离散型细菌觅食算法求解 TSP*

王勇臻，陈 燕，李桃迎
(大连海事大学 交通运输管理学院，辽宁 大连 116026)

摘 要： 旅行商问题(TSP)是组合优化问题的典型代表，针对 TSP 的求解提出一种离散型细菌觅食(DBFO)算法。该算法通过结合 2-opt 算法设计了一种适合处理离散型变量的趋化算子，将细菌觅食算法推广到了离散情形。同时，结合 TSP 的特点，在迁徙算子中引入基因库的思想来指导新个体的生成，提高了算法的搜索效率。通过对 TSPLIB 标准库中 22 个实例进行仿真实验。实验结果表明，该算法能够有效求解城市规模 500 以下的 TSP，与混合蚁群算法和离散型萤火虫群算法相比，具有更好的全局收敛性和稳定性。

关键词： 离散型细菌觅食优化算法；旅行商问题；2-opt；基因库
中图分类号： TP301.6 **文献标志码：** A **文章编号：** 1001-3695(2014)12-3642-04
doi:10.3969/j.issn.1001-3695.2014.12.032

Discrete bacteria foraging optimization algorithm for solving TSP

WANG Yong-zhen, CHEN Yan, LI Tao-ying
(*Transportation Management College, Dalian Maritime University, Dalian Liaoning* 116026, *China*)

Abstract: Traveling salesman problem(TSP) is a typical representative of combinatorial optimization problems. This paper proposed a discrete bacteria foraging optimization(DBFO) algorithm to tackle the TSP. It desinged a new chemotaxis operator which was combined with 2-opt algorithm so as to handle discrete variables, and it made the bacteria foraging algorithm extend to the discrete situation. With the characteristics of the TSP, it designed a new elimination operator which is combined with gene pool so as to direct the generation of new individuals, which improved the search efficiency. Simulation experiments on 22 TSP problems from TSPLIB show that, the proposed algorithm can solve the TSP which city scale is below 500 effectively, and is superior to hybrid ACO and discrete GSO algorithm.

Key words: discrete bacteria foraging optimization algorithm; traveling salesman problem(TSP); 2-opt; gene pool

0 引言

旅行商问题(TSP)是一个典型的 NP 难题，可以简要描述为：给定一个图 $G=\langle V,E\rangle$，每条边 $e\in E$ 上有非负权值 $w(e)$，寻找 G 的 Hamilton 圈 C，使得 C 的总权 $W(C)=\sum w(e)$ 最小。其模型在网络路由、物流配送、电路板布局等领域具有广泛的应用，但至今尚未找到非常有效的求解方法。随着城市数目的增多，求解问题的空间、时间复杂度将呈指数级增长，这时传统的一些精确型算法就会陷入困境。

目前求解 TSP 的优化算法分两种类型：与问题本身特征相关的局部启发式搜素算法，常见的有 2-opt、3-opt 和 Lin-Kernighan 等[1]，这类算法过于依靠问题本身特征，容易陷入局部最优；随着人工智能的发展，出现了许多独立于问题的优化算法，已成功地应用于 TSP，如遗传、蚁群、粒子群算法以及免疫算法等。

细菌觅食优化(bacteria foraging optimization，BFO)算法是近年来提出的群智能优化算法，其理论基础是最优觅食理论[2,3]。虽然 BFO 算法不像遗传算法、蚁群算法等应用那么广泛，但是已经逐渐被应用于多模态函数优化[4]、灰度图像边缘检测[5]、PID 控制器的设计[6]、认知无线网络频谱分配[7]、车间作业调度[8]等方面，且表现出了良好的优化性能。BFO 算法通常被用于解决连续优化问题，国内外文献还很少有使用 BFO 算法解决组合优化问题的方案。

基于此想法，针对 TSP 的求解，提出一种离散型细菌觅食(discrete bacteria foraging optimization，DBFO)算法。该算法通过引入 2-opt 算法设计了一种适合处理离散型变量的趋化算子；本文采纳了文献[9]提出的基因库的思想，在迁徙算子中引入启发信息来优化新生成的个体，提高算法的搜索效率。通过对 TSPLIB 标准库中的 22 个实例进行计算机仿真实验，实验结果表明，在中小规模 TSP 中算法均能够找到最优解，在大规模 TSP 中计算出的最优解与已知最优解的误差也在 1% 左右，比目前同类文献算法给出的结果更优。

