文章编号: 1001-0920 (2013) 06-0873-06

一种基于粒子群参数优化的改进蚁群算法

李 擎[1], 张 超[1], 陈 鹏[2], 尹怡欣[1]

(1. 北京科技大学 自动化学院，北京 100083；2. 中国科学院 国家天文台，北京 100012)

摘 要: 蚁群算法是一种应用广泛、性能优良的智能优化算法, 其求解效果与参数选取息息相关. 鉴于此, 针对现有基于粒子群参数优化的改进蚁群算法耗时较大的问题, 提出一种新的解决方案. 该方案给出一种全局异步与精英策略相结合的信息素更新方式, 且通过大量统计实验可以在较大程度上减少蚁群算法被粒子群算法调用一次所需的迭代代数. 仿真实验表明, 所提出算法在求解较大规模旅行商问题时具有明显的速度优势.

关键词: 粒子群算法；改进蚁群算法；迭代代数；旅行商问题

中图分类号: TP18 **文献标志码:** A

Improved ant colony optimization algorithm based on particle swarm optimization

LI Qing[1], *ZHANG Chao*[1], *CHEN Peng*[2], *YIN Yi-xin*[1]

(1. School of Automation，University of Science and Technology Beijing，Beijing 100083，China；2. National Astronomical Observatories，Chinese Academy of Sciences，Beijing 100012，China. Correspondent：LI Qing, E-mail：liqing@ies.ustb.edu.cn)

Abstract: Ant colony optimization(ACO) algorithm is an intelligent algorithm which has a wide range of applications and better performance, and its search quaility is closely related with the parameters selection. Therefore, aiming at the large time-consuming problem of the existing improved ACO alogorithm, a novel ACO algorithm based on particle swarm optimization(PSO) algorithm is proposed. The new pheromone update method is presented, which combines the global asynchronous feature and elitist strategy. Moreover, the iteration number of ACO algorithm invoked by PSO algorithm is reduced significantly by large amounts of statistical experiments. The simulation results show that the proposed ACO algorithm has obvious advantage in search speed when it is used for solving the large-scale traveling salesman problem.

Key words: particle swarm optimization；improved ant colony optimization；iteration number；traveling salesman problem

0 引 言

20 世纪 90 年代, Dorigo 等[1-2]受蚂蚁觅食行为的启发, 提出了蚁群算法. 常见的蚁群算法模型有 AS (ant system) 模型[3]、ACS(ant colony system) 模型[4]、MMAS(max-min ant system) 模型[5]等. 目前蚁群算法已成功应用于多种优化问题, 如车辆调度、旅行商问题 (TSP) 等. 蚁群算法的求解质量与算法参数的选取息息相关, 参数的给出依赖于实验者的个人经验和一次次的试凑. 这种方式效率较低, 智能性较差, 极大地制约了蚁群算法发挥其最佳性能, 因此蚁群算法参数的优化问题一直是国内外学者研究的热点之一.

粒子群优化算法 (PSO)由 Kennedy 等[6-7]提出, 该算法源于对鸟群觅食过程的模拟. 粒子群算法的初始状态为一组随机分布在解区间内的粒子, 每个粒子代表一个解. 算法综合利用种群信息和个体经验, 不断更新粒子的位置 (解的大小) 和速度 (解变化的步长), 以便向最优或较优解逼近.

如何应用粒子群算法优化蚁群算法的参数是本文探讨的主要内容. 闵克学等[8]采用粒子群算法优化了 AS 模型中的 α, β 两个参数, 柴宝杰等[9]优化了 ACS 模型中 β, ρ, q_0 三个参数, 夏辉等[10]和杨亚楠[11]均优化了 AS 模型中 α, β, ρ 三个参数. 虽然这些算法

收稿日期: 2012-02-13；**修回日期:** 2012-05-17.

基金项目: 国家自然科学基金项目(60374032)；教育部第36批留学回国人员科研启动基金项目(1341)；北京市重点学科建设项目(XK100080537).

作者简介: 李擎(1971−), 男, 教授, 博士, 从事智能控制算法及其应用等研究；尹怡欣(1957−), 男, 教授, 博士生导师, 从事智能控制系统等研究.

