Journal of Computer Applications ISSN 1001-9081 2015-12-15
计算机应用,2015,35(S2):114-117 CODEN JYIIDU http://www.joca.cn

文章编号:1001-9081(2015)S2-0114-04

# 面向旅行商问题的蚁群算法改进

姜坤霖,李美安*,张宏伟

(内蒙古农业大学 计算机信息与工程学院,呼和浩特 010018)

(* 通信作者电子邮箱 Limeian1973@126.com)

**摘 要:** 针对基本蚁群算法在处理旅行商问题(TSP)时会出现收敛速度慢且容易陷入局部最优解的缺陷,从城市选择策略和信息素挥发系数进行了改进,提出了一种基于赌盘算法的城市选择策略和挥发系数自适应的蚁群算法,并采用了任务提前终止策略来减少算法的运行时间。仿真结果表明,该算法与基本蚁群算法相比,收敛时间比基本蚁群算法运行时间缩短了60%~80%,改进后的算法最优解绝大部分优于基本蚁群算法,也有少部分不如基本蚁群算法,但都在可接受范围以内。

**关键词:** 蚁群算法;动态自适应;赌盘算法;TSP

**中图分类号:** TP311.5 **文献标志码:** A

## Improved ant colony algorithm for travelling salesman problem

JIANG Kunlin, LI Mei'an*, ZHANG Hongwei

(*College of Computer and Information Engineering, Inner Mongolia Agricultural University, Hohhot Inner Mongolia* 010018, *China*)

**Abstract:** The basic Ant Colony Algorithm (ACA) has some problems when processing Traveling Salesman Problems (TSP), such as slow convergence speed and easy to fall into local optimum. In order to solve these problems, this paper improved the city selection strategy based on roulette wheel algorithm and proposed an improved ant algorithm based on adaptive volatile coefficient. The run-time was reduced by using the strategy early to terminate the task. The simulations reveal that the proposed algorithm has some advantages compared to basic ACA. Its run-time is 60% to 80% less than basic ACA, and the most of optimization of improved algorithm is superior to that of basic ACA. Of course, it also have some problems inferior to basic ACA but within acceptable range.

**Key words:** Ant Colony Algorithm (ACA); dynamic adaptive; roulette wheel algorithm; Travelling Salesman Problem (TSP)

## 0 引言

旅行商问题,即TSP(Traveling Salesman Problem)问题[1]。假设有位商人要访问某些个城市而且每个城市必须访问一次不能重复访问,最后还要回到原来出发的城市。目标是在众多路径中选择一条所有路径之和最短的路径。该问题一经提出,便受到专家学者们广泛关注,他们利用组合优化提出了很多启发式搜索算法,如模拟退火算法[2]、遗传算法[3]、禁忌搜索算法[4-5]、人工神经网路[6]、蚁群算法[7]、免疫进化算法[8]等。在这些常用的算法中,蚁群算法是比较成熟的,能够有效地解决TSP。蚁群算法是一种仿生学算法[9],在1991年由Drigo等提出[10-11]。蚁群算法来自蚁群的行为,在蚁群寻找食物的过程中,走在前面的蚂蚁会释放信息素,走在后面的蚂蚁根据信息素的浓度选择路径,同时也释放信息素。经过大量蚂蚁的反复行走,一条高浓度信息素的路径便会出现,该路径便是蚂蚁蚁穴到食物源的最短路径。蚁群算法易与其他方法结合,有很强的鲁棒性,但与此同时它运行时间长,易陷入局部最优。

在本文中,提出了一种改进的蚁群算法来解决传统蚁群算法存在的易陷入局部最优和运行时间长的问题。该算法的基本思想是,采用的任务提前终止策略来缩短算法的运行时间;为了避免陷入局部最优,在对下一城市选择的问题上,采用了赌盘算法与其融合;为了得到更好的路径,对信息素挥发系数也进行了改进。仿真结果表明,与传统的蚁群算法相比,它的运行时间大幅缩短,并且大部分路径能到得到更优的路径。

