文章编号: 1001-0920(2012)12-1816-06

求解TSP的人工萤火虫群优化算法

周永权[a,b], 黄正新[a]

(广西民族大学 a. 信息科学与工程学院，b. 广西省混杂计算与集成电路设计分析重点实验室，南宁 530006)

摘 要: 人工萤火虫群优化算法是一种新型群体智能算法, 已在复杂多目标函数优化方面得到了成功的应用, 并表现出良好的性能. 为了充分发挥人工萤火虫群优化算法的优点, 将该算法与C2Opt算子相结合, 设计了求解旅行商问题(TSP)的一个新的高效人工萤火虫群优化算法, 并用其求解TSP这一经典的NP难问题. 通过对比TSP实例测试, 所得结果表明, 所提出算法在种群规模较小、迭代次数较少的情况下可以收敛到已知的最优解.

关键词: 人工萤火虫算法；荧光素；旅行商问题；C2Opt算子；组合优化

中图分类号: TP183 **文献标志码:** A

Artificial glowworm swarm optimization algorithm for TSP

ZHOU Yong-quan[a,b], *HUANG Zheng-xin*[a]

(a. College of Information Science and Engineering，b. Guangxi Key Laboratory of Hybrid Computation and Integrated Circuit Design Analysis，Guangxi University for Nationalities，Nanning 530006，China. Correspondent：ZHOU Yong-quan，E-mail：yongquanzhou@126.com)

Abstract: Artificial glowworm swarm optimization(AGSO) algorithm is proposed as a new bionic swarm of intelligent algorithm, which is successfully applied to complicated function optimization, and shows good performance. In order to give full play to the advantages of glowworm swarm optimization, the proposed algorithm is combined with C2Opt operator, and a new efficient AGSO algorithm about travelling salesman problem(TSP) is designed. The numerical experiment results show that the proposed algorithm can find the global optimal solution with less computation and evolving time.

Key words: artificial glowworm swarm algorithm；luciferin；TSP；C2Opt operation；combination optimization

1 引 言

旅行商问题(TSP)是组合优化领域中著名的问题之一. 设有一个旅行商人要拜访 N 个城市, 他必须选择所要走的路径, 路经的限制是每个城市只能拜访一次, 而且最后要回到原来出发的城市. 路径的选择目标是要求走过路径为所有路径之中最短的.

TSP问题所选择的路径数目与城市规模 N 呈指数型增长关系. 当城市规模较小时, 可用传统方法如枚举法获得最短路径; 但当城市规模较大时, 求解问题需要的时间复杂度很大, 导致无法获得其全局最优解. 目前, TSP广泛地应用于交通运输、物流配送、车间调度安排、电路板的布局、路由器的分布、机器人的控制等领域. 求解TSP一直是组合优化领域研究的热点和难点问题之一. 迄今求解TSP的算法有很多, 如模拟退火法、蚁群算法、遗传算法、量子算法以及免疫算法等, 然而这些方法存在着计算存储量大、运行时间长、求解城市规模太小等不足.

人工萤火虫群优化算法(AGSO)是近年来提出的一种新型群智能算法[1-5], 该算法已在复杂多目标函数优化、感应器噪音处理、信号源定位、有害气体泄漏检测[1]、多模态函数优化[6]、聚类分析[7]、数值优化计算和0-1背包问题[8]等方面得到成功的应用, 表现出了良好的性能. 为了充分发挥人工萤火虫群优化的优点, 本文将人工萤火虫群优化算法与C2Opt算子[9]相结合, 设计了求解TSP的一种新的高效人工萤火虫群优化算法. 通过对TSPLIB标准库中多个实例进行对比测试分析(http://www.iwr.uni- heidelberg.de/groups/comopt/software/TSPLIB95/tsp/), 所得结果表明, 本文所提出的算法在求解较大规模TSP问题时非常奏效, 具有收敛速度快、精度高等特点.