在求解TSP时都体现出良好的性能, 在避免蚁群算法参数选取主观性的同时还得出了质量较高的解, 但它们均在不同程度上存在时间开销较大的问题.

针对以上问题, 本文提出一种全局异步与精英策略相结合的信息素更新方式, 通过大量统计实验更加合理地确定了粒子群算法中每个粒子调用一次蚁群算法时, 蚁群算法需要迭代的代数, 大大减小了算法的计算规模. 为了验证本文所提算法的有效性, 对经典TSP进行了大量的仿真实验, 实验结果表明, 与现有算法相比, 本文所提出的改进算法能在保证求得可行解的前提下, 大大减小时间成本.

1 蚁群算法及其数学模型

TSP是数学与计算领域的经典问题. 该问题规定商人从某一城市出发, 到所有目标城市买卖货物, 要求寻找一条遍历所有城市且每个城市仅访问一次的最短路径. 蚁群算法求解TSP的过程, 实质是所有蚂蚁并行地构建TSP路径的过程. 以AS模型为例, 用$P_{ij}^k(t)$来表示蚂蚁k在城市i选择下一城市j的概率, 有

$$P_{ij}^k(t)=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum\limits_{s\in\text{allowed}_k}\tau_{is}^{\alpha}(t)\eta_{is}^{\beta}(t)}, & j\in\text{allowed}_k;\\ 0, \text{ otherwise.}\end{cases}\tag{1}$$

其中: $\tau_{ij}(t)$为t时刻城市i与城市j之间路径上的信息素; $\eta_{ij}(t)$为启发式因子, 取$\eta_{ij}(t)=1/d_{ij}$, d_{ij}为城市i与城市j之间的距离; α为τ相对重要程度; β为η相对重要程度; allowed_k为t时刻蚂蚁k所能选择的城市集合.

当一只蚂蚁访问过所有城市后, 该蚂蚁的行程即为一个包含所有城市的序列, 是可行解. 蚂蚁在经过的路径上释放信息素, 每当蚂蚁遍历完所有城市, 就对留存在环境中的信息素进行更新, 更新公式如下所示:

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t),\tag{2}$$

$$\Delta\tau_{ij}(t)=\sum\Delta\tau_{ij}^k(t).\tag{3}$$

其中: ρ为信息素的挥发程度, 取值范围为[0, 1]; $\Delta\tau_{ij}^k$为蚂蚁k本次循环中在路径$[i,j]$上释放的信息素量; $\Delta\tau_{ij}(t)$为本次循环后路径$[i,j]$上信息素的增量.

采用蚁周模型作为信息素挥发方式, 有

$$\Delta\tau_{ij}^k(t)=\begin{cases}Q/L_k, \text{ ant } k \text{ passes though } [i,j];\\ 0, \text{ otherwise.}\end{cases}\tag{4}$$

其中: Q为信息素总量; L_k为蚂蚁k在本次循环中所走路径的长度. 随着所有蚂蚁完成对TSP路径的构建, 信息素不断积累、挥发, 直到满足终止条件(迭代次数超过某一设定值或现有适应度达到某一值)为止. 此时, 现有蚂蚁构建的最优路径即为求得的最终解.

2 基于粒子群参数优化的改进蚁群算法

2.1 问题的概述及定义

文献[8-11]的方法在本质上是一致的, 这些方法用粒子群算法中的一个粒子表示蚁群算法的一组参数, 通过较优粒子的寻找完成蚁群算法参数的优化. 优化过程中, 粒子群算法将重复调用蚁群算法, 并选取蚁群算法中的路径长度作为判断参数优劣的标准.

为了更好地分析和比较各种基于粒子群参数优化的改进蚁群算法, 以文献[10]中的参数优化为标准, 作以下定义.

定义1(三维粒子及其位置) 粒子群算法用一个粒子对应一组α, β, ρ参数, 粒子在解空间中的坐标即为参数值, 记为$x_i=(x_{i0},x_{i1},x_{i2})$.

定义2(粒子的速度) 与位置相对应, 每个粒子在解空间中移动时具有3个方向的速度. 记为$V_i=(V_{i0},V_{i1},V_{i2})$.

