计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(8):219−225 [doi: 10.15888/j.cnki.csa.006497]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于改进萤火虫算法求解旅行商问题①

王　艳, 王秋萍, 王晓峰

(西安理工大学 理学院, 西安 710054)
通讯作者: 王秋萍, E-mail: wqp566@sina.com

摘　要: 鉴于 TSP 问题是古老的组合优化难题, 而萤火虫算法在求解函数优化问题中表现出优良的性能, 因此, 本文利用改进的萤火虫算法求解 TSP 问题. 首先, 在分析了旅行商问题的特点后, 采用整数编码的方式来表示萤火虫的位置. 然后, 在标准萤火虫算法的位置更新过程中引入了对数递减的惯性权重来影响萤火虫的迭代过程, 同时结合了遗传算法中的选择, 交叉, 变异以及进化逆转操作来提高每一次迭代中种群的多样性及种群的搜索能力, 并将改进的算法解决 TSP 问题. 最后, 通过 Matlab 仿真实验表明改进的算法在求解 TSP 问题时具有更好收敛速度和优化效果.

关键词: 旅行商问题; 整数编码; 萤火虫算法; 惯性权重; 遗传算法

引用格式: 王艳,王秋萍,王晓峰.基于改进萤火虫算法求解旅行商问题.计算机系统应用,2018,27(8):219–225. http://www.c-s-a.org.cn/1003-3254/6497.html

Solving Traveling Salesman Problem Based on Improved Firefly Algorithm

WANG Yan, WANG Qiu-Ping, WANG Xiao-Feng

(Faculty of Sciences, Xi'an University of Technology, Xi'an 710054, China)

Abstract: Traveling Salesman Problem (TSP) is an oldest combinatorial optimization problem, and Firefly Algorithm (FA) shows excellent performance to complicated function optimization. Hence, in this study, we used improved FA to solve TSP. Firstly, after the characteristics of TSP are analyzed, and the method of integer encoding is adopted to set the position of fireflies. Then, the logarithmic adjustment factor is introduced in the standard FA. Meanwhile, we combine the crossover, mutation, and reverse operation in Genetic Algorithm (GA) to improve the population diversity and search ability of each iteration, and it is applied to solve TSP. Finally, the numerical experiments show that the proposed algorithm has faster convergence speed and optimization effect.

Key words: Traveling Salesman Problem (TSP); integer encoding; Firefly Algorithm (FA); inertia weight; Genetic Algorithm (GA)

旅行商问题 (Traveling Salesman Problem, TSP)[1,2] 是组合优化中典型的 NP 难题, TSP 有着很高的理论和实际价值, 在过去的几十年中成为智能计算、优化问题等重要算法思想的测试平台, 也被广泛应用于网络路由, 车辆调度, 机器人控制和计算机联网等领域. 其求解方法主要包括两个方向: 完全算法和近似算法[3]. 前者能保证找到最优解, 但时间复杂度高, 难以满足大规模运算需求, 如分支定界法、分支剪支[4,5]等. 后者只能找到近似解, 却能在多项式时间内结束, 如最近邻算法以及启发式算法. 如何将各种启发式算法有效结合起来, 成为广大学者关注和研究的热点.

2008 年剑桥学者 Yang[6]提出萤火虫算法 (Firefly

① 基金项目: 国家自然基金项目 (61772416)
Foundation item: National Natural Science Foundation of China (61772416)
收稿时间: 2018-01-04; 修改时间: 2018-01-23; 采用时间: 2018-02-01; csa 在线出版时间: 2018-07-28