收稿日期: 2011-06-28；**修回日期:** 2011-09-04.

基金项目: 国家自然科学基金项目(61165015)；广西省自然科学基金项目(2012GXNSFDA053028)；智能感知与图像理解教育部重点实验室开放基金项目(IPIU012011001)；广西高等学校重大科研项目(2012ZD008).

作者简介: 周永权(1962−), 男, 教授, 博士, 从事神经网络、计算智能等研究；黄正新(1986−), 男, 硕士, 从事群智能算法及其应用的研究.

2　GSO 算法和 C2Opt 算子简述

2.1　GSO 算法

人工萤火虫群优化算法是受自然界萤火虫通过发光吸引同伴求偶或觅食行为的启发而提出的一种新型群智能优化算法. 一般GSO 算法实施过程描述如下[3]: 设 $x_i(t)$ 表示 t 时刻第 i 只萤火虫的位置, $f(x)$ 为适应度函数, $l_i(t)$ 表示 t 时刻第 i 只萤火虫的荧光素浓度, 萤火虫移动根据下式更新:

$$l_i(t) = (1-\rho)l_i(t-1) + \gamma f(x_i(t)). \tag{1}$$

其中: ρ 为荧光素挥发系数; γ 为荧光素增强因子, 即适应度提取比例. 设 r_s 表示萤火虫感知范围; $r_d^i(t)$ 表示 t 时刻第 i 只萤火虫的动态决策范围, 且 $0 < r_d^i(t) < r_s$, 并按下式进行调整:

$$r_d^i(t+1) = \min\{r_s, \max\{0, r_d^i(t) + \beta(n_t - |N_i(t)|)\}\}. \tag{2}$$

其中: β 为邻域变化率; n_t 为邻居阈值, 控制萤火虫的邻居数目; $N_i(t)$ 为 t 时刻第 i 只萤火虫的邻居集合, 由下式决定:

$$N_i(t) = \{j : \|x_j(t) - x_i(t)\| < r_d^i(t); l_i(t) < l_j(t)\}. \tag{3}$$

这里 $\|x\|$ 表示 x 的范数. 萤火虫在移动过程中, 据其邻居集合中各萤火虫的荧光素浓度来决定其移动方向. 设 $P_{ij}(t)$ 表示 t 时刻第 i 只萤火虫向其邻居集合中第 j 只萤火虫移动的概率, 通过下式计算:

$$P_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum\limits_{k \in N_i(t)} l_k(t) - l_i(t)}. \tag{4}$$

由移动概率 $P_{ij}(t)$, 采用轮盘法选择第 i 只萤火虫移动的方向, 其中 l_0 为荧光初始值. 设移动步长为 s, 则由下式确定第 i 只萤火虫在 $t+1$ 时刻的位置:

$$x_i(t+1) = x_i(t) + s\Big(\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}\Big). \tag{5}$$

2.2　Complete 2-Opt (C2Opt) 算子

在遗传算法求解 TSP 问题中, 通常随机选择路径上 20% ∼ 30% 的城市用 2-Opt 进行局部优化, C2Opt 是对构成路径的所有城市进行 2-Opt 优化. 下面以图 1 和图 2 为例, 说明 C2Opt 算子的操作过程.

设 $c_k(k = 0, 1, \cdots, n-1)$ 表示城市所在顶点, $d(c_i, c_j)$ 表示任意两个城市 c_i 和 c_j 之间的距离. C2Opt 算子实现的步骤描述如下[9]:

Step 1: 选取一路径

$$c = \{c_0, c_1, \cdots, c_i, c_{i+1}, \cdots, c_j, c_{j+1}, \cdots, c_{n-1}\},$$

如图 1 所示. 开始时, 令 $i = j = 0$.

Step 2: 选取一条边, 标记为 No.1: (c_i, c_{i+1}), 其中 $i < n$.