定义3(最优粒子位置) 如果将某个粒子所对应的参数值反馈回蚁群算法, 蚁群算法的求解结果最优, 则称该粒子为最优粒子. 最优粒子所对应的一组参数表明该粒子在解空间中的位置, 即最优粒子位置. 记每个粒子当前找到的最优粒子位置为P_{best}, 整个种群当前所找到的最优粒子位置为G_{best}, P_{best}和G_{best}的适应度评价值即为蚁群算法求得路径的长度, 路径长度越小参数质量越好.

定义4(粒子群算法的进化代数) 粒子群算法进化一代, 是指所有粒子从现有位置按照当前速度移动至新位置的过程.

定义5(调用蚁群算法的次数) 蚁群算法被调用一次, 是指进行一次完整蚁群算法求解的过程, 即所有蚂蚁完成构建TSP路径并进行X代迭代的过程, 其中X为蚁群算法规定的迭代代数.

定义6(蚁群算法的迭代代数) 蚁群算法进行一次迭代, 是指所有蚂蚁从起点开始, 遍历所有城市点, 构建TSP路径, 并释放信息素的过程.

2.2 已有基于粒子群参数优化的改进蚁群算法

以文献[10]为例, 应用粒子群算法优化蚁群算法参数的基本步骤如下.

Step 1: 初始一定数量的粒子$P_0, P_1, \cdots, P_n$.

Step 2: 将当前每个粒子所对应的参数值反馈回蚁群算法, 一个粒子对应一组参数α, β, ρ, 应用这组参数调用一次蚁群算法, 然后将环境中的信息素重新初始化.

Step 3: 根据蚁群算法的求解结果判断粒子位置

的优劣. 更新 $P_{\rm best}$ 和 $G_{\rm best}$.

Step 4: 按照下式更新粒子的速度和位置:

$$V_{i(k+1)} = wV_{i(k)} + c_1r_1(P_{best.i} - x_{i(k)}) + c_2r_2(G_{\rm best} - x_{i(k)}), \quad (5)$$

$$x_{i(k+1)} = x_{i(k)} + V_{i(k+1)}. \quad (6)$$

其中: w 为惯性权重; c_1 和 c_2 为两个常数, c_1 为调节粒子向自身最好位置靠近的权重, c_2 为调节粒子向全局最好位置靠近的权重, c_1 和 c_2 通常在 0～2 间取值; r_1 和 r_2 为两个相互独立的随机数; $V_{i(k)}$ 为粒子 i 在第 k 代的速度, $x_{i(k)}$ 为粒子 i 在第 k 代的位置; $P_{{\rm best}.i}$ 为当前粒子 i 找到的最优位置, $G_{\rm best}$ 为所有粒子找到的全局最优位置.

Step 5: 若满足终止条件 (达到粒子最大进化代数或粒子连续进化若干代未出现性能更好的粒子), 则算法结束, 返回当前的全局最优粒子位置, (α, β, ρ) 为蚁群算法中 3 个参数值的最佳组合; 若不满足终止条件, 则返回 Step2.

当采用粒子群算法优化蚁群算法中的参数 α, β, ρ 时, 不可避免地会引入一些新的参数, 如 number, w, c_1, c_2, 这 4 个参数的选取具有一定的经验, 取值方式相对简单, 且对最终解的影响不大, 文献 [10-11] 中已经给出.

在文献 [10] 提出的改进算法中, 每个粒子进行一次移动, 蚁群算法将被完整地调用一次. N 个粒子的 M 次移动过程需调用 $N \times M$ 次蚁群算法, 而每次蚁群算法都将进行几十甚至几百代迭代. 所以, 该算法虽具有较高的求解质量, 但所需时间成本极大.

2.3　所提基于粒子群参数优化的改进蚁群算法

如何在保证求解质量的同时, 能合理地减小算法的时间成本是本文所讨论的主要内容. 为达到这一目的, 开展以下两方面的工作: 1) 改善蚁群算法中信息素的更新方式; 2) 缩减蚁群算法被单个粒子调用一次时所需要的迭代代数, 由最初的几十、几百代甚至更多缩减至 10 代以内.