1 BFO 算法和 λ-opt 算子简述

1.1 BFO 算法

大肠杆菌在觅食过程中，一边感应自身周围的化学物质浓度(如营养液、有毒物质等)，一边做出远离或趋向该种物质的智能行为，基于此，Passino 等人[2,3]于 2002 年提出了细菌觅食

收稿日期：2013-12-23；修回日期：2014-02-12 基金项目：国家自然科学基金资助项目(71271034)；辽宁省科技重大项目(2011219009)；辽宁省教育厅科学研究一般项目(L2012173)

作者简介：王勇臻(1990-)，男，福建永定人，硕士研究生，主要研究方向为数据挖掘(KuaDMU@163.com)；陈燕(1952-)，女，辽宁大连人，教授，博导，博士，主要研究方向为管理科学与决策、知识管理、数据仓库与数据挖掘；李桃迎(1983-)，女，安徽宿州人，副教授，博士，主要研究方向为聚类分析.

算法。BFO 算法通过趋化、复制、迁徙三个算子对细菌群体的觅食过程进行了模拟[10]。

细菌在整个觅食过程中有两个基本动作: 翻转和游动。BFO 算法的趋化算子对这两种基本动作进行了模拟,首先定义细菌向任意方向移动单位步长为翻转,其更新公式为

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + c(i)\Delta(i)\sqrt{\Delta^T(i)\Delta(i)} \quad (1)$$

其中: $\theta^i(j,k,l)$ 为第 i 个细菌经历第 j 次趋化、第 k 次复制、第 l 次迁徙后所处的位置; $c(i)$ 表示单位步长; $\Delta(i)$ 表示随机翻转方向。如果细菌完成一次翻转后,其适应值得到了改善,则沿同一方向继续移动若干步,此过程定义为游动; 如果没有得到改善,则继续翻转; 如果达到最大尝试次数,则停止该细菌的趋化算子,跳转到下一个细菌执行趋化算子。

复制算子模拟了细菌群体优胜劣汰的过程,经过一段时间的觅食过程后,部分觅食能力弱的细菌会被自然淘汰,为了维持种群规模不变,剩余的觅食能力强的细菌会进行繁殖。设细菌种群规模为 S,在复制过程中,群体中要淘汰的个体个数为 $Sr = S/2$。首先将群体中的个体按照其适应值优劣进行排序,将排在较靠后的 Sr 个个体删除,剩下的 Sr 个个体进行自身的拷贝,即复制出与原来的个体一样的新个体。

迁徙算子模拟了细菌由于环境突变而随机移动到其他区域的过程,算法通过设定一个固定的迁徙概率,对算法迭代中的每个细菌进行随机性移动。若满足条件,则随机生成一个新个体并替换原个体,等同于将原细菌个体移动到了一个新位置。

1.2　λ-opt 算法

λ-opt 算法最早由 Lin 等人于 1965 年提出,是一种常用的路径改进算法,其核心思想是对于给定的初始路径,通过每次交换 λ 条边来改进当前路径[11]。对 TSP 而言,λ 值越大,改进后的路径接近最优路径的可能性也越大。然而,对 n 个城市进行 λ-opt 操作的时间复杂度是 $O(n^\lambda)$,当 λ 值稍大时,其工作量将非常巨大[11]。因此,文献中经常使用的是实现简单的 2-opt 来进行路径改进。求解极小化问题的 2-opt 算法核心步骤描述如下:

a) 选取初始路径 P。

b) 对路径 P 中所有不相邻的两点 C_i、C_j 执行步骤 c)。

c) $\Delta d \leftarrow d(C_i,C_{i+1}) + d(C_j,C_{j+1}) > d(C_i,C_j) + d(C_{i+1},C_{j+1})$,其中 $d(C_i,C_j)$ 表示 C_i 和 C_j 之间的距离。