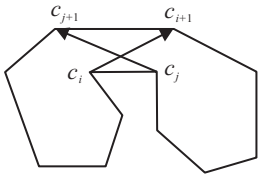


图 1　使用 C2Opt 算子前

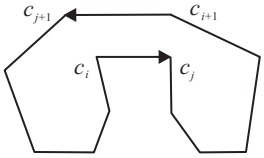


图 2　使用 C2Opt 算子后

Step 3: 选取另一条边, 标记为 No.2: (c_j, c_{j+1}), 其中 $j < n$.

Step 4: 若 $|j - (i+1)| \geqslant 2$ 且 $d(c_i, c_j) + d(c_{i+1}, c_{j+1}) < d(c_i, c_{i+1}) + d(c_j, c_{j+1})$, 则用 2-Opt 算子删除边 (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}) 且分别以相反箭头指向顶点 c_{i+1} 和 c_j.

Step 5: 以顶点 c_j 作为 No.2 边遍历开始的城市, 置 $j = j+1$, 重复执行 Step 3 和 Step 4, 直到 $j = n-1$. 若 $j = n-1$, 则 $j+1 = 0$.

Step 6: 以顶点 c_i 作为 No.1 边遍历开始的城市, 置 $i = i+1$, 重复执行 Step 2 ∼ Step 5, 直到 $i = n-1$. 若 $i = n-1$, 则 $i+1 = 0$.

Step 7: 重复执行 Step 2 ∼ Step 6, 直到 N 次, 使 N 足够大, 直到所选取路径无交叉边出现为止 (如图 2 所示).

3　求解 TSP 的人工萤火虫群优化算法

为方便求解 TSP, 先对 GSO 算法加以改进. 在 GSO 算法中, 萤火虫在其动态域内按概率大小移向荧光素值比自己大的个体. 在求解 TSP 问题中, 每只萤火虫在当前城市据荧光素值的大小选择下一个城市, 然后根据经过的路径适应度值更新路径荧光素值, 使得最终大多数萤火虫都在荧光素值最大的路径上飞行, 即为所求得的最优路径. 为便于描述, 本文给出以下定义:

定义 1　所有与城市编号 n 相连的城市组成的集合, 称为 n 的邻城市集, 记为 $\mathrm{ne}\{n\}$.

定义 2　萤火虫 i 在 t 循环中没有到过的城市组成的集合, 称为 i 的当前动态城市集, 记为 $\mathrm{rd}\{t, i\}$.

定义 3　假设萤火虫 i 在 t 循环中当前所在城市编号为 n_i, 则称 $\mathrm{ne}\{n_i\}$ 与 $\mathrm{rd}\{t, i\}$ 的交为 i 当前决策城市集, 记为 $\mathrm{ds}\{t, i\}$.

在 GSO 算法中, 萤火虫在其动态域内飞向概率最大的个体, 这种移动是萤火虫个体之间的移动. 在求解 TSP 问题中, 萤火虫是从当前城市飞向 $\mathrm{ds}\{t, i\}$

里的某个城市, 实现位置更新. 这里用一矩阵来记录城市间所有路径的荧光素值, 当萤火虫到达一个城市时, 它可以从矩阵中读取通向 $\mathrm{ds}\{t,i\}$ 各城市路径上的荧光素值, 并计算飞向各城市的概率. 为了扩大路径搜索范围, 本文对移动目标选择机制进行了修改, 改为按轮盘赌选择机制选择城市, 并沿城市所对应的路径移动. 当 $\mathrm{rd}\{t,i\}$ 为空时, 萤火虫 i 在 t 循环中找到了一条路径, 根据所选路径适应度值更新路径的边对应的矩阵元素的值, 然后继续下一次循环飞行.

在GSO算法中, 萤火虫每次按步长飞向其动态决策域内荧光素值比自己大的个体. 在求解TSP问题中, 萤火虫需要在城市之间移动, 每一次移动都会转移到下一个城市所在位置, 因所有城市之间的路径不相同, 在这里不设置移动步长, 每一次移动步长等于该移动所对应的两个城市间距离.

