# 快速蚁群算法求解圆排列问题

章义刚[1,2],贾瑞玉[1],张燕平[1],王会颖[1]
(1.安徽大学 计算机学院,安徽 合肥 230039;
2.合肥学院,安徽 合肥 230022)

**摘 要**:圆排列问题属于NP-完全问题,且蚁群算法已成功地解决了许多组合优化的难题。介绍一种基于蚁群算法求解圆排列问题的算法,并对此算法进行优化,提出一种求解圆排列问题的快速蚁群算法。它大大减少了蚁群算法的搜索时间,有效改善了蚁群算法易于过早地收敛于非最优解的缺陷。仿真实验取得了较好的结果。

**关键词**:圆排列问题快速蚁群算法;圆排列问题蚁群算法;遗传算法
**中图分类号**:TP301 **文献标识码**:A **文章编号**:1673-629X(2007)08-0048-03

# Quick Ant Colony Algorithm of Solving Circle Permutation Problem

ZHANG Yi-gang[1,2],JIA Rui-yu[2],ZHANG Yan-ping[2],WANG Hui-ying[2]
(1.School of Computer Science, Anhui University,Hefei 230039,China;
2.Hefei University,Hefei 230022,China)

**Abstract**: Circle permutation problem is a difficult NP problem. Ant colony algorithm was applied successfully to many hard combinational optimization problems. An ant colony algorithm of solving circle permutation problem is introduced. By optimizing it, a quick ant colony algorithm of solving circle permutation problem is presented. It greatly reduces the searching time of ant colony algorithm. It also effectively ameliorates the disadvantage of easily falling in local best of ant colony algorithm. The simulation results show that the algorithm is more efficient.

**Key words**: CPPQACA; CPPACA; genetic algorithm

## 0 引 言

圆排列问题有很强的应用背景,如铺设半径大小不等的电缆管道、下水道等等,都可以转化为圆排列问题。实际工程中经常涉及到工件切割问题,如把一个矩形钢板切割成半径不等的圆,尽可能节省材料,这就是圆排列问题。圆排列问题属于NP-完全问题,目前没有有效的算法解此问题[1]。

蚁群算法(ant colony algorithm)是由意大利学者Dorigo M.等人在20世纪90年代初首先提出来的。它是一种新型仿生类进化算法,是继模拟退火算法、遗传算法、禁忌搜索算法、人工神经网络算法等元启发式搜索算法以后的又一种应用于组合优化问题的启发式搜索算法[2~4]。Dorigo M.等人将蚁群算法先后应用于TSP问题、资源二次分配问题[5]等经典优化问题,得到了较好的效果。蚁群算法在动态环境下也表现出高度的灵活性和健壮性,如其在电信路由控制方面的应用被认为是目前较好的算法之一[6]。蚁群算法从提出到现在,短短十余年的时间,以其在离散型组合优化问题中的突出表现,吸引了人们的极大关注。

虽然蚁群算法在离散型组合优化问题中表现突出,但也存在着缺陷,如搜索时间较长,且易于过早地收敛于非最优解。

文中介绍一种基于蚁群算法求解圆排列问题的模型,并对模型进行优化,提出求解圆排列问题的快速蚁群算法。它有效地减少了蚁群算法的搜索时间,改善了蚁群算法易于过早地收敛于非最优解的缺陷,是快速蚁群算法应用的又一范例。仿真实验也取得了较好的效果。

## 1 圆排列问题

圆排列问题的描述[7]为:给定 $n$ 个大小不等的圆 $c_1,c_2,\cdots,c_n$,现要将这 $n$ 个圆排进一个矩形框中,且要求与矩形的底边相切。圆排列问题要求从 $n$ 个圆的所

**收稿日期**:2006-10-19
**基金项目**:安徽省教育厅科研资助项目(2006KJ088B,2005KJ056)
**作者简介**:章义刚(1965-),男,安徽南陵人,工程师,研究方向为群体智能;张燕平,教授,研究方向为人工智能、机器学习。

有排列中找出有最小长度的圆排列，如图 1 所示。设 $p_1, p_2, \cdots, p_n$ 是所给 $n$ 个圆 $c_1, c_2, \cdots, c_n$ 的一个排列，圆 $p_1, p_2, \cdots, p_n$ 的半径分别为 $r_1, r_2, \cdots, r_n$，则相邻的两个圆 $p_i, p_j$ 之间的距离：

$$d_{ij} = 2\sqrt{r_i * r_j}$$

整个圆排列的长度为：

$$d = r_1 + 2(\sqrt{r_1 * r_2} + \sqrt{r_2 * r_3} + \cdots + \sqrt{r_{n-1} * r_n}) + r_n$$