2.3.1　信息素更新方式的确定

信息素更新方式的选取对蚁群算法的求解质量具有极为重要的影响, 这也是本文对蚁群算法进行改进的重要内容之一. 文献 [10] 中改进算法的信息素承接过程只适用于每个粒子的一次移动, 且仅当独立的蚁群算法被调用时, 信息素才会承接, 一旦调用结束, 该过程中所积累的环境信息即被清零. 该过程是独立的、随时清空的. 本文试图通过大幅度减少被调用蚁群算法迭代代数的方式来降低时间成本, 若仍采取上述随时清空的信息素承接方式, 则蚁群在小规模的迭代代数内无法有效积累环境信息, 从而会一直停留在初始的随机状态, 不能得到较优解. 如果能够选取一种在粒子间承接并在整个算法求解过程中持续更新的信息素承接方式, 则可以在以较小时间成本切换粒子对应蚁群算法参数的同时, 保留并积累足够的环境信息, 有效进行寻优. 本文的改进信息素更新方式正是基于以上机理提出的.

本文提出以下几种改进的信息素更新方式, 并对它们进行对比分析.

方式 1　全局同步信息素更新方式. 在蚁群算法被调用的过程中, 蚁群每进化一代, 信息素随之更新, 参数切换时, 信息素不重新初始化, 继续进行更新, 更新公式如式 (2) 和 (3) 所示.

方式 2　全局异步信息素更新方式. 在蚁群算法被调用的过程中, 蚁群每一代进化, 信息素并不进行更新, 当且仅当出现适应度更强的解时, 更新信息素, 更新公式如下:

$$\tau_{ij}(t+1) = \tau_{ij1}(t). \quad (7)$$

其中: $\tau_{ij}(t)$ 为环境中的信息素; $\tau_{ij1}(t)$ 为按照方式1进行信息素更新时, 持续到当前代数 (即出现最优解时刻) 环境中的信息素量. 参数切换时, 信息素不重新初始化. 蚂蚁进行概率选择时, 始终按照最近一次出现最优解时的信息素状态进行选择, 有

方式 3　全局同步与精英策略相结合的信息素更新方式. 在蚁群算法被调用的过程中, 蚁群每进化一代, 信息素随之更新. 参数切换时, 信息素不重新初始化. 当未出现适应度更强的解时, 信息素更新公式如式 (2) 和 (3) 所示. 当出现适应度更强的解时, 对该精英蚂蚁进行信息素增强, 按照下式更新信息素:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) + \Delta\tau_{ij}(t)'. \quad (8)$$

其中: $\Delta\tau_{ij}(t)$ 由式 (3) 求得; $\Delta\tau_{ij}(t)'$ 为本次循环中最优蚂蚁在路径 [i,j] 上释放的信息素量, 且有

$$\Delta\tau_{ij}(t)' = \begin{cases} Q/G_k, & [i,j] \text{ in the best path;} \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

G_k 为最短路径的长度.

方式 4　全局异步与精英策略相结合的信息素更新方式. 蚁群每一代进化, 信息素并不进行更新, 当且仅当出现适应度更强的解时, 更新信息素, 并对该全局最优解经过的路径进行信息素加强. 参数切换时, 信息素不重新初始化. 信息素更新公式如下:

$$\tau_{ij}(t+1) = \tau_{ij1}(t) + \Delta\tau_{ij}(t)', \quad (10)$$

其中 $\Delta\tau_{ij}(t)'$ 由式 (9) 计算得出.

为了更好地说明不同信息素更新方式的效果, 针对 TSPLIB95 中的 St70 问题进行仿真研究. 取粒子数 number = 30, $Q = 2$, $c_1 = 1.9$, $c_2 = 0.8$, $w = 0.5$, 蚂蚁数量 AntCount= 35, 蚁群算法被单个粒子调用时,

每次仅迭代3步(具体取值在第2.3.2节中加以阐述). 终止条件为粒子群算法最大迭代次数10.