## 1 基本蚁群算法

一群蚂蚁从蚁穴出发去寻找某一食物源,经过一段时间后,它们便能够寻找到一条固定的最短路径。经研究发现,蚂蚁在寻找食物的过程中,会在其走过的路线上留下一种被称为信息素的物质,其他蚂蚁能够感知到这种物质并以此作为它们的前进方向。这种信息素可以叠加,该路径上走过的蚂蚁数量越多,该路径上的信息素就越多;同时,信息素也会挥发,随着时间的推移,之前留在路径上的信息素浓度会降低。经过一段时间后,一条具有高浓度信息素的路径便会形成,该路径便是从蚁穴到达食物源的最短路径。

针对于求解TSP,算法步骤如下:

1) 信息素初始化。

$$\tau_{ij}(0) = m/C_{\min} \tag{1}$$

其中:$\tau_{ij}(0)$ 表示初始时城市 $i$ 到城市 $j$ 路径上的信息素;$m$ 为蚂蚁的数量;$C_{\min}$ 为所有城市中距离最短的两个城市的距离。

2) 状态转移。

每只蚂蚁 $k$ 在 $t$ 时刻通过式(2)计算出由城市 $i$ 向城市 $j$ 转移的概率:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(ij) \times \eta_{ij}^{\beta}}{\sum\limits_{s \in allowed_k} \tau_{ij}^{\alpha}(ij) \times \eta_{ij}^{\beta}}, & j \in allowed_k \\ 0, & \text{其他} \end{cases} \tag{2}$$

收稿日期:2015-01-23;修回日期:2015-03-11。

作者简介:姜坤霖(1990-),男,辽宁鞍山人,主要研究方向:软件工程、智能信息系统、仿生学算法;李美安(1973-),男,四川达州人,教授,主要研究方向:先进计算;张宏伟(1989-),男,内蒙古通辽人,主要研究方向:软件工程、智能信息系统。

其中: $\tau_{i,j}(t)$ 为 $t$ 时刻在路径$(i,j)$上存在的信息素的浓度; $\eta_{ij}(t)=1/d_{ij}$ 蚂蚁从城市$i$到城市$j$的期望程度; $\alpha$为信息启发因子,即轨迹的相对重要性; $\beta$ 为期望启发因子,即能见度的相对重要性。$allowed=\{1,2,\cdots,n\}-tabu_k$,记录蚂蚁下一次可以访问的城市,$tabu_k$ 为蚂蚁 $k$ 所走过的城市。

3) 信息素的更新。

蚂蚁在寻找食物的过程中,会在所走的路径上释放信息素,根据式(3)对信息素进行局部更新。

$$\tau_{ij}(t+1)=(1-\xi)\tau_{ij}(t)+\xi\tau_0 \tag{3}$$

通常设定 $\xi=0.1$,$\xi\in(0,1)$。

为了防止过了一段时间之后,残留路径上的信息素过多,从而使启发信息变得不再重要,在蚂蚁行走固定时间或者固定长度或者到达食物源后,要对整个路径上的信息素进行更新。在 $t+n$ 时刻 $d(i,j)$ 按照下面公式进行更新。

$$\tau_{ij}(t+n)=(1-\rho)\tau_{ij}(t)+\rho\cdot\Delta\tau_{ij}(t) \tag{4}$$

$$\tau_{ij}(t)=\sum_{k=1}^{m}\tau_{ij}^{k}(t) \tag{5}$$

其中: $\rho$ 表示信息素挥发系数,$\Delta\tau_{ij}(t)$ 表示当前路径城市 $i$ 到城市$j$的信息素增量,$\tau_{ij}^{k}(t)$ 表示第 $k$ 只蚂蚁在路径$(i,j)$上信息素的浓度。

信息素更新策略满足:

$$\tau_{ij}^{k}(t)=\begin{cases}Q/L, & \text{若第 } k \text{ 只蚂蚁在本次循环中经过 } ij\\ 0, & \text{其他}\end{cases} \tag{6}$$