Algorithm, FA), 该算法调整参数少, 收敛速度快, 是一种基于群体搜索的启发式算法. 与 2005 年印度学者 Krishnanand 和 Ghose 提出的萤火虫算法 (Glowworn Swarm Optimization, GSO) 不同的是: FA 算法在位置迭代更新式中加入扰动项$\alpha(rand-1/2)$, 避免算法陷入局部最优. 因此, 本文选择 FA 进行研究. 2010 年 Sayadi, Ramezanian 和 Ghaffari-Nasab[7]首次将离散萤火虫算法应用于组合优化中最小化流水调度问题, 实验结果表明该方法寻优精度优于蚁群优化算法, 反映出该算法应用于组合优化问题中的优越性, 至此, 离散萤火虫算法得到广泛关注, 在各个研究领域得到迅速的发展. Jati, Suyanto[8]于 2011 年将离散萤火虫算法用于解决 TSP 问题, 所提出的两种方案都找到了问题的最优解. 然而, 文中只是单一的用萤火虫算法解决旅行商问题, 没有与其他算法相结合. Jati, Manurung, Suyanto [9]于 2013 年提出了基于边缘运动的方案解决 TSP 问题, 所提方案运算速度快, 保证萤火虫能更接近比它更亮的个体, 但是没能改善所求问题的适应度函数值. 文献[10]将萤火虫算法进行离散化, 提出一种确定性和随机性相结合的方法生成初始种群, 同时采取混合种群迭代更新策略. 仿真结果表明算法能够加速全局收敛, 然而所提算法的平均最优值较 FA 变化不明显. 因此, 有必要进一步改进萤火虫算法来提高算法的性能.

基于此, 本文在 FA 算法的基础上, 采用整数编码, 接着对 FA 算法位置更新部分引入对数递减的惯性权重, 并结合遗传算法中选择、交叉、变异、逆转操作, 来调整随着算法迭代次数增加导致种群多样性下降的缺陷, 提出一种改进的萤火虫算法, 用于求解 TSP 问题, 并采用标准 TSPLIB 数据库中的算例和多种群智能优化算法对比仿真验证所提算法的有效性.

1 旅行商问题

TSP 是指给定 n 个城市的地点及每两城市间的距离, 某个旅行商以其中一个城市为起点出发, 访问给定的所有城市且仅访问一次, 再回到城市起点, 求最短路程的巡回方案. 其数学描述为[11]:

$$f(x)=\sum_{i=1}^{n-1}d(x_i,x_{i+1})+d(x_1,x_n) \tag{1}$$

其中, $f(x)$表示路径长度, $d(i,j)$表示城市 i 到城市 j 的距离, $x=(x_1,x_2,\cdots,x_n)$为一个访问的顺序, n 表示城市个数, 1, 2,$\cdots$,n表示城市的序号.

2 萤火虫算法

2.1 萤火虫算法

萤火虫算法的基本假设: 萤火虫是发光亮度跟当前的位置有关, 位置越好, 发光亮度越高. 此时它具有更大的吸引度, 从而能吸引其范围内亮度弱的其它萤火虫向其靠近, 而且它们之间的相对亮度与吸引度和距离成反比. 在算法的实现中, 利用问题的目标值来衡量萤火虫位置的好坏, 而用迭代式子模拟萤火虫的搜索飞行移动, 以此构造算法, 逐步向问题的最优解移动, 达到优化目的.

首先建立萤火虫 i 的绝对亮度I_i和目标函数$f(x)$之间的联系[12], 通常以目标函数值代表绝对亮度, 即 $I_i=f(x)$, $x=(x_{i1},x_{i2},\cdots,x_{id})$. 若萤火虫 i 的绝对亮度大于萤火虫 j 的绝对亮度, 则萤火虫 j 被萤火虫 i 吸引而向 i 移动. 萤火虫 i 对萤火虫 j 的吸引力为β_{ij}, 即:

$$\beta_{ij}=\beta_0\exp(-\gamma r_{ij}^2) \tag{2}$$

其中, β_0为最大吸引力, γ为光吸收系数, 通常取$\beta_0=1$, $\gamma\in[0.01, 100]$[10]; r_{ij}为萤火虫 i 到萤火虫 j 的笛卡尔距离, 即:

$$r_{ij}=\left\|x_i-x_j\right\|=\sqrt{\sum_{k=1}^{d}(x_{ik}-x_{jk})^2} \tag{3}$$

其中, d为变量的维数; x_i、x_j为萤火虫 i 和萤火虫 j 所处的空间位置.

由于萤火虫 j 被萤火虫 i 吸引而向 i 移动, 则萤火虫 j 的位置更新公式为:

$$x_j(t+1)=x_j(t)+\beta_{ij}(x_i(t)-x_j(t))+\alpha\varepsilon_j \tag{4}$$

其中, t 为迭代次数; β_{ij}为萤火虫 i 对萤火虫 j 的吸引力; α为常数, 一般可以取[0, 1]内的数; ε_j是由均匀分布得到的随机数向量.