萤火虫个体选择飞行目标的概率公式只与个体间荧光素值的大小有关, 而与距离没有关系. 在求解TSP问题中, 选择移动目标与路径(距离)关系很密切. 本文对萤火虫选择移动目标的概率公式进行了改进, 其计算公式为

$$P_{n,j\in\mathrm{ds}\{t,i\}}=\frac{l(n_i,j)}{\sum\limits_{k,j\in\mathrm{ds}\{t,i\}}d(n_i,k)+d(n_i,j)^2}. \tag{6}$$

其中: n_i 为萤火虫当前所在的城市编号, $P_{n_i,j}$ 为萤火虫 i 从当前城市 n_i 飞向城市 j 的概率, $l(n_i,j)$ 为城市 n_i 与 j 边上的荧光素值, $d(n_i,j)$ 为城市 n_i 与 j 间的距离.

在GSO算法中, 初始化萤火虫位置结束时, 其所在位置对应于一个可行解, 因此可直接对该位置进行评价, 更新其荧光素值. 在求解TSP问题中, 初始化萤火虫位置结束时, 其所在位置不构成问题的一个可行解, 所以萤火虫需要完成飞行一圈才能得到一条可行路径, 进而对其进行评价, 更新该路径的荧光素值. 因此, 初始化萤火虫位置后, 萤火虫先选择路径飞行, 再据所选路径适应度进行荧光素值更新. 每只萤火虫选择的路径按以下2个公式更新其荧光素值:

$$l^{t+1}(m_1,m_2)=(1-\rho)^{*}l^{t}(m_1,m_2)+\frac{l_0}{d(m_1,m_2)^2}\times\left(\frac{\mathrm{best}}{ls^{t}(i)}\right)^2, \tag{7}$$

$$l^{t+1}(m_1,m_2)=l^{t+1}(m_2,m_1)=(l^{t+1}(m_1,m_2)+l^{t+1}(m_2,m_1))/2. \tag{8}$$

没有萤火虫经过的路径可令 $l_0=0$, 即可记录的荧光素为0, 此时只进行挥发更新. 为了扩大路径搜索范围, 提高找到最优路径的概率, 这里采用轮盘赌选择移动目标城市. 经验表明, 由具有自适应性的随机搜索性能的算法和局部启发式搜索算法混合构成新算法, 可在提高算法收敛速度和寻求全局最优解之间找到一个较好的平衡点[10]. 因此, 在算法执行到一定代数后用C2Opt算子进行局部优化, 以提高GSO算法求解TSP问题的效率. 另外, 为了减少执行C2Opt算子操作的代价, 先对所有萤火虫找到的路径进行排序; 然后对前面最优部分使用C2Opt算子操作调整优化, 重复路径只选其中一条; 最后, 用优化后的新路径替代部分较差路径. 这种替代意味着在更新萤光素值时, 与未被选择的路径边一样, 本次迭代所得到的部分较差路径边只进行挥发更新, 相对减少了在下一次迭代中随机选择这些路径边的概率. 这样做可相对增加选择未被选中的边的概率, 扩大路径搜索范围; 另一方面, 可以保留更多较优的路径, 有利于加快算法收敛到最优(最短)的路径.

有了以上对GSO算法的改进分析, 求解TSP问题的GSO算法实施步骤如下.

Step 1: 初始化 N 个城市位置萤火虫数量 M, 计算两两城市间距离 $d(i,j)$ 和

$$\mathrm{ne}\{n\},\ n=1,2,\cdots,N,$$

$$\text{While } t\leqslant \mathrm{itermax}.$$

Step 2: 每只萤火虫寻找最优路径.