其中: $Q$ 表示信息素强度增加的系数; $L$ 表示第 $k$ 只蚂蚁在本次循环中所走路径的长度。

## 2　基本蚁群算法求解 TSP

基本蚁群算法求解 TSP 流程如图 1 所示。

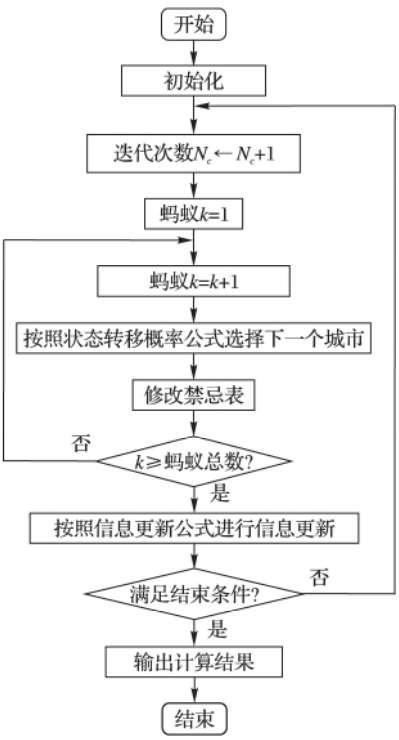


图 1　蚁群算法求解 TSP 流程

## 3　改进后蚁群算法求解 TSP 原理

由于基本蚁群算法存在易陷入局部最优、运行时间长的问题,本文提出了 3 个改进方法来避免陷入局部最优、运行时间长的缺点。

### 3.1 任务提前终止策略

蚁群算法存在易陷入局部最优的缺点,当设置的最大周期比较大时,会出现这种情况: 迭代到一定次数后,无论再迭代多少次,得到的解决方案都与之前得到的最优解相同。本文提出了一种任务提前终止策略,即在当前迭代次数中,出现了 $N$ 次距离相等的并且是最优路径,那就停止迭代; 若出现了 $N$ 次相同的解决方案,但是 $N$ 次解决方案并不是当前的最优解决方案,那么继续迭代,直到出现另一条满足 $N$ 次距离相等的路径,且该路径是最优的情况为止,否则迭代至最大周期。在迭代的过程中,为了防止陷入到局部最优,下一次迭代得到的最优路径与它前一次迭代的最优路径不同。

### 3.2　赌盘算法

在基本蚁群算法中,蚂蚁 $k$ 在城市 $i$ 时根据式(2)选择下一个要遍历的城市,要遍历的城市是式(2)求得的最大值。但研究发现,蚂蚁 $k$ 要遍历的下一个城市未必就是求得概率的最大的城市,而是概率比较大的集合中的一个。本文在基本蚁群算法执行完式(2)后,加入了赌盘算法。

首先,计算出蚂蚁 $k$ 在城市 $i$ 上要遍历的下一个 $M$ 个最大转移概率的城市,将它们按照概率的大小分别分配到 [0,1] 的转盘上相应区域,然后生成一个 [0,1] 的随机数,根据该随机数在转盘上的区域选择下一个要遍历的城市。

本文选择 3 个最大转移概率的城市。假设它们的概率分别为 $a$、$b$、$c$,则选择与 $a$ 概率对应的城市的概率为 $a/(a+b+c)$,选择与 $b$ 概率对应的城市的概率为 $b/(a+b+c)$,选择与 $c$ 概率对应的城市的概率为 $c/(a+b+c)$。它们在转盘上所对应的区域分别为 $[0,a/(a+b+c)]$、$[a/(a+b+c),(a+b)/(a+b+c)]$、$[(a+b)/(a+b+c),1]$,生成的随机数在哪个区域就选择哪个城市,这样可以大大地降低算法陷入局部最优解的概率。