本文为粒子的位置和速度设定了取值范围, 规定$\alpha \in [0,5]$, $\beta \in [0,5]$, $\rho \in [0,1]$, $V_\alpha \in [-0.5,0.5]$, $V_\beta \in [-0.5,0.5]$, $V_\rho \in [-0.2,0.2]$. 对粒子位置采取分布均匀、适当加强的方式进行初始化, 分别取$\alpha = \{1,2,3,4\}$, $\beta = \{1,2,3,4\}$, $\rho = \{0.2,0.4,0.6,0.8\}$. 通过排列组合, 可形成$4\times4\times4 = 64$个初始粒子. 此后根据经验, 初始化一些粒子, 如蚁群算法中常取$\alpha = 1, \beta = 3$或$\alpha = 2, \beta = 4$, 因此初始化时可包含粒子$\{\alpha,\beta,\rho\} = \{1,3,0.4\},\{2,4,0.6\}$等. 粒子速度的初始化采用随机方式, 即选择其取值范围内的任意数. 当粒子的位置或速度超越其取值范围时, 对其作取极限值处理, 当$\alpha < 0$时取$\alpha = 0$, 当$\rho > 1$时取$\rho = 1$, 当$V_\beta < -0.5$时取$V_\beta = -0.5$.

为了减少粒子群和蚁群算法中的随机性, 将4种不同信息素更新方式下的改进蚁群算法各运行5次, 结果如表1所示. 由表1可见, 当采用全局异步与精英策略相结合的信息素更新方式时, 求解效果最佳.

表 1 4种信息素更新方式下改进算法求解结果

信息素更新方式	解/m		运行时间/s	
	平均值	最优解	平均值	最优解
全局同步	777	729	47	42
全局异步	733	719	46	44
全局同步与精英策略结合	729	714	42	40
全局异步与精英策略结合	723	712	40	36

2.3.2 蚁群算法迭代代数 X 的确定

在改进算法中, 一代中的每个粒子均将调用一次蚁群算法. 此时, 蚁群算法迭代代数X的确定是一个值得探讨的问题, 也是决定改进算法搜索速度的关键所在. 当X取值过大时, 不仅会造成时间成本的浪费, 还可能导致蚁群算法陷入局部最优, 不能合理优化参数. 文献[10-11]正是该原因使得其搜索效率相对较低.

本文对TSPLIB95中的41个对称TSP进行统计实验, 以确定适用于不同TSP的X值. 改进算法的参数设定如下: 粒子数number$= 30$, $Q = 0.33L + K$. 其中: L为某次蚁群算法求得的最优解, K为使Q满足整百、整千的实数. $c_1 = 2$, $c_2 = 2$, $w = 0.5$, AntCount $= n/2$, n为城市数. 终止条件为粒子群算法进化30代, 粒子群的初始化方法如第2.3.1节所述, 信息素更新采用全局异步与精英策略相结合的方式. 同样, 为了克服随机性, 对每个不同X值下的改进蚁群算法各运行5次, 并选取5个结果的平均值作为求解结果. 由于篇幅有限, 文中仅列出10个TSP在不同X值下的求解结果, 如表2所示, 表2中数值已取整.

表 2 TSP在不同 X 值下的求解结果(部分)

TSP	X						
	2	3	4	5	6	7	8
Att48	11 664	11 191	11 143	11 117	11 215		
Bayg29	1 714	1 698	1 691	1 707	1 672	1 680	1 677
Berlin52	7 803	8 269	7 770	7 636	7 862	7 797	7 748
Eil51	449	472	465	463			
Eil76	568	576	566	565	572	570	
Gr137	77 290	75 448	74 367	74 247	74 622	76 385	
KroA100	23 430	23 615	23 274	22 611	23 039	23 334	23 329
KroA150	29 457	29 051	29 156	29 247	29 168	29 547	29 353
KroB100	23 800	23 728	23 694	23 245	23 558	23 344	23 634
Pr136	111 039	111 442	111 274	110 851	111 174	115 006	114 483

为了更好地说明问题, 本文选取5次平均解最小的X值作为每次蚁群算法合理的迭代代数, 如表3所示. 由表3可见, 对于多数TSP, 其X值取2～5间的数字即可.