Step 2.1: 随机将 M 只萤火虫放置到 N 个城市上;

Step 2.2: 对每一只萤火虫 i 用式(6)计算其飞向 $\mathrm{ds}\{t,i\}$ 中各个城市的概率, 然后用轮盘赌方法随机选择移动到下一个城市, 并将选择的城市编号从 $\mathrm{rd}\{t,i\}$ 中删除;

Step 2.3: 重复Step 2, 直到 $\mathrm{rd}\{t,i\}$ 为空.

Step 3: 进行路径评价.

Step 3.1: 计算路径的长度;

Step 3.2: 当 $t\leqslant$ itermax (最大迭代次数)时, 调用C2Opt算子进行局部调整优化;

Step 3.3: 比较 t 最优路径长度与当前最优值, 记录较小值为当前最优值.

Step 4: 进行荧光素更新.

Step 4.1: 对每只萤火虫飞行路径用式(7)和(8)进行荧光素更新;

Step 4.2: 其他路径只进行挥发计算, $t=t+1$.

4 实验结果分析

4.1 实验环境与参数设置

实验环境为CPU T4400 2.20 GHz, 内存2048 MB, 操作系统为Windows XP. 编程软件为Matlab 2008a. 所有实例荧光素初始值均为5, 其他参数设置如表1所示.

表 1　实验参数设置

实例名称	Dantzig 42	Eil 51	Berlin 52	St 70	Kroa 100	Krob 100	Lin 105	Pr 107	Pr 124	Pr 136	Pr 152	Pr 226
l_0	30	30	20	30	10	10	10	10	10	10	30	10
ρ	0.4	0.38	0.4	0.35	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
N	42	51	52	70	100	100	105	107	124	136	152	226
M	28	34	34	46	83	83	87	89	103	113	126	188
itermax	100	100	100	100	120	120	120	120	120	120	120	120

针对求解不同规模的TSP问题, 对表1中参数 l_0, ρ, N, M, itermax 随机选取不同值, 通过多次实验观察参数对数值仿真结果的影响, 在所有实验结果中取最短路径所对应的参数值作为本文实验的参数取值.

4.2　结果对比分析

为了测试本文提出的算法在求解TSP问题上的性能, 选用多个TSPLIB标准库实例进行实验测试, 并将测试所得到的结果与其他算法结果进行对比分析.

表2给出了本文算法在实例Dantzig42, Eil51, Berlin52和Pr107的实验结果与ACS+2Opt算法的实验结果比较. 其中最优值项是最近文献提供的最优值或TSPLIB标准库提供的最优值.

表 2　本文算法实验结果与 ACS+2Opt 结果的对比

实例	最优值	TSPLB 提供的最优路径长度	本文算法			ACS+2Opt[4]		
			计算最好结果	偏差率	迭代次数	计算最好结果	偏差率	迭代次数
Dantzig42	699 [11]	N/A	679.201 9	–	100	700	0.014 3	450
Eil51	426	429.983 3	428.871 8	–	100	431	0.023 7	900
Berlin52	7542[11]	7544.365 9	7 544.365 9	0.0	100	7 573	0.038 0	1 200
Pr107	44 303[11]	N/A	44 337.363 8	0.007 8	120	44 352	0.011 1	2 600

TSPLB提供的最优路径长度项值是用Matlab 2008a计算TSPLB标准库提供的当前最优路径的长度值. 偏差率项计算公式为

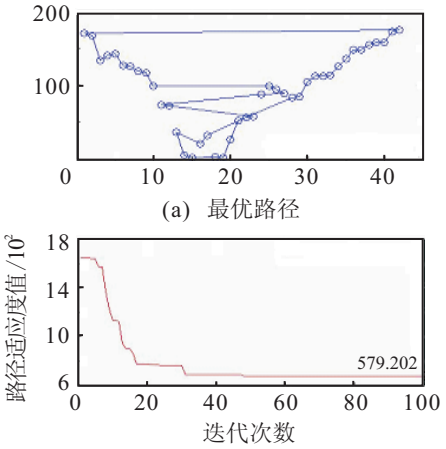
$$\text{偏差率} = \frac{\text{计算最好结果} - \text{TSPLIB 提供的最优路径长度}}{\text{TSPLIB 提供的最优路径长度}} \times \%. \tag{9}$$

如果TSPLB提供的最优路径长度项值是N/A, 则使用最优值项的值替代计算. 当计算最好结果项值小于TSPLB提供的最优路径长度项值时, 对应项值为"−", 表示找到更优的路径. 表2中符号N/A表示文献没有提供该项数值.