### 3.3　信息素挥发系数的改进

信息素挥发系数 $\rho$ 能对基本蚁群算法的搜索能力和收敛速度造成直接影响。在基本蚁群算法中,$\rho$ 是定值,若 $\rho$ 的值过大,会造成从来没有被搜索到的路径上的信息素趋近于 0,同时信息素挥发过快,遗留的信息素也就对算法造成不了太大影响; 若 $\rho$ 的值过小,会造成每条路径上的信息量挥发过慢,搜索时间更长。本文给 $\rho$ 赋一个较大的初始值,每次迭代完成之后动态地减小 $\rho$ 的值。$\rho$ 的值不断地减小能够有效避免局部收敛。信息素挥发系数 $\rho$ 采用下面公式进行自适应调整。

$$\rho(t)=\begin{cases}0.95\rho(t-1), & 0.95\rho(t-1)\geqslant\rho_{\min}\\ \rho_{\min}, & \text{其他}\end{cases} \tag{7}$$

其中 $\rho_{\min}$ 为 $\rho$ 的最小值。

### 3.4　改进后蚁群算法的实现

改进后蚁群算法步骤如下:

1) 参数初始化。表 1 为所有参数的初始值。

表 1　参数赋初始值

| 序号 | 参数 | 值 |
|---|---|---|
| 1 | 时间 | $t=0$ |
| 2 | 循环次数 | $N_c=0$ |
| 3 | 信息素初始值 | $\tau_{ij}(t)=$常量 |
| 4 | 信息素增量 | $\Delta\tau_{ij}(t)=$常量 |
| 5 | 最大周期 | $N_{\max}=$常量 |

2) 将 $m$ 只蚂蚁随机分配到 $n$ 个城市中,蚂蚁根据式(2)算出概率最大的 3 个城市,然后采用赌盘算法选取其中一个城市 $j$,$j$ 属于 $C-tabu_k$,$C$ 为所有城市的集合。

3) 修改禁忌表。将步骤 2) 中的城市 $j$ 放入到禁忌表中。

4) 重复步骤 2) 和 3),直到该蚂蚁走完所有城市。

5) 信息素更新。根据式(3) ~ (5) 更新每个路径上的信息素。

6) 判断当前迭代有没有得到 $N$ 次相同的最优解,若没得到跳转到步骤 7),若得到则输出结果。

7) 若满足迭代结束条件 $N_c \geqslant N_{max}$,结束循环并输出结果;若不满足,进行信息素挥发系数更新。根据式(7) 得到 $\rho$,并清空禁忌表进行下一循环。

## 4 仿真实验结果及分析

本文的算法仿真实验在 CPU 为 Pentium(R) Pual-Core,型号为 E5200 @2.50 GHz 环境下使用 MyEclipse Version:8.5 运行的。采用 MATLAB Version7.0.0.19920(R14) 进行绘图。本文算法的最大迭代次数是 100,$m = 100$,$\rho = 0.95$,$\alpha = 1$,$\beta = 2$。从 TSPLIB[13] 中选取了 25 个使用欧氏距离的 TSP 实例进行测试,每个实例运行 5 次。

由图 2 可知,改进后的蚁群算法运行时间比基本蚁群算法要少,图 3 展示了改进后蚁群算法的运行时间降低的百分比。改进后蚁群算法跟基本蚁群算法相比,效率至少提高了 60%,最多可达到 84%。本文实验又对比了基本蚁群算法和改进蚁群算法的路径最优解,具体如表 2 所示。

为了更形象地说明实验结果,挑选了路径在 10 000 ~ 70 000的 7 个城市节点库来对比基本蚁群算法最优解、改进后蚁群算法最优解及国际最优解,如图 4 所示。

由上述图表可以看出,与基本蚁群算法相比,改进后的蚁群算法的运行效率明显提高;大多数改进后的蚁群算法得到最优解比基本蚁群算法得到的更优,由于城市间的不确定性,也有少部分得到的最优解不如基本蚁群算法,但误差都在可接受的范围之内,牺牲空间换时间也是值得的。