若 $\Delta d > 0$,则路径长度得到改善,删除边 (C_i,C_{i+1}) 和 (C_j,C_{j+1}),然后分别连接边 (C_i,C_j) 和 (C_{i+1},C_{j+1}) 且分别相反箭头指向顶点 C_{i+1} 和 C_j。

d) 当满足停止条件时,输出路径 P 和 $W(P)$。

2　求解 TSP 的离散细菌觅食算法

细菌觅食算法中,求解最优解的过程就是大肠杆菌寻找食物最丰富区域的过程。考虑简单性,以数字量"适应值"来衡量细菌所在区域的特点,表 1 给出了 TSP 与细菌觅食行为对应关系。

表 1　TSP 与细菌觅食行为对应关系

细菌觅食行为	TSP
细菌所在区域	遍历所有城市的路径
细菌的适应值	路径长度
细菌的移动	路径的变换
最高适应值	最短路径长度

2.1　编码设计

TSP 的编码策略主要有近邻编码、次序编码、二进制编码、矩阵编码、边编码和路径编码[12]。本文选用文献中常用的路径编码。设 n 个城市编号为 $1,2,\cdots,n$,$P_k = C_1^k, C_2^k, \cdots, C_n^k$ 为第 k 个细菌的区域编码,它是 $1,2,\cdots,n$ 的一个随机排列,表示一条可行路径,路径起点城市是 C_1^k,最后一个城市是 C_n^k,则路径的总长度为

$$d(P_k) = \sum_{i=1}^{n-1} d(C_i^k, C_{i+1}^k) + d(C_n^k, C_1^k) \quad (2)$$

其中: $d(C_1^k, C_n^k)$ 表示城市 C_i^k 与城市 C_{i+1}^k 之间的距离。在算法中 P_k 的总长 $d(P_k)$ 用来评价个体的好坏,适应值函数 $f(P_k)$ 取 $d(P_k)$ 的倒数,即 $f(P_k) = 1/d(P_k)$。

2.2　改进的趋化算子

细菌觅食算法最初是为解决连续的函数优化问题而设计的,其趋化算子只能操作连续型变量,TSP 是离散型问题,所以本文通过引入 2-opt 算法将趋化算子进行离散化。为了便于描述,首先给出以下定义:

定义 1　$C = \{1,2,\cdots,n\}$ 为所有城市编号的集合。

定义 2　两个细菌所在区域之间的距离是其对应区域编码之间的 Hamming 距离。

设某细菌的区域编码为 $C_1,\cdots,C_iC_{i+1},\cdots,C_jC_{j+1},\cdots,C_n$,通过随机选择一个城市 C_i 来模拟细菌在翻转过程中任意选择的方向,同时,为了避免方向的重复选择,设置一个列表 visited 存储细菌已经探测过的方向。确定方向 C_i 之后,在集合 C-visited 中选择与 C_i 不相邻的另一点 C_j 执行 2-opt 操作,直到适应值得到改善或者不存在满足条件的点 C_j,此过程模拟了细菌在该方向上的移动。同时,根据定义 2 可知,细菌移动的单位步长为 2。

若翻转后适应值得到改善,则清空列表 visited,然后细菌将保持方向 C_i 继续向前移动,这模拟了细菌的游动; 否则,细菌继续翻转,选择新的方向 C_i。细菌每进行一次翻转或游动,其尝试次数加 1,直到达到趋化算子次数 Nc,跳转到下一个细菌执行趋化算子。

下面以 7 城市 TSP 为例对趋化算子进行讲解($Nc = 3$),表 2 给出了 7 个城市的横坐标和纵坐标。

表 2　7 城市 TSP 坐标表

city	横坐标	纵坐标
1	5	10
2	7	7
3	7	13
4	11	20
5	13	4
6	15	6
7	21	12

假设某细菌的区域编码为 $(1,2,3,4,5,6,7)$,所对应的初始路径如图 1 所示,路径长度为 61.23。

若 $C_i = 4$,visited $= \{4\}$,C_j 取值范围是 $\{1,2,6,7\}$。当 $C_j = 7$ 时,$d(4,5) + d(7,1) > d(4,7) + d(5,1)$,经过 2-opt 操作后适应值得到改善,区域编码变成 $(1,2,3,4,7,6,5)$,图 2 给出了经过一次调整后所对应的路径,路径长度为 48.51。沿着 $C_i = 4$ 方向继续移动,visited $= \{4\}$,C_j 取值范围是 $\{1,2,5,6\}$。循环判定后发现无法改善适应值,则重新选择一个方向。若 $C_i = 5$,visited $= \{4,5\}$,即 C_j 取值范围是 $\{2,3,7\}$。当 $C_j = 2$ 时,$d(5,1) + d(2,3) > d(5,2) + d(1,3)$,经过 2-opt 操作后适应值