从表2中可以看出, 在这4个实例中本文算法比ACS+2Optcite [11] 能够找到更优的路径. 在实例Dantzig42中, 本文算法找到的最优路径长度为679.2019, 比文献[11]提供的当前最优结果699少了19.798 1. 图3为实例Dantzig42的实验仿真图. 其中: 上部分是当前最佳路径图, 下部分是路径适应度随迭代变化曲线图.

在实例Eil51中, 本文算法找到的最优路径长度为428.871 8, 比当前TSPLIB标准库提供的最优路径长度429.983 3少了1.111 5. 图4为实例Eil51的实验仿真图.

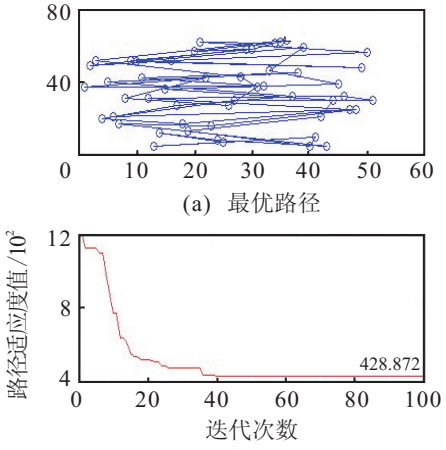
在实例Berlin52中, 本文算法找到的最优解与当前最优解一样.



(a) 最优路径

(b) 路径适应度随迭代变化曲线

图 3　实例 Dantzig42 的实验最优路径及路径适应度随迭代变化曲线



(a) 最优路径

(b) 路径适应度随迭代变化曲线

图 4　实例 Eil51 的实验最优路径及路径适应度随迭代变化曲线

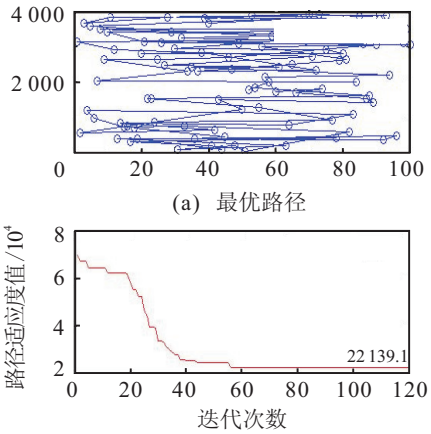
除实例Pr107外, 本文算法都能获得当前最优路径或者更优的路径. 相比于ACS+2Opt算法, 本文算法使用的迭代次数更少, 计算结果更优. 随着城市规模增大, ACS+2Opt所需要的迭代次数明显比本文算法多. 在实例Pr107中, ACS+2Opt算法运行了2 600次, 接近本文算法迭代次数的22倍; 而本文算法找到了比它更好的路径. 可见, 本文算法求解TSP问题时表现出的性能比ACS+2Opt算法好.

表3是本文算法所得到的最好结果与其他算法最好结果的对比. 表中最优值项、TSPLB提供的最优路径长度项、计算最好结果项和偏差率项的表示意义与表2相同.