2.2 萤火虫算法的改进 (Improved Firefly Algorithm, IFA)

萤火虫算法中存在容易陷入局部最优, 求解精度不高等问题. 而萤火虫的位置迭代在萤火虫优化过程中占举足轻重的地位, 过大的步长会导致萤火虫移动过程中跳过全局最优值, 搜索精度下降; 过小的步长会增加它的局部寻优能力而无法得到全局最优值, 且影响算法的执行效率. 此外, 种群的多样性大大影响萤火

虫寻优的速率和解的效果. 针对以上不足, 这里在算法的位置迭代和种群多样性方面作如下改进:

2.2.1 设计惯性权重ω来保证全局收敛性

下面是常见的惯性权重表示:

(1) Shi 和 Eberhart 提出了惯性权重线性递减策略, 将ω的初始值从 0.9 随迭代次数增加线性递减为 0.4, 调整公式为[13]:

$$\omega_k = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{iter_{\max}} \times iter \tag{5}$$

其中, $\omega_{\max}$、$\omega_{\min}$分别为ω的最大值和最小值; $iter$和$iter_{\max}$分别表示当前迭代次数和最大迭代次数; ω_k表示当前迭代权重值.

(2) Baykasoğlu, Ozsoydan[14]提出一种惯性权重正弦调整的粒子群算法, 该权值的调整使算法开始时, 粒子在附近做局部寻优, 在一定迭代次数后, 相互协作程度加大, 进行全局寻优, 最后让粒子进行局部寻优, 调整方案为:

$$\omega_k = \omega_{\min} + (\omega_{\max} - \omega_{\min})\sin(\pi t / iter_{\max}) \tag{6}$$

(3) 文献[15]中提出一种惯性权重线性递减的粒子群算法:

$$\omega_k = \omega_{\max} - \alpha \times (\omega_{\max} - \omega_{\min})\log_{iter_{\max}} k \tag{7}$$

其中, α为对数调整因子, 当$0 < \alpha < 1$时称为压缩因子, 当$\alpha > 1$时称膨胀因子. 图 1 是$\alpha = 1$, $iter_{\max} = 200$的线性递减惯性权重、正弦调整惯性权重和对数调整权重策略与迭代次数的关系曲线.

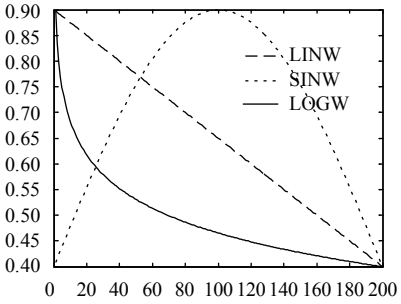


图 1 几种惯性权重策略与迭代次数关系曲线

式 (5), 式 (6), 式 (7) 引入萤火虫算法的位置更新公式变为 :

$$x_j(t+1) = \omega(t) * x_j(t) + \beta_{ij}(x_i(t) - x_j(t)) + \alpha\varepsilon_j \tag{8}$$

萤火虫位置迭代过程中, 要求在前期有较高的全局搜索能力, 如果在最优解附近ω值能迅速减小进入局部搜索, 就可以加快收敛速度. 图 1 可以看出, 惯性权重对数递减策略可以满足要求. 经文献[15]仿真表明, 引入的权重策略增大了种群搜索范围, 使复杂问题的解能够迅速达到全局最优.

2.2.2 基于 GA 的改进萤火虫算法

为了保持种群多样性, 提高算法的全局搜索能力, 在萤火虫进行位置更新后, 引入遗传算法 (Genetic Algorithm, GA)[16]中的选择、交叉、变异和进化逆转, 对萤火虫个体执行选择、交叉、变异和进化逆转操作.

(1) 选择操作: 从群体中以一定概率选择适应度函数值大的个体, 进行交叉, 变异, 逆转产生新个体.