将得到改善，区域编码变成(1,3,4,7,6,5,2)，图 3 给出了经过二次调整后所对应的路径，路径长度为 46.10。

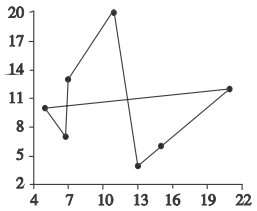
图1 初始路径

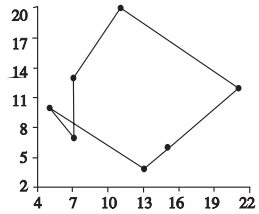
图2 经过一次调整后的路径图

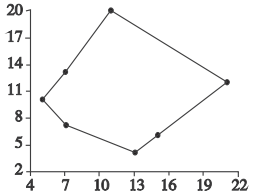
图3 经过二次调整后的路径

若某细菌已经探测完所有的方向，即 C-visited = ∅，则其适应值无法再通过 2-opt 操作改善，相当于该细菌已经达到了某个局部最优区域。在细菌觅食过程中，若细菌陷入某一区域不再移动，则区域内食物会随着时间推移而不断消耗。本文通过插入算子模拟了这一现象，帮助算法跳出局部最优点。

定义 3 插入算子定义为如下操作：假设某细菌的区域编码为 P，随机选择两点 C_r 和 C_r'，将 C_r' 插入到 C_r 的前面，同时修改其适应值，然后清空列表 visited。图 4 给出了插入算子的操作示意图。

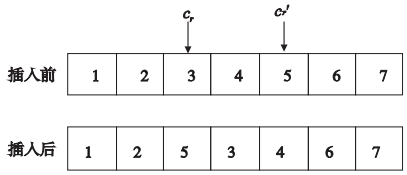

图4 插入算子操作示意图

2.3 改进的迁徙算子

标准细菌觅食算法中的迁徙算子是随机产生一个新个体并替换掉原个体。笔者通过实验分析发现，随机生成的新个体适应值通常很低，在后续的趋化算子中将对其进行多次路径调整，而这一部分的时间开销很大。文献[9]提出，利用最小生成树构建 TSP 的基因库来指导种群的进化方向，能够加快算法的收敛速度和寻优能力。本文受这一观点的启发，对包含 n 个城市的 TSP，利用普利姆算法求解得到一棵最小生成树(MST)，并将这棵 MST 中的每一条边(C_i,C_j)对应地存储在一个 $n\times(n-1)$ 的矩阵 M 中，矩阵 M 称为基因库。由于一个 TSP 能生成多棵 MST，操作可以重复多次，这样，基因库中的基因片段就很多，增强了细菌群体的全局性。

本文对迁徙算子作如下改进：当某细菌满足迁徙概率时，新产生细菌的区域编码总是优先选择存在于基因库中的边，如果基因库中没有候选边，则随机生成新边。实验结果表明，这种策略有效提高了算法的搜索效率。

2.4 算法描述

综上所述，本文提出的求解 TSP 的 DBFO 算法可描述如下：

a) 参数初始化。种群规模为 S，趋化算子次数为 Nc，复制算子次数为 Nre，迁徙算子次数为 Ned，迁徙概率为 Ped。

b) 细菌位置初始化。计算细菌的初始适应值。设 $P(i,j,k,l)$ 表示细菌 i 的区域编码，visited $[i]$ = ∅，其中：j 为第 j 代趋化循环，k 为第 k 代复制循环，l 为第 l 代迁徙循环。

c) 迁徙循环 $l=1:Ned$。

d) 复制循环 $k=1:Nre$。

e) 趋化循环 $j=1:Nc$，对各细菌个体执行趋化算子。

f) 如果 $j<Nc$，转 e)。

g) 复制算子。

h) 如果 $k<Nre$，转 d)。

i) 迁徙算子。

j) 若 $l<Ned$，转 c)，否则整个算法结束，输出结果。

3 仿真实验及分析

为了更好地说明算法的有效性和正确性，选用了 TSPLIB 标准库中的 22 个实例(规模从 14 ~ 442)进行测试。本文用 Java 语言编写程序实现所提出的算法，并在一台配置为 Intel® Core TM i7-3770 CPU @ 3.40 GHz 的 PC 上运行该程序进行仿真实验。