因此圆排列为求 $\min(d)$。

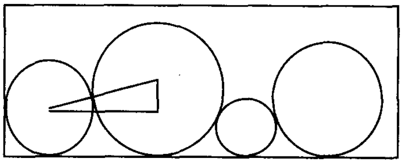


图 1　圆排列

由于要从 $n$ 个圆的所有排列中找出有最小长度的圆排列，所以圆排列问题的解空间是一棵排列树。文献[7]用回溯法解决此问题，其最坏情况接近于枚举法，时间复杂性为 $O((n+1)!)$。

## 2　求解圆排列问题的蚁群算法(CPPACA)

圆排列问题要求从 $n$ 个圆的所有排列中找出有最小长度的圆排列，圆排列的长度 $d$ 是由圆相互之间的排列位置决定。类似于 TSP 问题，在用蚁群算法求解时，信息素 $\tau(i,j)$ 表示两个圆之间信息素的浓度，启发函数 $\eta(i,j) = 1/d_{ij}$，蚂蚁 $k$ 从圆 $i$ 选择下一个圆 $j$ 的概率，计算公式见式(1)。$J(k)$ 为蚂蚁 $k$ 在圆 $i$ 时，没有访问的圆的集合。

信息素的更新采用 MMAS(max - min ant system)算法[8]中的信息素的处理方式，一圈中仅对圆排列的长度最小的蚂蚁所形成的排列进行信息素的修改增加，其余衰减。信息素的更新公式为式(2)。$\triangle\tau(i,j) = 1/L_{\mathrm{mb}}$，$L_{\mathrm{mb}}$ 为一代内 $m$ 只蚂蚁形成的 $m$ 个圆排列中的最短长度。

$$p_{ij}^k = \begin{cases} \dfrac{[\tau(i,j)]^\alpha \cdot [\eta(i,j)]^\beta}{\sum [\tau(i,j)]^\alpha \cdot [\eta(i,j)]^\beta} & s \in J(k) \\ 0 & \text{其它} \end{cases} \tag{1}$$

$$\tau(i,j) = \rho\tau(i,j) + \triangle\tau(i,j) \tag{2}$$

算法 CPPACA 描述为：

① 初始化：设定各参数的值，蚂蚁数 $m$，两个圆之间信息素的浓度 $\tau(i,j) = 1$。

② 每只蚂蚁独立地构造一个解：蚂蚁 $k(k = 1,2,\cdots,m)$ 随机选择一个圆作为起始解，并置于它的当前解集 $Z_k$ 中。蚂蚁 $k$ 按公式(1)计算从圆 $i$ 选择下一个圆 $j$ 的概率 $p_{ij}^k$，按概率 $p_{ij}^k$ 大者选择下一个圆 $j$，将 $j$ 置于 $Z_k$ 中，如此循环，直到蚂蚁 $k$ 访问完所有的圆。

③ 若 $m$ 只蚂蚁都构造完成各自的解，则转 ④，否则转 ②。

④ 根据这一代中圆排列的最小长度 $L_{\mathrm{mb}}$，按式(2)进行信息素更新。

⑤ 若满足结束条件，则输出最优解，否则 GEN = GEN + 1，转 ②。

算法 CPPACA 分析：

在算法 CPPACA 的第 ② 步中，概率 $p_{ij}^k$ 的计算次数为：$n-1+n-2+\cdots+1 = n*(n-1)/2$ 次，因此在一代内，概率 $p_{ij}^k$ 的计算次数为 $m*n*(n-1)/2$ 次。下一个圆的被选择概率 $p_{ij}^k$ 的大小主要依据信息素和两个圆间的距离长短。由于只有第 ④ 步时信息素才发生变化，所以在一代内，在第 ② 至 ③ 步时信息素 $\tau(i,j)$ 没有变化，蚂蚁 $k$ 和蚂蚁 $s$ 计算的各 $p_{ij}^k$ 和各 $p_{ij}^s$ 的大小次序不变。因而不必对每个蚂蚁都计算 $p_{ij}^k$，可在一代的开始统一计算 $p_{ij}$，这样一代内 $p_{ij}$ 的计算次数将降低为 $n*(n-1)/2$ 次，从而大大地减少计算量，缩短了搜索时间。另一点，为了增加搜索的随机性，② 步中蚂蚁 $k$ 按概率 $p_{ij}^k$ 大者从圆 $i$ 选择下一个圆 $j$，可改为蚂蚁 $k$ 按"轮盘赌"方式[9]根据概率 $p_{ij}$ 选择下一个圆，这样既兼顾了概率大小，又增加了搜索的随机性，兼顾解空间的多种情况，有效改善了蚁群算法易于过早地收敛于非最优解的缺陷。