(2) 交叉操作: 在萤火虫吸引度, 位置, 亮度更新后, 确定交叉操作的父代并随机选取的两个位置进行交叉操作, 当交叉后的萤火虫适应度优于原来的萤火虫时, 萤火虫位置更新, 否则萤火虫位置不变. 交叉采用个体位置互换的方法如下:

$$\begin{array}{r} [9\ 5\ 1\,|\,3\ 7\ 4\ 2\,|\,0\ 8\ 6] \xrightarrow{\text{交叉}} \\ [0\ 5\ 4\,|\,6\ 3\ 8\ 7\,|\,2\ 1\ 9] \\ \text{新个体}\ [9\ 5\ 1\,|\,6\ 3\ 8\ 7\,|\,0\ *\ *] \\ [0\ 5\ *\,|\,3\ 7\ 4\ 2\,|\,*\ 1\ 9] \end{array} \tag{9}$$

其中, 带*位置表示同一个体由于维数冲突不确定的数字, 此处采用部分映射进行消除冲突, 即用中间维数对应关系进行映射. 映射结果如下:

新个体 [9 5 1 6 3 8 7 0 4 2]

[0 5 8 3 7 4 2 6 1 9]

(3) 变异操作: 在萤火虫吸引度, 位置, 亮度更新后, 随机选取变异的 2 个位置, 进行位置互换, 当变异后萤火虫适应度优于原来的萤火虫时, 萤火虫位置更新, 否则萤火虫位置不变. 变异采取的个体变换方法如下:

若随机选择的个体位置为第 4 维和第 7 维, 变异后得到的个体为:

$$\begin{array}{r} [9\ 5\ 1\ 3\ 7\ 4\ 2\ 0\ 8\ 6] \xrightarrow{\text{变异}} \\ \text{新个体}\ [9\ 5\ 1\ 2\ 7\ 4\ 3\ 0\ 8\ 6] \end{array} \tag{10}$$

(4) 进化逆转操作. 为改善种群的多样性, 在选择、交叉和变异后引进逆转操作. 把进化逆转后适应度高的个体留下来, 适应度低的个体进化无效. 逆转采取的个体位置变换方法如下:

若随机选择的个体位置为第 4 维到第 7 维, 逆转后得到的个体为:

$$\begin{array}{r} [9\ 5\ 1\ 3\ 7\ 4\ 2\ 0\ 8\ 6] \xrightarrow{\text{逆转}} \\ \text{新个体}\ [9\ 5\ 1\ 2\ 4\ 7\ 3\ 0\ 8\ 6] \end{array} \tag{11}$$

3 改进的萤火虫算法求解旅行商问题

对包含 n 个城市的 TSP 问题, 这里采用整数编码表示萤火虫的位置[9], 并且设置萤火虫的迭代步长为整数, 以确保有效解决 TSP 问题. 每个萤火虫的位置对应一条巡回路径, 以最短路为目标函数, 运用 IFA 对目标函数进行求解, 即以式 (1) 作为目标函数, 寻找$f(x)$的最小值, 对应于路径的长度. 萤火虫的个体位置为一个 n 维矢量$x=(x_{i1},x_{i2},\cdots,x_{in})$, 其中$i=1,2,\cdots,M$代表种群数目, n 代表萤火虫的维数或所要访问的城市个数 (min=1, max=n). x_{ij}为萤火虫i的第j维元素. 由于每个城市必须遍历且只能遍历一遍, 因此x中任意两个元素互不相等. 具体步骤如下:

Step 1. 设定萤火虫群算法的参数值, 种群大小, 光强吸收系数, 迭代总次数 $nMax$ 等参数.

Step 2. 对萤火虫初始位置进行整数编码, 即用 $randperm(n)$ 给出种群中每只萤火虫初始位置.

Step 3. 计算萤火虫个体当前的目标函数值, 找出萤火虫当前的最优位置并记录对应的目标函数值.

Step 4. 利用式 (7) 更新权重, 计算惯性权值.

Step 5. 利用式 (8) 更新萤火虫位置, 计算位置更新后每个萤火虫的亮度$f(x)$.

Step 6. 如 2.2 进行选择, 交叉, 变异和逆转产生新个体.

Step 7. 进行终止条件判断, 达到预定的最大迭代次数 $nMax$, 则停止计算, 输出最优位置和最优函数值就是 TSP 的最短路线和最短路径值; 否则重复执行 Step 3–Step 6.