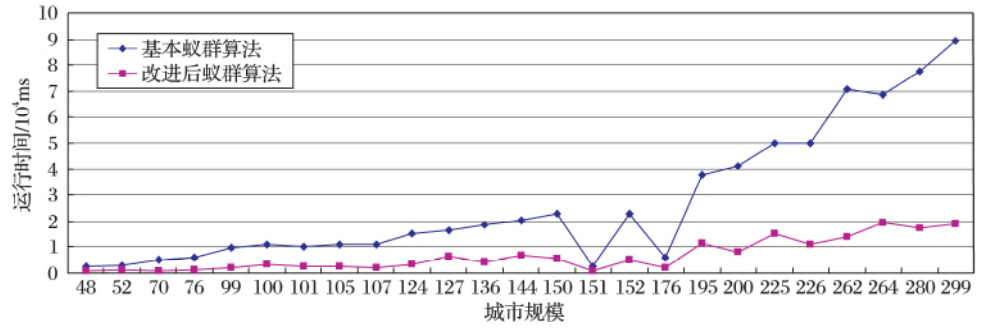


图 2 基本蚁群算法与改进后的算法运行时间对比

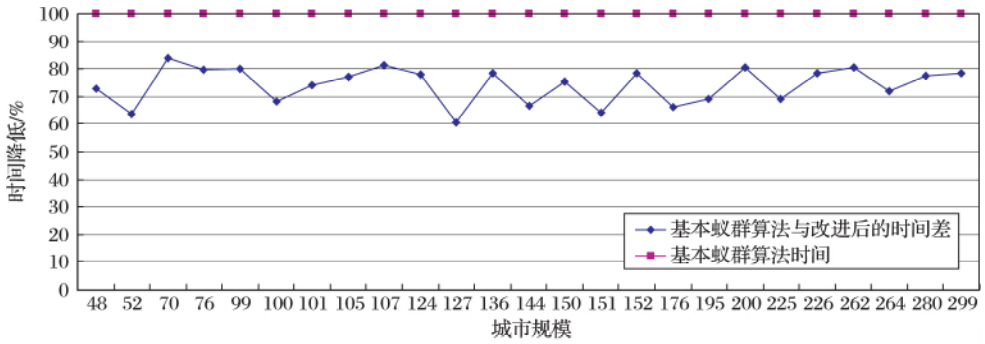


图 3 蚁群算法改进后时间降低百分比

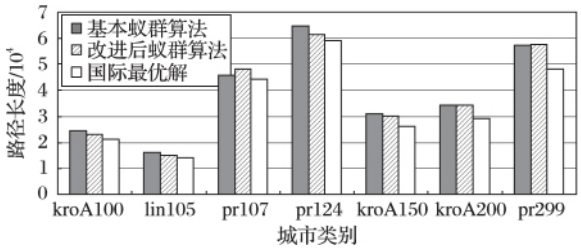


图 4 两种算法与国际最优解路径长度对比

下面以 berlin52 为例,图 5 ~ 7 分别描述了基本蚁群算法、改进后算法、国际最优的解决方案。

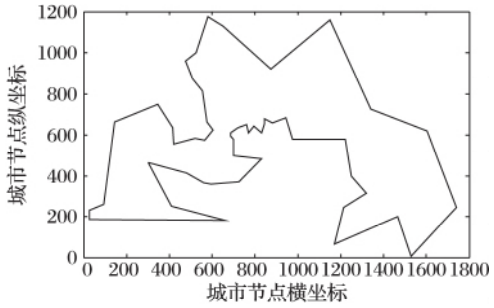


图 5 基本蚁群算法求解 berlin52 最优路径

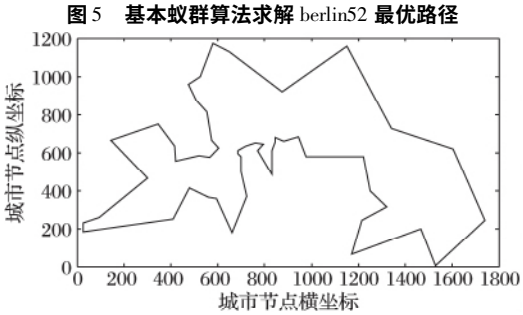


图 6 改进后蚁群算法求解 berlin52 最优路径

## 5 结语

本文研究了面向 TSP 的蚁群算法,解释了 TSP 和蚁群算法的基本原理,分析了基本蚁群算法解决 TSP 的缺点,并提出了相应的解决方案。实验结果表明,使用文中提出的3种解决方案后,算法的运行效率有了显著提升,大部分得到的最优解决方案比基本蚁群算法得到的最优解决方案更优,然而由于城市的之间的不确定性以及参数造成的影响,也有一些解不如基本蚁群算法,但是都相差不大或在可接受的范围之

