p=1,\cdots,(L+1)\times n$)，分三种情况：

当 $(p-1)\bmod(L+1)=0$ 时，$\tau(i,0,0)$ 选择“0”作为解组成部分 $d_0^{(i)}$ 的信息素，即 $x^{(i)}\geqslant0$。同理，$\tau(i,0,1)$ 是选择“1”的信息素，即 $x^{(i)}<0$。此时，选择概率为：

$$P\{d_p=e\}=\frac{\tau(i,0,e)}{\tau(i,0,0)+\tau(i,0,1)} \tag{1}$$

其中，$i=1,\cdots,n$ 和 $e\in\{0,1\}$。

当 $(p-1)\bmod(L+1)=1$ 时，$\tau(i,1,0,a)$ 选择“a”作为解组成部分 $d_1^{(i)}$ 的信息素($x^{(i)}\geqslant0$)。同理，$\tau(i,1,1,a)$ 是选择 $d_1^{(i)}=a$，即($x^{(i)}<0$)。因此，

$$P\{d_p=a\mid d_{p-1}=e\}=\frac{\tau(i,1,e,a)}{\sum_{l=0}^{9}\tau(i,1,e,l)} \tag{2}$$

其中，$i=1,\cdots,n$ 和 $a\in\{0,\cdots,9\}$。

当 $(p-1)\bmod(L+1)>1$ 时，$\tau(i,j)$ 中的 $\tau(i,j,a,b)$ 是选择“b”作为 $d^{(i)}$ 的第 j 个组成元素，其中 a 是第 $j-1$ 个元素。因此，

$$P\{d_p=b\mid d_{p-1}=a\}=\frac{\tau(i,j,a,b)}{\sum_{l=0}^{9}\tau(i,j,a,l)} \tag{3}$$

其中，$i=1,\cdots,n,j=2,\cdots,L(L>m)$ 和 $a,b\in(0,\cdots,9)$。

因此，CACO-DE 的选择机制就是用公式(1)～(3)，且加入了 ACS 算法[14]的模拟退火方式。

4.2 信息素更新

CACO-DE 算法有两种信息素更新方式：局部更新和全局更新。

在局部更新中，信息素向量 τ 根据每只蚂蚁找到的解 $s_k(t)=(x_k^{(1)},\cdots,x_k^{(i)},\cdots,x_k^{(n)})$ 进行修改，其中 $S_k(t)$ 可以编码为 $d_k(t)=(d^{(1)},\cdots,d^{(i)},\cdots,d^{(n)})=(d_1,\cdots,d_p,\cdots,d_{(L+1\times n)})$。更新时根据 p 值($p=1,\cdots,(L+1)\times n$)，分三种情况，如公式(4)～(6)所示：

当 $(p-1)\bmod(L+1)=0$ 时，

$$\tau(i,0,e)\leftarrow\tau(i,0,e)\times(1-\rho)+\Delta\tau\times\rho \tag{4}$$

其中 $d_p=d_0^{(i)}=e\in\{0,1\}$。

当 $(p-1)\bmod(L+1)>1$ 时，

$$\tau(i,j,a,b)\leftarrow\tau(i,j,a,b)\times(1-\rho)+\Delta\tau\times\rho \tag{5}$$

其中 $d_p=d_1^{(i)}=a\in\{0,1,\cdots,9\}$，$d_{p-1}=d_0^{(i)}=e\in\{0,1\}$。

当 $(p-1)\bmod(L+1)>1$ 时，

$$\tau(i,j,a,b)\leftarrow\tau(i,j,a,b)\times(1-\rho)+\Delta\tau\times\rho \tag{6}$$

其中 $d_{p-1}=d_{j-1}^{(i)}=a\in\{0,1,\cdots,9\}$，$d_p=d_j^{(i)}=b\in\{0,1,\cdots,9\}$，$j=2,\cdots,L$。

当 $f(s_k(t))>f(s_k(t-1))$ 时，$\Delta\tau=\left|\frac{f(s_k(t))-f(s_k(t-1))}{f(s_k(t-1))}\right|$。当 $f(s_k(t))\leqslant f(s_k(t-1))$，$\Delta\tau=0$。

在全局更新中，信息素向量 τ 根据每代最优解 $s_{best}=(x_{best}^{(1)},\cdots,x_{best}^{(i)},\cdots,x_{best}^{(n)})$ 进行修改，其中 $s_{best}(t)$ 可以编码为 $s_{best}(t)=(d^{(1)},\cdots,d^{(i)},\cdots,d^{(n)})=(d_1,\cdots,d_p,\cdots,d_{(L+1)\times n)})$。更新时根据 p 值($p=1,\cdots,(L+1)\times n$)，分三种情况，如公式(7)～(9)所示：