4 仿真实验

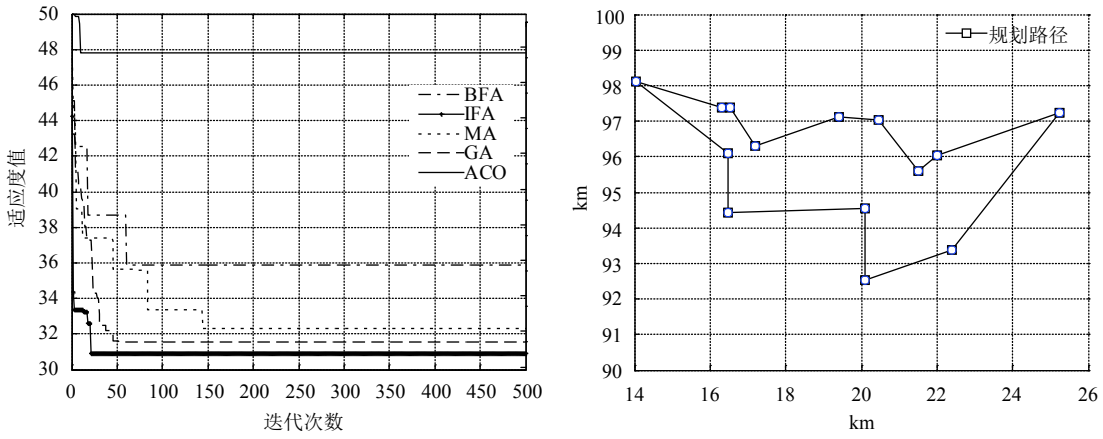
本文的实验平台是 Windows7 操作系统, 编程软件为 Matlab R2013a. 针对 TSPLIB 库中的 Burma14, Oliver30,ei151 和 st70 四组 Benchmark 城市坐标数据, 分别用基本 FA 算法[6,12], IFA 算法, 猴群算法 (Monkey Algorithm, MA)[17], GA 算法[16]和蚁群算法 (Ant Colony Optimization, ACO)[18]做 20 次实验. 求得实验的最优值, 最差值, 平均值, 最优迭代次数以及偏差率. 表 1 所示为 IFA 算法与 FA 算法、MA 算法、GA 算法和 ACO 算法用算例 Burma14, Oliver30, ei151 和 st70 的测试结果, 各算法的偏差率是根据公式 (9) 计算得到的数据.

$$\text{偏差率}=\frac{\text{计算最好结果}-\text{已知最优值}}{\text{已知最优值}}\times 100\% \quad (12)$$

图 2(a)–图 5(a) 分别表示各算法对这 4 组城市数据进行优化时各适应度值的变化趋势, 图 2(b)–图 5(b) 分别表示 IFA 优化 Burma14, Oliver30, ei151 和 st70 问题的最优路径, 从起点至终点表示该算法优化的最短距离.