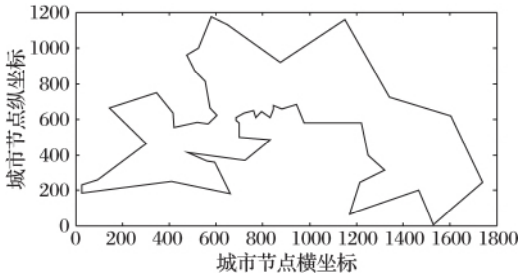


图 7　berlin52 国际最优路径

表 2　基本蚁群算法与改进后蚁群算法路径长度对比

| TSP 实例 | 基本蚁群算法 | | 改进后蚁群算法 | | 国际 |
|---|---|---|---|---|---|
| | 最优解 | 最差解 | 最优解 | 最差解 | 最优解 |
| att48 | 36 649 | 37 082 | 35 185 | 35 721 | 2 879 |
| berlin52 | 8 092 | 8 092 | 7 662 | 7 829 | 7 542 |
| st70 | 778 | 786 | 733 | 768 | 675 |
| pr76 | 126 212 | 127 369 | 118 892 | 122 651 | 108 159 |
| rat99 | 1 433 | 1 453 | 1 339 | 1 374 | 1 211 |
| kraA100 | 24 524 | 24 698 | 23 190 | 23 768 | 21 828 |
| eil101 | 742 | 762 | 706 | 741 | 629 |
| lin105 | 16 460 | 16 718 | 15 126 | 16 217 | 14 379 |
| pr107 | 45 658 | 46 133 | 48 040 | 49 411 | 44 303 |
| pr124 | 64 524 | 65 500 | 61 430 | 63 915 | 59 030 |
| bier127 | 129 318 | 133 953 | 125 245 | 12 911 | 118 282 |
| pr136 | 111 181 | 112 420 | 113 993 | 115 647 | 96 772 |
| pr144 | 60 954 | 60 954 | 60 370 | 62 272 | 58 537 |
| kroA150 | 31 073 | 31 170 | 30 312 | 30 903 | 26 524 |
| ei151 | 460 | 471 | 449 | 467 | 426 |
| pr152 | 79 152 | 79 564 | 82 462 | 84 392 | 73 682 |
| ei176 | 576 | 590 | 573 | 593 | 538 |
| rat195 | 2 595 | 2 614 | 2 568 | 2 717 | 2 323 |
| kroA200 | 34 525 | 35 080 | 34 530 | 35 117 | 29 368 |
| ts225 | 137 391 | 138 675 | 134 886 | 139 877 | 126 643 |
| pr226 | 90 389 | 90 896 | 88 355 | 90 774 | 80 369 |
| gil262 | 2 707 | 2 738 | 2 828 | 2 953 | 2 378 |
| pr264 | 54 424 | 54 491 | 55 247 | 57 358 | 49 135 |
| a280 | 3 252 | 3 327 | 3 315 | 3 506 | 2 579 |
| pr299 | 57 098 | 57 748 | 57 533 | 58 965 | 48 191 |

内。本文提出的算法中蚂蚁选择下一城市节点采用的赌盘策略随着城市节点的数量增多,其找到的最优解虽然有所改善,但效率有所降低。接下来要研究的是让该算法如何更好地适用于大规模 TSP,同时也要更好地应用于实际问题的解决。

参考文献:


[1] DOERGO M, GAMBARDELLA L M. Ant colonies for the traveling salesman problem [J]. Bio-Systems, 1997, 43(2): 73 – 81.

[2] MURRAY A T, CHURCH R L. Applying simulated annealing to location-planning models [J]. Journal of Heuristics, 1996, 2(1): 31 – 53.