表 3 不同TSP最优的 X 值

序号	TSP	问题类型	最优X	序号	TSP	问题类型	最优X
1	Att48	ATT	5	22	Hk48	MATRIX	3
2	Bayg29	GEO	6	23	KroA100	EUC	5
3	Bays29	GEO	4	24	KroA150	EUC	3
4	Berlin52	EUC	5	25	KroB100	EUC	5
5	Bier127	EUC	3	26	KroB150	EUC	5
6	Brazil58	MATRIX	2	27	KroC100	EUC	5
7	Burma14	GEO	2	28	KroD100	EUC	7
8	Ch130	EUC	3	29	KroE100	EUC	3
9	Ch150	EUC	3	30	Lin105	EUC	4
10	Dantzig42	MATRIX	2	31	Pr76	EUC	4
11	Eil51	EUC	2	32	Pr107	EUC	2
12	Eil76	EUC	5	33	Pr124	EUC	2
13	Eil101	EUC	5	34	Pr136	EUC	5
14	Fri26	MATRIX	2	35	Pr144	EUC	3
15	Gr17	MATRIX	2	36	Rat99	EUC	8
16	Gr21	MATRIX	2	37	Rd100	EUC	4
17	Gr24	MATRIX	2	38	St70	EUC	3
18	Gr48	MATRIX	3	39	Swiss42	MATRIX	2
19	Gr96	GEO	3	40	Ulysses6	GEO	3
20	Gr120	MATRIX	3	41	Ulysses22	GEO	4
21	Gr137	GEO	5				

3 仿真研究

分别对Eil51问题和Eil262问题运用本文所提出的改进算法求解, 作为对比, 利用文献[8,10]的方法求解同样的问题. 为了客观对比各种算法的时间代价, 对文献[8,10]的算法进行了时间尺度统一化处理, 即粒子每次调用蚁群算法时所使用的迭代代数相同. 算法具体描述如下.

算法1(文献[8]方法) 固定ρ, 采用粒子群算法优化α和β. 每判断一次粒子优劣, 需调用一次蚁群算法, 蚁群算法迭代几十到几百步, 调用结束后, 信息素重新初始化.

算法 2 (文献 [10] 方法)　采用粒子群算法优化 α, β 和 ρ. 每判断一次粒子优劣, 需调用一次蚁群算法. 蚁群算法迭代几十到几百步, 调用结束后, 信息素重新初始化.

算法 3 (文献 [8] 方法, 蚁群算法被调用时仅迭代 X 步)　固定 ρ, 采用粒子群算法优化 α 和 β. 每判断一次粒子优劣, 需调用一次蚁群算法, 蚁群算法迭代 X 步, 调用结束后, 信息素重新初始化.

算法 4 (文献 [10] 方法, 蚁群算法被调用一次时仅迭代 X 步)　采用粒子群算法优化 α, β 和 ρ. 每判断一次粒子优劣, 需调用一次蚁群算法, 蚁群算法迭代 X 步, 调用结束后, 信息素重新初始化.

算法 5 (本文所提算法)　采用粒子群算法优化 α, β 和 ρ, 每判断一次粒子优劣, 需调用一次蚁群算法, 蚁群算法迭代 X 步, 信息素在算法整个过程中不断更新, 调用结束后不重新初始化, 而采用第 2.3.1 节提出的方式 4 更新信息素.

所有实验程序均用 C++ 编写, 在 Visual Studio 2008 环境下运行, PC 机主频为 2.40 GHz.

3.1　Eil51 问题

Eil51 问题取值如下[10]: 粒子数 number = 30, w = 0.8, $c_1 = 1.2$, $c_2 = 1.6$, AntCount = 25, $Q = 120$, X = 3, 算法 1 和算法 2 中, 蚁群算法迭代次数 ItCount = 30, 算法 1 和算法 3 中固定 ρ 为文献 [10] 中所得的最优值 0.491 852. 粒子群的初始方式如第 2.3.1 节所述. 当粒子群算法迭代到 15 代时, 算法终止. 对每种算法各求解 5 次, 统计结果如表 4 所示.

表 4　Eil51问题的统计结果

算法	解/m		运行时间/s	
	平均值	最优解	平均值	最优解
1	455	442	135	135
2	448	436	149	131
3	464	448	32	32
4	462	445	32	31
5	477	447	40	31

由表 4 可见, 综合最优解的质量、平均值和时间各指标, 算法 5 的求解效果并不理想. 该算法虽在一定程度上节省了时间成本 (平均耗时分别为算法 1 的 29.63% 和算法 2 的 26.85%), 但其所得解质量逊于算法 1 和算法 2 (其平均解比算法 1 长 4.84%, 比算法 2 长 6.47%). 因此, 对于规模不大的 TSP, 算法 5 并不适用.