在计算机仿真实验中，主要参数设置为：趋化算子次数 Nc 为 50；复制算子次数 Nre 为 20；迁徙算子次数 Ned 为 15；迁徙概率 Ped 为 0.6；城市规模小于等于 150 的实例种群规模 S 为 50；城市规模大于 150 的实例种群规模 S 为 100。

3.1 DBFO 算法的收敛性能分析

由于 eil51 实例被广泛应用于各种算法的测试，首先以 eil51 实例为例对 DBFO 算法的收敛性能进行分析。图 5 给出了 DBFO 算法求解 eil51 实例的收敛曲线，图 6 给出了 DBFO 算法求解 eil51 实例的最优路径。

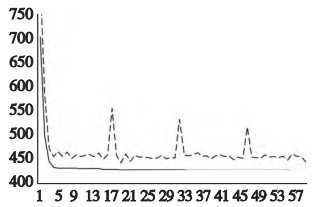
图5 DBFO求解eil51实例的收敛曲线

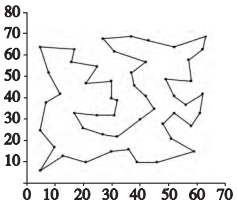
图6 DBFO算法求解eil51实例得到的最优路径

图 5 中，实线代表最短路径长度，虚线代表种群平均路径长度，横坐标代表复制算子执行次数。可以看出，算法前期几乎是垂直收敛的，大约经历了 4 次复制算子后最短路径长度就下降到了比较低的水平。种群平均路径长度反映了细菌种群的多样性，其收敛曲线在下降到较低水平后仍然存在不停的扰动，并且会间隔地出现大幅度的震荡。不难发现，大幅度震荡是由于迁徙算子引入了全新个体导致的，而产生扰动是因为插入算子随机地对原个体进行破坏而导致个体适应值下降。插入算子和迁移算子共同维持了种群的多样性，使得 DBFO 算法可以有效地跳出局部最优，进一步提高收敛精度。图 5 中，DBFO 算法约经历 60 次复制算子后找到了目前非整数路径的最优解 428.8717564。

3.2 DBFO 算法和 ACALA 算法的比较

表 3 是 DBFO 算法与文献[13]的 ACALA 算法求解 14 个 TSP 实例的实验结果比较，括号内的值是已知非整数路径的最优解，两种算法在各实例上连续运行 10 次。

表 3　DBFO 算法与 ACALA 算法求解 TSP 的结果比较

TSP	TSPLIB 已知最优解	方法	最优解	平均解
att48	10628	ACALA	33523.71	33523.71
	(33523.71)[13]	DBFO	33523.71	33523.71
eil51	426	ACALA	428.87	430.15
	(428.8718)[13]	DBFO	428.87	428.93
st70	675	ACALA	677.11	679.45
	(678.5975)[13]	DBFO	677.11	677.11
eil76	538	ACALA	544.37	550.04
	(544.37)[13]	DBFO	544.37	544.47
gr96	55209	ACALA	510.89	512.39
	(512.31)[13]	DBFO	510.89	511.14
kroA100	21282	ACALA	21285.44	21289.84
	(21285.44)[13]	DBFO	21285.44	21286.34
kroD100	21294	ACALA	21294.29	21333.03
	(21294.3)[13]	DBFO	21294.29	21316.50
rd100	7910	ACALA	7910.3962	7917.1814
	(7910.4)[13]	DBFO	7910.3962	7914.4870
eil101	629	ACALA	640.21	644.92
	(642.3)[13]	DBFO	640.21	642.67
gr120	6942	ACALA	1615.81	1622.01
	(1666.5)[13]	DBFO	1610.31	1619.39
d198	15780	ACALA	15825.53	15847.12
	(15808.65)[13]	DBFO	15811.34	15840.05
gr202	40160	ACALA	487.53	488.74
	(549.99)[13]	DBFO	487.10	489.43
tsp225	3916	ACALA	3895.53	3925.08
	(3859)[13]	DBFO	3890.65	3925.64
pcb442	50778	ACALA	5118.0	5193.0
	(5078.35)[13]	DBFO	5172.64	5233.32