表 3 本文算法实验最好结果与其他算法最好结果的对比

实例	最优值	TSPLB最优路径长度	本文算法		文献[13]		ACS[12]		ACOMGR[12]	
			计算最好结果	偏差率	计算最好结果	偏差率	计算最好结果	偏差率	计算最好结果	偏差率
St70	675	678.597 5	677.109 6	–	677.109 6	–	N/A	N/A	N/A	N/A
Kroa100	21 282	21 285.443 2	21 285.443 2	0.0	21 285.443 2	0.0	N/A	N/A	N/A	N/A
Krob100	22 141[13]	N/A	22 139.074 6	–	22 139.074 6	–	N/A	N/A	N/A	N/A
Lin105	14 379	14 382.996 0	14 382.996 0	0.0	N/A	N/A	N/A	N/A	N/A	N/A
Pr124	59 030[12]	N/A	59 030.735 7	0.000 1	N/A	N/A	60 702.4	0.283	59 091.8	0.01
Pr136	96 772[12]	N/A	97 684.416 0	0.942 9	N/A	N/A	99 324.6	2.637	96 916.5	0.149
Pr152	73 682[12]	N/A	73 683.640 6	0.002 2	N/A	N/A	77 969.7	5.81	73 881.7	0.27
Pr226	80 369[12]	N/A	80 422.977 8	0.067 2	N/A	N/A	85 254.5	6.079	8 0442.4	0.091 3

从表3可以看出, 在实例Krob100实验中, 本文算法找到的最优路径长度为22 139.074 6, 比当前TSPLB提供的最优值22 141少了1.925 4. 图5为实例Krob100的实验仿真图.



(a) 最优路径

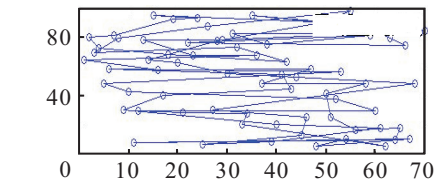
(b) 路径适应度随迭代变化曲线

图 5 实例Krob100的实验最优路径及路径长度随迭代变化曲线

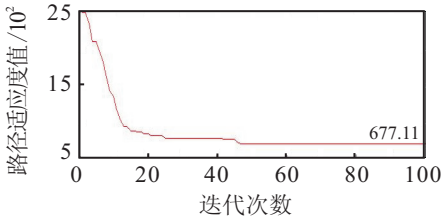
在实例St70实验中, 虽然本文算法没有找到比当前TSPLIB标准库提供的最优值更优, 但找到的最优路径长度为677.109 6, 比当前TSPLIB标准库提供的最优路径长度678.597 5少了1.487 9, 且比该最优路径更好. 图6为实例St70的实验仿真图.

实例Kroa100中, 本文算法与文献[13]的算法一样, 也找到了当前TSPLIB库提供最优路径.

与ACS[12]相比, 本文算法在8个TSP实例中均找到了最优路径; 而ACS在TSP实例St70, Kroa100, Krob100和Lin105中均未找到最优路径, 虽在Pr124, Pr136, Pr152和Pr226中找到了最优路径, 但其偏差率均比本文算法大. 此外, 通过其他实验可以发现, 随着



(a) 最优路径



(b) 路径适应度随迭代变化曲线

图 6 实例St70的实验最优路径及路径长度随迭代变化曲线

TSP问题规模的增加, ACS在解决此类问题时没有本文算法效果好.

在其余4个实验中, 尽管本文算法与文献[12]的算法都没有找到当前已知最优路径, 但除了实例Pr136以外, 本文算法比文献[12]的算法均获得了更好的结果. 例如本文算法求解实例Pr124的最好结果与最优值偏差率为0.000 1, 分别比文献[12]采用经典的ACS算法和ACOMGR算法减少了0.282 9和0.009 9; 实例Pr152的偏差率为0.002 2, 分别比经典的ACS算法和ACOMGR算法减少了5.807 8和0.267 8; 实例Pr226的偏差率分别比经典的ACS算法和ACOMGR算法减少了6.011 8和0.024 1.

通过上述实验结果对比分析可知, 本文提出的算法都能找到当前TSPLIB标准库所给出的最优路径值. 此外, 对TSPLIB标准库中其他实例进行了随机测试, 也能找到最优的路径值, 表明本文算法在求解

TSP 问题时非常奏效.