5 种算法关于 Eil51 问题的粒子群迭代代数与最优路径长度的关系如图 1 所示. 采用本文所提算法求得的 TSP 最优路径如图 2 所示.

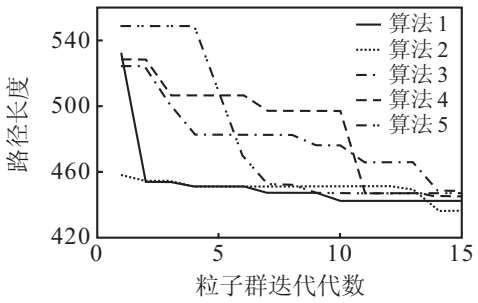

图 1　最优路径长度与粒子群迭代代数关系

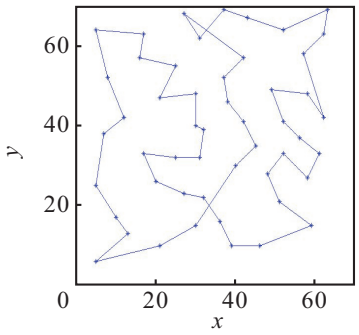

图 2　本文所提算法求得的 Eil51 问题最优路径

3.2　Gil262 问题

针对 Gil262 问题, 取粒子数 number = 30, Q = 1 000, $w = 0.5$, $c_1 = 2$, $c_2 = 2$. AntCount = 131, $X = 3$. 粒子群的初始方式如第 2.3.1 节所述. 算法 1 和算法 2 中, 蚁群算法迭代代数 ItCount = 100, 算法 1 和算法 3 中固定 $\rho = 0.6$. 终止条件为粒子群算法迭代 30 代. 对以上 5 种算法各求解 5 次, 统计结果如表 5 所示.

表 5　Gil262 问题的统计结果

算法	解/m		运行时间/s	
	平均值	最优解	平均值	最优解
1	2 583	2 433	129 072	122 063
2	2 545	2 452	129 724	115 069
3	2 877	2 618	4 639	4 321
4	2 795	2 626	4 368	4 068
5	2 639	2 564	4 729	4 537

由表 5 可见: 1) 从求解质量上看, 算法 5 略逊于算法 1 和算法 2, 其平均解比算法 1 长 2.17%, 比算法 2 长 3.69%, 但在平均时间方面, 相对于算法 1 和算法 2, 算法 5 大大节省了时间成本, 其耗时仅为算法 1 的 3.66%, 算法 2 的 3.65%; 2) 在同等时间尺度下, 相对于算法 3 和算法 4, 算法 5 的平均解为算法 3 的 91.73%, 算法 4 的 94.42%, 由此可见其求解质量相对较高.

5 种算法关于 Gil262 问题的粒子群迭代代数与最优路径长度的关系如图 3 所示. 采用本文所提算法求得的 TSP 最优路径如图 4 所示.

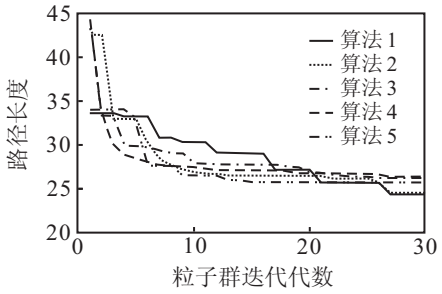

图 3 最优路径长度与粒子群迭代代数的关系

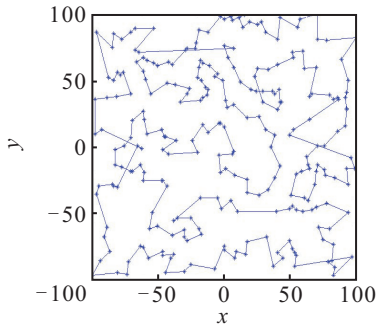

图 4 本文所提算法求得的 Gil262 问题最优路径

3.3 对比分析

为了更好地表明所提算法的有效性, 本文对 TSPLIB95 中的 St70 和 Ch150 两种 TSP 进行仿真实验研究. 在解的质量相差较小的条件下, 算法 1, 算法 2, 算法 5 的求解时间如表 6 所示. 由表 6 可见, 算法 5 在保证求解质量的同时, 大大减小了时间成本, 且随着问题规模的增大, 其速度优势越来越明显.