当 $(p-1)\bmod(L+1)=0$ 时，

$$\tau(i,0,e)\leftarrow\tau(i,0,e)\times(1-\alpha)+\Delta\tau_0\times\alpha \tag{7}$$

其中 $d_p=d_0^{(i)}=e\in\{0,1\}$。

当 $(p-1)\bmod(L+1)=1$ 时，

$$\tau(i,1,e,a)\leftarrow\tau(i,0,e,a)\times(1-\alpha)+\Delta\tau_0\times\alpha \tag{8}$$

其中 $d_p=d_1^{(i)}=a\in(0,1,\cdots,9)$，$d_{p-1}=d_0^{(i)}=e\in\{0,1\}$。

当 $(p-1)\bmod(L+1)>1$ 时，

$$\tau(i,j,a,b)\leftarrow\tau(i,j,a,b)\times(1-\alpha)+\Delta\tau_0\times\alpha \tag{9}$$

其中 $d_{p-1}=d_{j-1}^{(i)}=a\in\{0,1,\cdots,9\}$，$d_p=d_j^{(i)}=b\in\{0,1,\cdots,9\}$，$j=2,\cdots,L$。

因为 $f(s_{best}(t))\geqslant f(s_{best}(t-1))$，$\Delta\tau_0=\left|\frac{f(s_{best}(t))-f(s_{best}(t-1))}{f(s_{best}(t-1))}\right|$。除此以外，按照 MMAS 算法[15]设置了信息素值的下界 $\tau_{\min}>0$。

4.3 局部搜索策略

在大多数的 ACO 算法[4]中，局部搜索是一种常用且必要的辅助策略[16]。根据 4.2 小节的更新公式(4)～(9)，选择机制仅仅与信息素有关。为了加速 CACO-DE 算法的全局和局部搜索速度，加入了一个局部搜索策略(图 3)。

子算法 局部搜索

输入：将 $s_{best}(t)$ 编码为 $d=(d^{(1)},...,d^{(i)},...,d^{(n)})=(d_1,...,d_p,...,d_{(L+1)\times n})$

for i=1 to n **do** {n is the number of variables}

产生一个均匀分布的随机数 $r\in\{1,...,L\}$

从{0,1,...,9}中选择最优的 $d^{(i)}_r$ 更新 $s_{best}(t)$

endfor

输出：局部优化后的 $s_{best}(t)$

图 3 局部搜索策略的流程

5 实验结果与分析

本节给出了 CACO-DE 和其他优化算法的求解连续优化问题的对比效果。测试的问题来源于文[8,11]，每个测试问题各计算了 100 次，与最优解的平均误差值见表 1，函数的评估次数见表 2。CACO-DE 参数设置如：$\alpha=\rho=0.1$ 和 $n_k=10$。

CGA、ECTS、ESA、ACO 算法求得误差 1.00E-3 水平的平均函数评估次数直接引用文[8,11]的结果，所以在一些问题没有给出计算结果(记为"---")。CACO[8]和 CACO-DE 则是在相同的计算平台下实现的，当连续 50 次迭代得不到更优解时，算法停机。

表 1 CACO-DE 和其他蚁群算法求解效果对比

No.	测试问题名称	Optimal	CACO	ACO	CACO-DE
1	De Jong's	3905.93	0	---	0
2	Goldstein & Price	3	0	1.00E-4	0
3	Martin & Gaddy	0	1.00E-3	---	4.44E-07
4	Rosenbrock #1	0	1.00E-3	---	4.44E-07
5	Rosenbrock #2	0	1.00E-3	3.00E-3	2.03E-04
6	Sphere model	0	1.00E-3	1.00E-04	1.00E-06

表 2 CACO-DE 和其他优化算法计算评估函数平均次数的对比

No.	CGA	ECTS	ESA	ACO	CACO	CACO-DE
1	---	---	---	---	6000	304
2	---	---	---	---	5330	582
3	410	231	783	364	1688	296
4	---	---	---	---	6842	313
5	960	480	796	2905	8471	475
6	750	338	---	695	22050	566