5 结　论

本文提出了一种求解 TSP 的人工萤火虫群优化算法, 拓展了人工萤火虫算法的应用领域. 给出了一种新的荧光素值更新公式, 特别针对算法运行后期收敛速度慢的特点, 在算法中引入了 C2Opt 算子进行局部优化, 极大地提高了人工萤火虫群优化算法求解 TSP 问题的效率. 研究表明, 该算法在解决组合优化及相关的问题上具有重要的应用价值.

参考文献(References)

[1] Krishnand K N, Ghose D. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics[C]. Proc of IEEE Swarm Intelligence Symposium. Piscataway: IEEE Press, 2005: 84-91.

[2] Krishnand K N, Ghose D. Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications[J]. Multiagent and Grid System, 2006, 2(3): 209-222.

[3] Krishnand K N, Ghose D. Glowworm swarm optimisation: A new method for optimising multi-modal functions[J]. Int J of Computational Intellingence Studies, 2009, 1(1): 93-119.

[4] Krishnand K N, Ghose D. Theoretical foundations for rendezvous of glowworm-inspired agent swarm at multiple locations[J]. Robotics and Autonomous Systems, 2008, 56(7): 549-569.

[5] Krishnand K N, Ghose D. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions[J]. Swarm Intell, 2009, 3: 87-124.

[6] 李咏梅, 周永权, 姚祥光. 基于追尾行为的改进型人工萤火虫群算法[J]. 计算机科学, 2011, 38(3): 248-251.
(Li Y M, Zhou Y Q, Yao X G. Improved glowworm swarm optimization based on the behavior of follow[J]. Computer Science, 2011, 38(3): 248-251.)

[7] Huang Zhengxin, Zhou Yongquan. Using glowworm swarm optimization algorithm for clustering analysis[J]. J of Convergence Information Technology, 2011, 6(2): 78-85.

[8] Yang Yan, Zhou Yongquan, Gong Qiaoqiao. Hybrid artificial glowworm swarm optimization algorithm for solving system of nonlinear equations[J]. J of Computational Information Systems, 2011, 6(10): 3431-3438.

[9] Peng Gang, Ichiro Iimura, Shigeru Nakayama. An evolutionary multiple heuristic with genetic local search for solving TSP[J]. Int J of Information Technology, 2008, 14(2): 1-11.

[10] 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.
(Yang H, Kang L S, Chen Y P. A gene-based genetic algorithm for TSP[J]. Chinese J of Computer, 2003, 26(12): 1753-1758.)

[11] 冀俊忠, 黄振, 刘椿年. 一种快速求解旅行商问题的蚁群算法[J]. 计算机研究与发展, 2009, 46(6): 968-978.
(Ji J Z, Huang Z, Liu C N. A fast ant colony optimization algorithm for traveling salesman problems[J]. J of Computer Research and Development, 2009, 46(6): 968-978.)

[12] 冀俊忠, 黄振, 刘椿年. 基于多粒度的旅行商问题描述及其蚁群优化算法[J]. 计算机研究与发展, 2010, 47(3): 434-444.
(Ji J Z, Huang Z, Liu C N. An ant colony algorithm based on multiple-grain representation for the traveling salesman problems[J]. J of Computer Research and Development, 2010, 47(3): 434-444.)

[13] 蔡之华, 彭锦国, 高伟, 等. 一种改进的求解 TSP 问题的演化算法[J]. 计算机学报, 2005, 28(5): 823-829.
(Cai Z H, Peng J G, Gao W, et al. An improved evolutionary algorithm for the traveling salesman problem[J]. Chinese J of Computer, 2005, 28(5): 823-829.)

[14] Gong Qiaoqiao, Zhou Yongquan, Yang Yan. Artificial glowworm swarm optimization algorithm for solving 0-1 knapsack problem[J]. Advanced Materials Research, 2011, 143-144: 166-171.