## 3　求解圆排列问题的快速蚁群算法(CPPQACA)

根据对算法 CPPACA 的分析，对算法 CPPACA 进行改进，提出求解圆排列问题的快速蚁群算法(CPPQACA)，CPPQACA 的描述如下：

① 初始化。

② 按概率公式(1)计算从圆 $i$ 选择下一个圆 $j$ 的概率 $p_{ij}$。

③ 每只蚂蚁独立地构造一个解；蚂蚁 $k(k = 1,2,\cdots,m)$ 随机选择一个圆并置它于当前解集 $Z_k$ 中作为开始。蚂蚁 $k$ 根据概率 $p_{ij}$，按轮盘赌的方式从圆 $i$ 选择下一个圆 $j$，将 $j$ 置于 $Z_k$ 中，如此循环，直到蚂蚁 $k$ 访问完所有的圆。

④ 若 $m$ 只蚂蚁都构造完成各自的解，则转 ⑤，否则转 ③。

⑤ 根据这一代中圆排列的最小长度 $L_{\mathrm{mb}}$，按式(2)进行信息素更新。

⑥ 若满足结束条件，则输出最优解，否则 GEN = GEN + 1，转 ②。

## 4 仿真实验

仿真实验引用文献[1]的数据,并和文献[1]、圆排列问题蚁群算法 CPPACA 的实验结果相比较,结果见表1。圆排列问题 $n$ 取30,50,100,半径 $r = i(i = 1, 2, \cdots, n)$。

仿真实验运行环境:Intel4,CPU2.26G,VB6.0。在 CPPACA 和 CPPQACA 算法中,参数的取值为:$Q = 1$,$\alpha = 1, \beta = 1$ 或 $2, m = n, \rho$ 一般取 0.9 左右。实验结果比较见表1。表中的结果为连续运行 20 次的平均值及其中的最好、最差值。表1中平均时间及表2中的运行时间的单位都为秒。以下数据为 $n$ 取不同值时,用圆排列问题快速蚁群算法 CPPQACA 求得的各最优值对应各最优解。

$n = 30$,最优值为 750.7518 时,最优解各圆的排列半径为:15 17 13 19 11 21 9 23 7 25 5 27 3 29 1 30 2 28 4 26 6 24 8 22 10 20 12 18 14 16

$n = 50$,最优值为 2037.5342 时,最优解各圆的排列半径为:26 24 28 22 30 20 32 18 34 16 36 14 38 12 40 10 42 8 44 6 46 4 48 2 50 1 49 3 47 5 45 7 43 9 41 11 39 13 37 15 35 17 33 19 31 21 29 23 27 25

$n = 100$,最优值为 8015.4608 时,最优解各圆的排列半径为:51 54 50 49 47 53 39 68 33 66 31 72 29 67 36 62 38 58 41 65 35 64 40 61 42 57 45 55 43 59 48 56 46 63 34 70 28 78 22 80 21 75 20 77 30 69 32 76 27 73 24 88 13 84 15 79 23 74 17 81 18 86 26 85 16 96 4 90 11 97 9 93 6 99 1 98 2 95 5 100 3 94 7 91 8 87 10 92 12 89 14 82 19 83 25 71 37 60 44 52

表1 $n$ =30,50,100 的几种算法的结果比较

| 算法 | $n$ | 平均时间 | 平均值 | 最好解 | 最差解 |
|---|---|---|---|---|---|
| 文献[1]的蚁群模拟退火算法Ⅱ | 30 | 0.67 | 765.778 | 750.751 | 770.459 |
| | 50 | 0.98 | 2041.5 | 2037.5 | 2045.5 |
| | 100 | 1.05 | 8023.1 | 8019.8 | 8049.7 |
| 圆排列问题蚁群算法 CPPACA | 30 | 0.0546 | 787.469 | 784.417 | 792.951 |
| | 50 | 0.2815 | 2224.82 | 2218.76 | 2238.61 |
| | 100 | 2.2734 | 8843.46 | 8821.88 | 8867.70 |
| 圆排列问题快速蚁群算法 CPPQACA | 30 | 0.0390 | 752.843 | 750.751 | 754.662 |
| | 50 | 0.1718 | 2040.61 | 2037.53 | 2042.96 |
| | 100 | 1.0031 | 8021.54 | 8015.46 | 8036.47 |