表 6 算法 1, 算法 2, 算法 5 的求解时间 s

算法	Eil51 问题		St70 问题		Ch150 问题		Gil262 问题	
	平均	最优	平均	最优	平均	最优	平均	最优
1	135	135	256	250	6 032	5 892	129 072	122 063
2	149	131	262	253	7 013	6 302	129 724	115 069
5	40	31	40	36	892	860	4 729	4 537
5/1	29.6%	23.0%	15.6%	14.4%	14.8%	14.6%	3.7%	3.7%
5/2	26.9%	23.7%	15.3%	14.2%	12.7%	13.7%	3.7%	3.9%

在相同的时间尺度下, 算法 3～算法 5 求得解的情况如表 7 所示. 由表 7 可见, 在运行时间相差较小的前提下, 对于小规模的 TSP, 算法 5 的求解质量稍差, 在求解大规模的 TSP 时, 求解质量较高.

表 7 算法 3～算法 5 的所得解情况

算法	Eil51 问题		St70 问题		Ch150 问题		Gil262 问题	
	平均	最优	平均	最优	平均	最优	平均	最优
3	464	448	731	718	7 083	6 725	2 877	2 618
4	462	445	739	714	7 179	6 704	2 795	2 626
5	477	447	723	712	7 062	6 664	2 639	2 564
5/3	102.8%	99.8%	98.9%	99.2%	99.7%	99.1%	91.7%	97.9%
5/4	103.3%	100.5%	97.8%	99.7%	98.4%	99.4%	94.4%	97.6%

综合分析表 6 和表 7 可见, 本文所提出的改进算法 (算法 5) 更适用于求解大规模的 TSP.

4 结 论

本文提出了一种基于粒子群参数优化的改进蚁群算法, 该算法给出了一种全局异步与精英策略相结合的信息素更新方式, 并合理确定了蚁群算法被粒子群算法调用时的迭代代数. 仿真结果表明, 该算法同已有算法相比, 具有以下特点:

1) 在解的质量相差较小时, 可以缩短搜索时间;

2) 在同一时间尺度下, 可求得质量较高的解;

3) 在处理较大规模 TSP 时优越性更为明显.

对于本文所提出的改进算法, 还需在以下几方面进行更加深入的研究:

1) 对 TSPLIB95 标准库中其余 71 个问题进行更全面的统计实验, 以充分说明本文所提算法的普适性;

2) 针对图 2 和图 4 中“十字”交叉的现象, 可引入适当的局部优化策略进一步提高解的质量.

参考文献(References)

[1] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies[C]. Proc of European Conf on Artificial Life. Paris: Elsevier Press, 1991: 134-142.

[2] Colorni A, Dorigo M, Maniezzo V. An investigation of some properties of an “Ant algorithm”[C]. Proc of the Parallel Problem Solving From Nature Conference. Brussels: Elsevier Press, 1992: 509-520.

[3] Dorigo M, Maniezzo V, Colorni A. The ant system: Optimization by a colony of cooperating agents[J]. IEEE Trans on Systems, Man and Cybernetics, 1996, 26(1): 29-41.

[4] 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): 53-56.

[5] Stützle T, Gambardella M. Distributed optimization by ant colonies[C]. Proc of the Int Conf on Artificial Neural Networks and Genetic Algorithms. Norwich: Czech Press, 1997: 245-249.

[6] Eberhart R C, Kennedy J. Particle swarm optimization[C]. Proc of IEEE Int Conf on Neural Networks. Nagoya: IEEE Press, 1995: 39-43.

[7] Kennedy J, Eberhart R C. A new optimizer using particles swarm theory[C]. Proc of 6th Int Symposium on Micro Machine and Human Science. Perth: IEEE Press, 1995: 1942-1948.

(下转第883页)