观察表 3 数据,除了在 att48 实例上两种算法一样,每次运行时都能找到问题的最优解,在 gr120、d198、gr202、tsp225 这 4 个实例上 DBFO 算法比 ACALA 算法找到了更优的解,而在 eil51、st70、eil76、gr96、korA100、kroD100、rd100、eil101 这 8 个实例上虽然两种算法找到的最优解相同,但是 DBFO 算法得到的平均解优于 ACALA 算法。特别地,DBFO 算法在 tsp225 实例上得到的最优解为 3890.65,比目前 TSPLIB 标准库提供的最优路径长度 3916 小 25.35。图 7 是 DBFO 算法求解 tsp225 实例得到的最优路径。

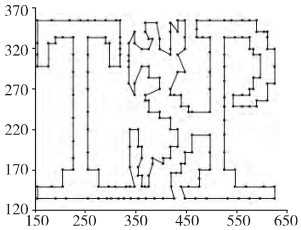

图7　DBFO算法求解tsp225实例得到的最优路径图

3.3　DBFO 算法和 DGSO 算法的比较

表 4 是 DBFO 算法与文献[14]的 DGSO 算法求解 10 个 TSP 实例的实验结果比较,"—"表示文献[14]没有提供该项数据,两种算法在各实例上连续运行 20 次。

观察表 4 数据,在 burma14、bays29、oliver30、att48、eil51、pr76、kroB100 这 7 个实例上两种算法找到的最优解相同,而在 ch130、kroB150、kroB200 这 3 个实例上 DBFO 算法比 DGSO 算法找到了更优的解。在 burma14 和 oliver30 这两个实例上两种算法一样,每次运行时都能找到问题的最优解,在 eil51 实例上 DBFO 算法得到的平均解优于 DGSO 算法。特别地,DBFO 算法在 kroB100、kroB150 两个实例上找到的最优解分别为 22139.07 和 26127.36,比目前 TSPLIB 标准库提供的最优路径长度分别小 1.93 和 2.64。图 8、9 分别是 DBFO 算法求解 kroB100、kroB150 这两个实例得到的最优路径图。

表 4　DBFO 算法与 DGSO 算法求解 TSP 的结果比较

TSP	TSPLIB 已知最优解	方法	最优解	平均解
burma14	3323	DGSO	30.8785	30.8785
	(30.8785)[14]	DBFO	30.8785	30.8785
bays29	2020	DGSO	9074.15	—
	(9291.35)[14]	DBFO	9074.15	9074.15
oliver30	420	DGSO	423.7406	423.7406
	(423.7406)[14]	DBFO	423.7406	423.7406
att48	10628	DGSO	33523.71	—
	(33523.71)[14]	DBFO	33523.71	33523.71
eil51	426	DGSO	428.8718	429.4730
	(428.8718)[14]	DBFO	428.8718	428.9893
pr76	108159	DGSO	108159.44	—
	(108159.44)[14]	DBFO	108159.44	108275.20
kroB100	22141	DGSO	22139.07	—
	(—)	DBFO	22139.07	22150.32
ch130	6110	DGSO	6125.07	—
	(6110.86)[14]	DBFO	6110.72	6144.87
kroB150	26130	DGSO	26206.69	—
	(—)	DBFO	26127.36	26197.05
kroB200	29437	DGSO	29605.13	—
	(—)	DBFO	29441.38	29737.53

对上述实验结果分析可知,DBFO 算法具有较快的收敛速度,并且能够维持解的多样性,有效克服停滞行为的过早出现。在求解城市规模 150 以内的实例中,DBFO 算法均能够搜索到目前已知的最优解,在城市规模大于 150 的实例中,虽然 DBFO 算法未必能搜索到已知最优解,但其所得到的最优解与已知最优解的误差最大(pcb442 实例)也仅为 1.87%。由此可见,DBFO 算法具有较好的鲁棒性,可有效求解 TSP。同时,与文献[13]的 ACALA 算法以及文献[14]的 DGSO 算法相比较,DBFO 算法求解结果更好,具有更强的稳定性。