[3] 定建立,陈增强,袁著祉. 遗传算法与蚂蚁算法的融合 [J]. 计算机研究与发展, 2003, 40(9): 1531 – 1536.

[4] ROLLAND E, SCHILLING D A, CURRENT J R. An efficient tabu search procedure for p-Median problem [J]. European Journal of Operational Research, 1996, 96(2): 329 – 342.

[5] GLOVER F. Tabu search: part I [J]. ORSA Journal on Computing, 1989, 1(3): 190 – 206.

[6] 熊祯,郑兰芬,童庆禧. 分层神经网络算法 [J]. 测绘学报, 2000, 29(3): 229 – 234.

[7] RANDALL M. A parallel implementation of ant colony optimization [J]. Journal of Parallel and Distributed Computing, 2002, 62(9): 1421 – 1432.

[8] 孙学刚,员超,崔一辉 改进免疫算法在函数优化中的应用 [J]. 北京航空航天大学学报, 2010, 36(10): 1180 – 1188.

[9] DORIGO M, MANIEZZO V, COLORNI A. The ant system: optimization by a colony of cooperating Agents [J]. IEEE Transactions on Systems, Man and Cybernetics, 1996, 26(1): 23 – 31.

[10] DOERGO M, MANIEZZO V, COLORNT A. The ant system: optimization by a colony of coorperating Agents [J]. IEEE Transactions on Systems, Man, and Cybernetics—Part B, 1996: 26(1): 29 – 41

[11] DORIGO M, GAMBARDELLA LM. Antcolony system: a cooperative learning approach to the traveling salesman problem [J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53 – 66.

[12] TSPLIIB [EB/OI]. [2014-12-22]. http: //www. iwr. uni-heidelberg. de/groups/comopt/software/TSPLIB95/.


(上接第 110 页)


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

[5] KAUR R, GIRDHAR A, GUPTA S. Color image quantization based on bacteria foraging optimization [J]. International Journal of Computer Applications, 2011, 25(7): 33 – 42.

[6] ULAGAMMAI M, VANKATESH P, KANNAN P S, *et al*. Application of bacteria foraging technique trained and artificial and wavelet neural networks in load forecasting [J]. Neurocomputing, 2007, 70(16/17/18): 2659 – 2667.

[7] BANERJEE S, CHAKRABARTY A, MAITY S, *et al*. Feedback linearizing indirect adaptive fuzzy control with foraging based on-line plant model estimation [J]. Applied Soft Computing, 2011, 11(4): 3441 – 3450.

[8] 李明,杨成梧. 细菌菌落优化算法 [J]. 控制理论与应用, 2011, 28(2): 223 – 228.

[9] 宋德逻,孔德福,李明. 一种混合的离散细菌菌落优化算法 [J]. 计算机应用研究, 2014, 31(2): 358 – 360.

[10] 张娜. 细菌觅食优化算法求解车间调度问题的研究 [D]. 长春: 吉林大学, 2007.

[11] 崔静静,孙延明,车兰秀,等. 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.

[12] 易军,李太福. 求解作业车间调度的变邻域细菌觅食优化算法 [J]. 机械工程学报, 2012, 48(12): 178 – 183.

[13] KENNEDY J, EBEHART R C. Particle swarm optimization [C] // Proceedings of the 1995 IEEE International Conference on Neural Network. Piscataway: IEEE Service Center, 1995: 1942 – 1948.

[14] 姜建国,周佳薇,郑迎春,等. 一种双菌群细菌觅食优化算法 [J]. 深圳大学学报: 理工版, 2014, 31(1): 43 – 51.

[15] 胡洁. 细菌觅食优化算法的改进及应用研究 [D]. 武汉: 武汉理工大学, 2012.

[16] 张超勇,饶运清,李培根,等. 柔性作业车间调度问题的两级遗传算法 [J]. 机械工程学报, 2007, 43(4): 119 – 124.

[17] PAOLO B. Routing and scheduling in flexible job shop by tabu search [J]. Annals of Operations Research, 1993, 22(2): 157 – 183.