从表1可以看出,圆排列问题快速蚁群算法 CPPQACA,在 $n$ =30,50,100 时,求解的平均时间低于其它算法,平均值、最好解、最差解都不差于圆排列问题蚁群算法 CPPACA 和文献[1]的蚁群模拟退火算法Ⅱ,取得了较好的结果。说明圆排列问题快速蚁群算法 CPPQACA 在提高计算速度的同时,又增加了搜索的随机性,兼顾解空间的多种情况,有效改善了蚁群算法易于过早地收敛于非最优解的缺陷。

表2比较了圆排列问题蚁群算法 CPPACA 和圆排列问题快速蚁群算法 CPPQACA 每一代的平均运行时间,CPPQACA 比 CPPACA 运行时间要快得多,体现了 CPPQACA 的快速性。

表2 CPPACA 与 CPPQACA 运行时间的比较

| $n$ | CPPACA 时间 s/代 | CPPQACA 时间 s/代 | $n$ | CPPACA 时间 s/代 | CPPQACA 时间 s/代 | $n$ | CPPACA 时间 s/代 | CPPQACA 时间 s/代 |
|---|---|---|---|---|---|---|---|---|
| | | | 250 | 35.3515 | 18.9375 | 400 | 146.3359 | 77.9062 |
| 150 | 7.8828 | 4.1875 | 300 | 61.8203 | 33.2890 | 450 | 217.9296 | 110.3752 |
| 200 | 18.2890 | 9.9296 | 350 | 98.4453 | 52.3515 | 500 | 288.7815 | 151.0269 |

## 5 结 论

介绍了基于蚁群算法求解圆排列问题的圆排列问题蚁群算法 CPPACA,并对 CPPACA 进行改进,提出了圆排列问题快速蚁群算法 CPPQACA。CPPQACA 通过改变概率计算的时机,使一代中概率 $p_{ij}$ 计算的次数从 $mn(n-1)/2$ 次降低到 $n(n-1)/2$ 次,大大提高了计算速度。蚂蚁采用"轮盘赌"方式根据概率 $p_{ij}$ 选择下一个圆,这样既兼顾了概率大小,又增加了搜索的随机性,兼顾解空间的多种情况,有效改善了蚁群算法易于过早地收敛于非最优解的缺陷。仿真实验取得了较好的结果。快速蚁群算法和遗传算法、禁忌搜索算法、模拟退火算法等模拟进化算法融合,和现代研究的改进新方法结合,进一步改善蚁群算法的搜索时间较长,且易于过早地收敛于非最优解的缺陷,是进一步研究的问题。

**参考文献:**

[1] 高 尚,杨静宇,吴小俊,等. 圆排列问题的蚁群模拟退火算法[J]. 系统工程理论与实践,2004,8(8):102-106.

[2] Dorigo M, Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem[J]. IEEE Trans on Evolutionary Computation, 1997, 1(1):53-66.

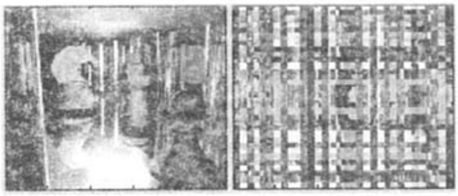
[3] Colorni A, Dorigo M. Heuristics from nature for hard combinatorial optimization problems[J]. International Trans Operational Research, 1996, 3(1):1-21.

[4] Dorigo M. ACO Algorithms for the Traveling Salesman Problem[C]//Evolutionary Algorithms in Marco Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Application. [s. l.]: John Wiley & Sons, 1999.

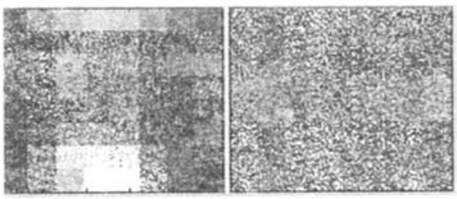
[5] Stutzle T, Dorigo M. ACO Algorithms for the Quadratic Assignment Problem[C]//In: New Ideas in Optimization. New York: McGraw-Hill, 1999.