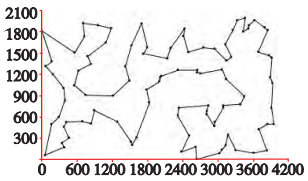

图8　DBFO算法求解kroB100实例得到的最优路径

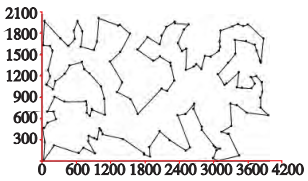

图9　DBFO算法求解kroB150实例得到的最优路径

4　结束语

细菌觅食算法思想直观易理解,在已有应用中都表现出强鲁棒性、高收敛速度和高精度等优点。本文针对求解组合优化领域中著名的 TSP,提出一种离散型细菌觅食算法。该算法结合 2-opt 算法对趋化算子进行改进,使其适合处理离散型变量。同时,为了提高搜索效率,根据最小生成树构建 TSP 的基因库,在迁移算子中引入启发信息来优化新生成的个体。通过仿真实验表明,该算法能够克服早熟现象,具有较快的收敛速度和较强的稳定性,对求解城市规模在 500 以下的 TSP 非常奏效,与其他典型的智能优化算法相比具有竞争力。今后将进一步研究细菌觅食算法在不同模型、不同参数情况下的优化效果以及细菌觅食算法在其他优化领域的应用。

参考文献:

[1]　LIN S, KERNIGHAN B W. An effective heuristic algorithm for the traveling salesman problem [J]. Operation Research, 1973, 21(2): 498-516.

(下转第 3650 页)

4 结束语

基于不确定数据的决策树算法实验中，雨量值无法较好地获取，致使定量刻画降雨值存在技术难题，因此引入不确定数据，建立基于不确定决策树算法的滑坡危险性评价模型。评价结果与实际情况基本吻合，预测精确度达到要求，因此评价方法合理有效。

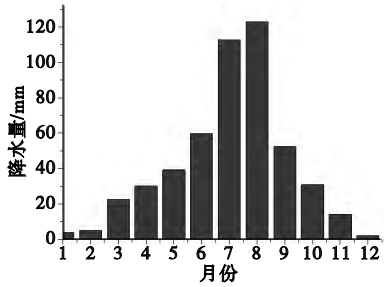

图2 延安市宝塔区月平均雨量统计图

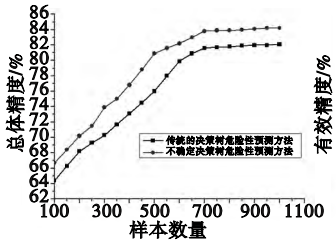

图3 两种方法的总体精度比较

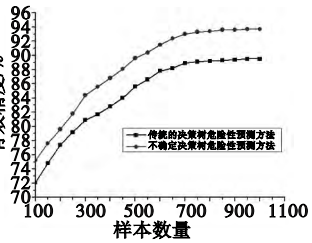

图4 两种方法的有效精度比较

评价属性的选择以及属性值的划分非常重要。本例实验中划分充分地结合了研究区当地的实际情况，提出了较好的划分标准，得到了较高的总体精度和有效精度。但本例实验中未对其余属性进行分析和划分，因此，高效地选择评价属性和划分属性值是接下来需要研究的重点。

参考文献:

[1] ZHANG M，LIU Jie. Controlling factors of loess landslide in western China [J]. Environmental Earth Sciences，2010，59(3)：1671-1680.

[2] HAN Jia-wei，KAMBER M. 数据挖掘概念与技术 [M]. 范明，孟小峰，译. 北京：机械工业出版社，2001.

[3] 赵建华，陈汉林. 基于决策树算法的滑坡危险性区划评价 [J]. 浙江大学学报：自然科学版，2004，31(4)：465-470.

[4] WANG Xian-min，NIU Rui-qing. Landslide intelligent prediction using object-oriented method [J]. Soil Dynamics and Earthquake Engineering，2011，19(4)：223-232.

[5] 亓呈明，郝玲，崔守梅. 一种新的模糊决策树模型及其应用 [J]. 山东大学学报：自然科学版，2007，42(11)：107-113.

[6] YEON Y K，HAN J，GLAND J G. Landslide susceptibility mapping in Injae，Korea，using a decision tree [J]. Engineering Geology，2010，116(2010)：274-283.