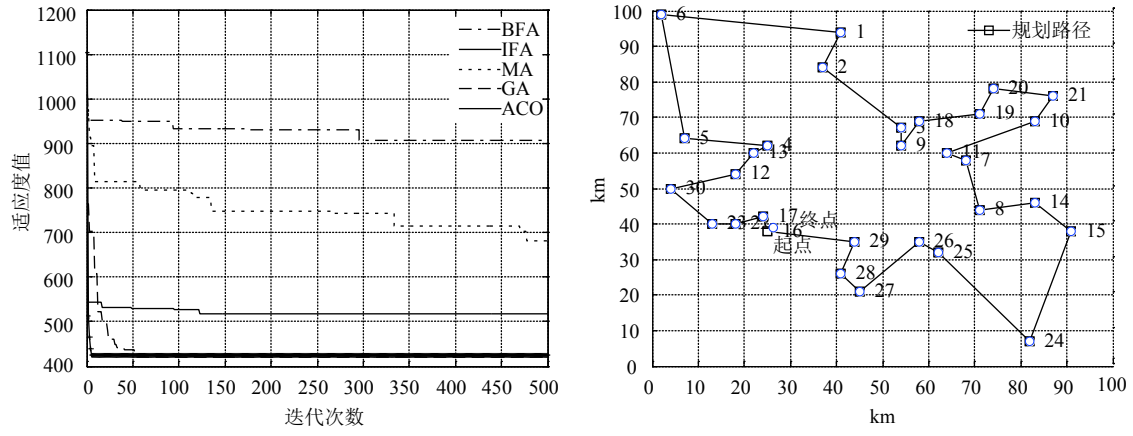
表 1 标准测试问题的计算结果

TSP 问题	已知最优值 (TSPLIB 路径长度)	算法	最优值	最差值	平均值	最优迭代次数	偏差率 (%)
Burma14	30.8785 (30.8785)	FA	34.4620	38.1748	36.2041	56	11.6
		IFA	**30.8785**	**30.8785**	**30.8785**	**2**	**0**
		MA	31.4562	32.1784	31.5944	143	1.9
		GA	31.6729	50.0382	44.7029	52	2.5
		ACO	44.4658	51.0679	46.4854	10	44.0
Oliver30	423.7406 (423.7406)	FA	678.7750	915.0141	879.7265	58	60.1
		IFA	**424.6976**	**426.7506**	**425.6496**	**17**	**0.2**
		MA	666.5300	714.0475	681.1855	286	57.2
		GA	384.6631	510.5752	453.7675	53	–9.2
		ACO	468.7001	468.7001	468.7001	33	10.6
eil51	426 (429.9)	FA	824.8429	1.2746e+03	1.0357e+03	76	93.6
		IFA	**428.8718**	**432.4828**	**429.9538**	**35**	**0.6**
		MA	696.6946	1.1090e+03	865.4473	251	63.5
		GA	721.1757	924.7538	797.9873	82	69.2
		ACO	740.7226	807.1753	771.1768	49	73.7
St70	675 (678.5975)	FA	1.8024e+03	2.9167e+03	2.1603e+03	136	166
		IFA	**683.6404**	**688.0342**	**685.6584**	**53**	**1.2**
		MA	1.5691e+03	2.6925e+03	2.2896e+03	274	132
		GA	1.4584e+03	1.8363e+03	1.6406e+03	125	116
		ACO	1.4319e+03	1.4413e+03	1.4382e+03	74	112