(下转第 54 页)

$\mu=3.8, x_0=0.8, k=500$ 为第 2 组密钥的实验结果，(a)为 256×256 的 chess 原图像；(b)为把图像分成 8×8 子块后的块间置乱；(c)为把图像分成 32×32 子块的直接置乱；(d)为块间置乱后以 32×32 子块置乱。



(a)原图像 (b) 8×8 子块块间置乱

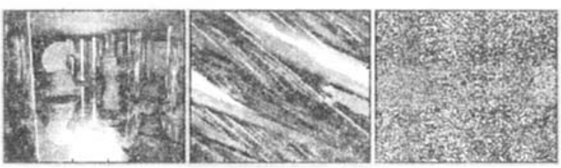


(c) 32×32 子块直接置乱 (d) 对(b)以 32×32 子块置乱

图 3 交叉块加密结果

## 5 实验结果

如图 4 所示，(a)为 256×256 的 chess 原图像；(b)为 Arnold 算法加密结果；(c)为把图像分成 8×8 子块块间置乱后的块加密。无论从视觉效果，还是从图像置乱程度，与 Arnold 算法相比，本算法明显优于 Arnold 算法加密结果。



(a)原图像 (b) Arnold 算法加密 (c)块置乱的块加密

图 4 块置乱的块加密算法和 Arnold 算法效果比较

块间置乱的块尺寸为 8 * 8，以 32 * 32 的块尺寸对块间置乱后的图像进行加密实验，从表 1 的实验数据结果不难看出（实验环境为主频 2.8GHz 的 P4 机器，内存 512M，MATLAB6.5），文中提出的算法明显优于 Arnold 算法。

表 1 与 Arnold 算法的处理效率对比

| 图像尺寸 | 块置乱的分块加密 | Arnold 加密 |
|---|---|---|
| 256 * 256 | 0.0630 | 0.2190 |
| 512 * 512 | 0.2260 | 1.7810 |
| 1024 * 1024 | 4.000 | 13.500 |
| 1280 * 1280 | 9.7970 | 26.0780 |

## 6 结 论

无论从图像置乱程度来看，还是从图像的处理效率来看，文中提出的算法均优于 Arnold 算法。由于混沌序列对初始值非常敏感，即使密钥值有微小的变化也会得到完全不同的解密结果，实验证明，本算法具有加密速度快、加密效果好等特点。

**参考文献：**

[1] 陈 刚，赵晓宇，李均利. 一种自适应的图像加密算法[J]. 软件学报，2005，16(11)：175－182.

[2] 吴昊升，王介生，刘慎权. 图像的排列变换[J]. 计算机学报，1998，21(6)：514－519.

[3] 丁 玮，齐东旭. 数字图像变换及信息隐藏与伪装技术[J]. 计算机学报，1998，21(9)：838－843.

[4] Shi C, Bhargava B. A Fast MPEG Video Encryption Algorithm[C]//In Proc. of ACM Multimedia'98, Electronic Proceeding. New York: ACM Press, 1998: 151－160.

[5] Scharinger J. Fast Encryption of Image Data Using Chaotic Kolmogorov Flows[J]. SPIE, 1997, 3022: 278－289.

[6] 卢朝阳，周幸妮. 一种新的数据信息值乱算法[J]. 计算机工程与科学，1998，20(3)：28－41.

[7] Wong P W, Memon N. A public key watermark for image verification and authentication[C]//In: Proceedings of the IEEE International Conference on Image Processing. Chicago: [s.n.], 1998: 455－459.

[8] Memon N, Vora P, Yeo B L, et al. Distortion bounded authentication techniques[C]//In: Proceedings of the SPIE, Security and Watermarking of Multimedia Contents II. San Jose, USA: [s.n.], 2000: 164－174.

[9] 沃 焱，韩国强，张 波. 一种新的基于特征的图像内容认证方法[J]. 计算机学报，2005，28(1)：105－112.

[10] 孙 鑫，易开祥，孙优贤. 基于混沌系统的图像加密算法[J]. 计算机辅助设计与图形学报，2002，13(2)：136－139.

（上接第 50 页）

[6] 吴 斌，史忠植. 一种基于蚁群算法的 TSP 问题分段求解算法[J]. 计算机学报，2001，24(12)：1328－1333.

[7] 王晓东. 计算机算法设计与分析[M]. 第 2 版. 北京：电子工业出版社，2005.

[8] Stutzle T, Hoos H H. MAX－MIN ant system[J]. Future Generation Computer System, 2000, 16(8): 889－914.

[9] 徐精明，曹先彬，王煦法. 蚁群算法求解问题时易产生的误区及对策[J]. 计算机工程，2004，30(16)：25－26.