[7] CHU Chien-min. Integrating decision tree and spatial cluster analysis for landslide susceptibility zonation [J]. World Academy of Science，Engineering and Technology，2009，59(9)：479-483.

[8] NEFESLIOGLU H A. Assessment of landslide susceptibility by decision trees in the metropolitan area of Istanbul，Turkey [J]. Mathematical Problems in Engineering，2010，90109(5)：15-23.

[9] QIN Biao，XIA Yu-ni，WANG Shan，*et al*. A novel Bayesian classification for uncertain data [J]. Knowledge-Based Systems，2010，24：1151-1158.

[10] 孟飞翔，帅立国，姜昌金. 决策树在客户价值分析中的应用 [J]. 计算机技术与发展，2007，17(4)：60-63.

[11] 张茂省. 延安宝塔区滑坡崩塌地质灾害 [M]. 北京：地质出版社，2008.

[12] QIN Biao，XIA Yu-ni，LI Fang. DTU：a decision tree for uncertain data [C] //Lecture Notes in Computer Science，vol 5476. Berlin：Springer，2009：4-15.

[13] 陈良维. 决策树算法在农户小额贷款中的应用研究 [J]. 计算机工程与应用，2008，44(31)：242-248.

[14] 吴树仁，石菊松，张春山. 滑坡风险评估理论与技术 [M]. 北京：科学出版社，2012.

[15] 辛彭，吴树仁，石菊松. 基于降雨响应的黄土丘陵区滑坡危险性预测研究 [J]. 地球学报，2012，33(3)：349-359.

[16] 王卫东，陈燕平，钟晟. 应用 CF 和 Logistic 回归模型编制滑坡危险性区划图 [J]. 中南大学学报：自然科学版，2009，40(4)：1127-1132.

[17] 陈冠，孟兴民，郭鹏. 白龙江流域基于 GIS 与信息量模型的滑坡危险性等级区划 [J]. 兰州大学学报：自然科学版，2011，47(6)：1-6.

(上接第 3645 页)

[2] PASSINO K M. Biomimicry of bacteria foraging for distributed optimization and control [J]. IEEE Control Systems Magazine，2002，22(3)：52-67.

[3] LIU Y，PASSINO K M. Biomimicry of social foraging bacteria for distributed optimization：models，principles，and emergent behaviors [J]. Journal Optimization Theory Application，2002，115(3)：603-628.

[4] 李明. 模拟细菌菌落进化过程的群体智能算法 [J]. 系统仿真学报，2013，25(2)：251-255.

[5] 李杰，彭月英，元昌安，等. 基于细菌觅食优化算法的自适应阈值边缘检测 [J]. 计算机工程，2012，38(21)：178-181.

[6] 黄伟锋，林卫星，范怀科，等. 细菌觅食优化的智能 PID 控制 [J]. 计算机工程与应用，2011，47(21)：82-85.

[7] 李岳洪，万频，王永华，等. 改进的细菌觅食算法求解认知无线网络频谱分配问题 [J]. 计算机科学，2013，40(8)：49-52.

[8] 崔静静，孙延明，车兰秀. 改进细菌觅食算法求解车间作业调度问题 [J]. 计算机应用研究，2011，28(9)：3324-3326.

[9] 杨辉，康立山，陈毓屏. 一种基于构建基因库求解 TSP 问题的遗传算法 [J]. 计算机学报，2003，26(12)：1753-1758.

[10] 周雅兰. 细菌觅食优化算法的研究与应用 [J]. 计算机工程与应用，2010，46(20)：16-21.

[11] HELSGAUN K. An effective implementation of the Lin-Kernighan traveling salesman heuristic [J]. European Journal of Operation Research，2000，12(1)：106-130.

[12] 王凌. 智能优化算法及其应用 [M]. 北京：清华大学出版社，2001.

[13] 刘朝华，张英杰，章兢，等. 蚁群算法与免疫算法的融合及其在 TSP 中的应用 [J]. 控制与决策，2010，25(5)：695-700.

[14] 周永权，黄正新，刘洪霞. 求解 TSP 问题的离散型萤火虫群优化算法 [J]. 电子学报，2012，40(6)：1164-1170.