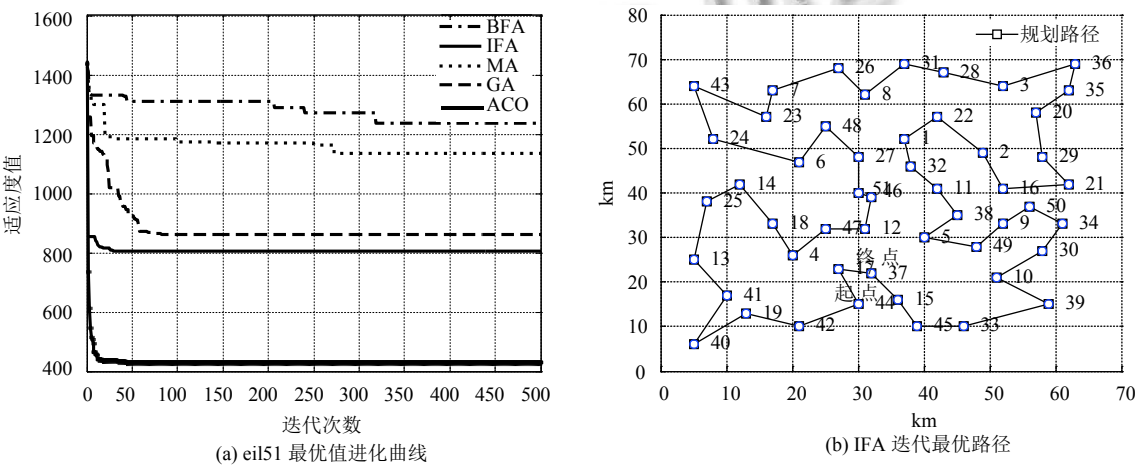
(a) Burma14最优值进化曲线 (b)IFA 迭代最优路径

图 2 实例 Burma14 的优化过程和最优路径图



(a) Oliver30最优值进化曲线 (b) IFA迭代最优路径

图 3 实例 Oliver30 的优化过程和最优路径图

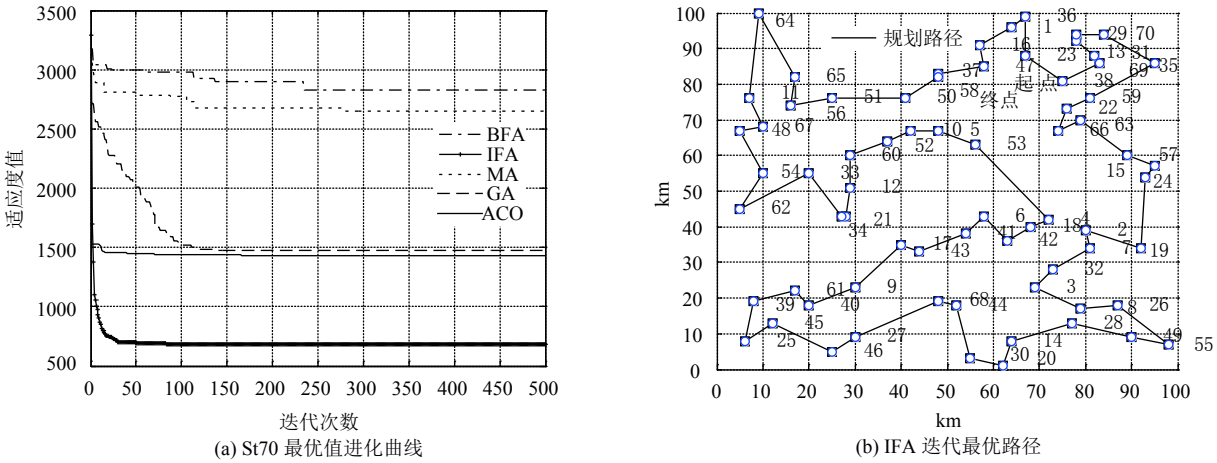


(a) eil51 最优值进化曲线 (b) IFA 迭代最优路径

图 4 实例 eil51 的优化过程和最优路径图

4.1 参数设置

各算法种群规模 M=100, 城市数目 n 由具体测试数据决定, 总迭代次数$nMax$ = 500, 其它参数依照相关文献的基本原则设置如下:



(a) St70 最优值进化曲线　　(b) IFA 迭代最优路径

图 5　实例 St70 的优化过程和最优路径图

BFA 和 IFA 算法的最大加权系数$\omega_{\max}=0.9$, 最小加权系数$\omega_{\min}=0.4$, 常数$\alpha=0.2$, 吸引力$\beta_{\min}=0.2$, 光照吸收率为$\lambda=1$.

MA 算法中爬步长$\alpha=1$, 爬的迭代次数$Nc=10$, “好动策略”的执行次数$Nn=10$, 视野长度$b=1$, 望的迭代次数$Nw=5$, 跳区间$[c,d]=[-1,1]$.

GA 算法中交叉概率$Pc=0.9$, 变异概率$Pm=0.05$, 代沟$GGAP=0.9$.

ACO 中信息素重要程度参数$\alpha=1$, 启发式因子$\beta=1$, 信息挥发素系数$\gamma=0.95$, 信息素增加强度系数$Q=100$.

4.2　实验结果对比

在 TSP 问题中, 针对 Burma14, 从表 1 可以看出, IFA 算法 100% 能找到最优解, 算法精度高, 迭代次数少, 性能优化效果明显. MA,GA 的偏差率都在 2% 内, 收敛速度较快, 但是迭代过程中波动较明显. ACO 算法的偏差率较高, 所需迭代次数少. 从图 2 可以看出, IFA 的收敛速度和收敛精度比其他 4 种优化算法更优; 针对 Oliver30, 由表 1 可以看出, BFA 和 MA 偏差率较大. GA 所得的实验最优值小于已知最优值和 TSPLIB 路径长度, 故得出的偏差率为负值, 同时, 其最优值和最差值相差较大, 说明算法迭代最优值不稳定. ACO 算法迭代次数少, 趋于最优值较平稳. 从图 3(b) 可以直观的看出巡回路径没有打结, 所去城市不重不漏, 以此可直观验证 IFA 算法的正确率; 针对 ei151, 从表 1 可以看出, FA 的偏差率最大, 随着城市规模的增大, FA 的迭代次数和偏差率比较显著. MA, GA, ACO 偏差率相对较大, 算法迭代稳定. IFA 算法随城市规模的增大, 最优值和偏差率变化不太显著, 说明算法的改进效果较好. 从图 4(a) 可以看出在迭代进行时, IFA 算法以相对较快的速度持续降低, 且快速搜索到了最优解; 针对 St70, 由表 1 可以看出, BFA, MA, GA, ACO 的偏差率都显著增加, GA 迭代平稳, 收敛速度快. 随城市规模的增大, IFA 的最优迭代次数和偏差率没有显著增加, 最优值, 最差值, 平均值相差较小, 可见 IFA 算法有较好的稳定性和鲁棒性. 从图 5(a) 可以看出, 当其他算法下降缓慢时, IFA 算法以较快的速度, 较高的下降趋势, 有效的搜索到全局最优解.