实验结果不仅说明了 CACO-DE 算法的可行性，也体现其在平均求解效果和计算时间上的优势。表 1 表明 CACO-DE 比 ACO 算法[11]和 CACO 算法[8]求解精度更高，而表 2 则说明 CACO-DE 是所有对比连续优化算法中消耗时间最少的。

表中 CACO 算法的设计思想是：将连续区间分段，离散化后区间段视为 TSP 问题中的城市，求解精度上受到很大制约。从表 1 的实验结果看，除了第 1 和 2 个问题外，CACO 算法在其他问题得到的最优解精度很有限，大大逊于 CACO-DE 算法。根据设计原理，CACO 算法提高求解准确率需要进一步细分搜索区间，进而造成求解时间急剧增长。因此，表 2 的实验结果也反映了 CACO 计算消耗时间明显大于 CACO-DE。可以认为，CACO-DE 超越了以往 ACO 连续优化算法[8,11]的性能，离散数字编码机制是算法成功的关键。通过离散数字编码机制，CACO-DE 不需要用额外计算消耗得到更优解，信息素更新机制可以引导人工蚂蚁选择更优数字编码，从而构成更高质量的解。

结束语 从蚁群算法提出以来，关于求解连续优化问题的研究结果极少。本文正是在这一背景下，提出了一种基于离散数字编码的蚁群连续优化算法(CACO-DE)。CACO-DE 的关键设计在于将实数转化为离散数字编码，从而可以用经典 ACO 算法的流程进行处理。结合离散编码机制，我们设计了相应的选择策略、信息素更新方式和局部搜索策略。最后，求解连续优化问题的实验结果表明：CACO-DE 克服了以往 ACO 连续优化算法在求解精度和消耗时间上的缺点，比多种同类算法求解速度更快且质量更高。未来的研究将对 CACO-DE 算法进行改进，测试更多更高维的优化问题，并将算法应用于带约束优化问题以及对其收敛性和收敛速度进行分析。

参 考 文 献

1 Dorigo M, Caro G D, Gambardella L M. Ant algorithms for Discrete Optimization. Artificial Life, 1999, 5: 137～172

2 Dorigo M, Gambardella L M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, 1997, 1 (1): 53～66

3 Dorigo M, Stützle T. Ant Colony Optimization. Cambridge, MA: MIT Press, 2004

4 李有梅，王文剑，徐宗本. 关于求解难组合优化问题的蚁群优化算法. 计算机科学，2002，29(3)：115～118

5 陈岩，杨华江，沈林成. 基于再励学习蚁群算法的多约束 QoS 路由方法. 计算机科学，2007，34(5)：25～27

6 Bilchev G, Parmee IC. The Ant Colony Metaphor for Searching Continuous Design Spaces. In: Fogaty T C, ed. Proceedings of the AISB Workshop on Evolutionary Computation, Berlin, Germany: LNCS, Springer-Verlag, 1995, 993: 25～39

7 Wodrich M, Bilchev G. Cooperative distributed search: the ant's way. Control & Cybernetics, 1997 (3): 413～446

8 Mathur M, Karale S B, Priye S, et al. Ant Colony Approach to Continuous Function Optimization. Ind. Eng. Chem. Res., 2000, 39: 3814～3822

9 Monmarché N, Venturini G, Slimane M. On how Pachycondyla apicalis ants suggest a new search algorithm. Future Generation Computer Systems, 2000, 16: 937～946

10 Dréo J, Siarry P. A New Ant Colony Algorithm Using the Hierarchical Concept Aimed at Optimization of Multimedia Continuous Functions. In: Dorigo M, et al., eds.. ANTS 2002, Lecture Notes in Computer Science: 2002, 2463: 216～221

11 Socha K. ACO for Continuous and Mixed-Variable Optimization. In: Dorigo M, et al., eEds. ANTS 2004, Lecture Notes in Computer Science, 2004, 3172: 25～36

12 Huang H, Hao Z F. ACO for continuous optimization based on discrete encoding. In: Dorigo M, et al., eds. ANTS 2006, Lecture Notes in Computer Science, 2006, 4150: 504～505

13 Dorigo M, Blum C. Ant colony optimization theory: A survey. Theoretical Computer Science, 2005, 344: 243～278

14 Dorigo M, Gambardella L M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, 1997, 1 (1): 53～66

15 Stützle T, Hoos H H. MAX-MIN ant system, Future Generation Computer Systems, 2000, 16 (8): 889～914

16 黄翰，郝志峰，吴春国，秦勇. 蚁群算法的收敛速度分析. 计算机学报，2007，30 (8)：1343～1353