5　结论

本文针对 TSP 问题的特点, 对 FA 算法的离散化和种群多样性进行了改进, 提出了一种改进的萤火虫算法, 并采用该算法解决 TSP 问题. 仿真结果表明改进的萤火虫算法能有效地解决 TSP 问题, 保证种群的多样性, 同时算法有较好的收敛速度优于 MA, GA. 且 IFA 算法的最优解明显比 GA, ACO 更接近全局最优解, 所以说基于改进的萤火虫算法求解旅行商问题是有效的, 并且改进的算法的寻优能力, 稳定性更好, 收敛速度更快.

参考文献

1 Changdar C, Mahapatra GS, Pal RK. An efficient genetic algorithm for multi-objective solid travelling salesman problem under fuzziness. Swarm and Evolutionary Computation, 2014, 15: 27–37. [doi: 10.1016/j.swevo.2013.

11.001]
2 Caballero R, Hernández-Díaz AG, Laguna M, *et al*. Cross entropy for multiobjective combinatorial optimization problems with linear relaxations. European Journal of Operational Research, 2015, 243(2): 362–368. [doi: 10.1016/j.ejor.2014.07.046]
3 雷秀娟. 群智能优化算法及其应用. 北京: 科学出版社, 2012.
4 Smith SL, Imeson F. GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. Computers & Operations Research, 2017, 87: 1–19.
5 Kinable J, Smeulders B, Delcour E, *et al*. Exact algorithms for the equitable traveling salesman problem. European Journal of Operational Research, 2017, 261(2): 475–485. [doi: 10.1016/j.ejor.2017.02.017]
6 Yang XS. Nature-inspired metaheuristic algorithms. Frome: Luniver Press, 2008.
7 Sayadi MK, Ramezanian R, Ghaffari-Nasab N. A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. International Journal of Industrial Engineering Computations, 2010, 1(1): 1–10. [doi: 10.5267/j.ijiec]
8 Jati GK, Suyanto. Evolutionary discrete firefly algorithm for travelling salesman problem. Bouchachia A. Adaptive and Intelligent Systems. Austria: Springer-Verlag, 2011: 393–403.
9 Jati GK, Manurung R, Suyanto. Discrete firefly algorithm for traveling salesman problem: A new movement scheme. In: Yang XS, Cui ZH, Xiao RB, eds. Swarm Intelligence and Bio-Inspired Computation. London: Elsevier, 2013: 295–312.
10 吴宏超, 刘检华, 唐承统, 等. 基于萤火虫算法的管路系统布局序列优化技术. 计算机集成制造系统, 2016, 22(8): 1837–1848.
11 戚远航, 蔡延光, 蔡颢, 等. 旅行商问题的混沌混合离散蝙蝠算法. 电子学报, 2016, 44(10): 2543–2547. [doi: 10.3969/j.issn.0372-2112.2016.10.037]
12 赵玉新, Yang XS, 刘利强. 新兴元启发式优化方法. 北京: 科学出版社, 2013.
13 陈兆芳, 张岐山. 基于粒子群算法的电梯系统选择性维修模型. 计算机系统应用, 2015, 24(8): 229–233. [doi: 10.3969/j.issn.1003-3254.2015.08.041]
14 Baykasoğlu A, Ozsoydan FB. Adaptive firefly algorithm with chaos for mechanical design optimization problems. Applied Soft Computing, 2015, 36: 152–164. [doi: 10.1016/j.asoc.2015.06.056]
15 戴文智, 杨新乐. 基于惯性权重对数递减的粒子群优化算法. 计算机工程与应用, 2015, 51(17): 14–19, 52. [doi: 10.3778/j.issn.1002-8331.1412-0259]
16 Wang JQ, Ersoy OK, He MY, *et al*. Multi-offspring genetic algorithm and its application to the traveling salesman problem. Applied Soft Computing, 2016, 43: 415–423. [doi: 10.1016/j.asoc.2016.02.021]
17 徐小平, 张东洁. 一种改进的猴群算法. 计算机系统应用, 2017, 26(6): 193–197. [doi: 10.15888/j.cnki.csa.005822]
18 Saenphon T, Phimoltares S, Lursinsap C. Combining new fast opposite gradient search with ant colony optimization for solving travelling salesman problem. Engineering Applications of Artificial Intelligence, 2014, 35: 324–334. [doi: 10.1016/j.engappai.2014.06.026]