

基于优化成熟度的自适应蚁群优化算法

王海宁, 孙守迁

(浙江大学 计算机学院, 浙江 杭州 310027)

摘 要: 分析了蚁群算法局部信息素更新系数与全局信息素更新系数对算法寻优能力与收敛速度的关系, 定义平均路径相似度(ATS)来表征寻优过程的成熟程度, 并据此自适应调整信息素更新系数, 提高算法收敛速度并避免陷入局部最优. 经过与典型蚁群算法在多个旅行商问题测试用例上进行收敛速度与全局寻优能力的全面比较, 证明了新的算法具有较好的效果.

关键词: 蚁群优化; 平均路径相似度; 自适应参数控制

中图分类号: TP31

文献标识码: A

文章编号: 1000-7180(2010)08-0140-05

Adaptive Ant Colony Optimization Algorithm Based on Maturity

WANG Hai-ning, SUN Shou-qian

(School of Computer Science, Zhejiang University, Hangzhou 310027, China)

Abstract: By observing the effect of parameters on the performance of the ACS algorithm in different optimization state, this paper presents a novel version of ACS based on the optimization maturity for obtaining self-adaptive parameters control. The adaptive ACS has been applied to optimize several benchmark TSP instances. The solution quality, convergence rate and global searching ability are favorably compared with the ACS. Experimental results confirm that our proposed method is effective and outperforms the conventional ACS.

Key words: ant colony optimization; average tour similarity; adaptive parameters control

1 引言

蚁群优化(Ant Colony Optimization, ACO)是 M. Dorigo 受蚁群觅食行为的启发提出一种新型的群智能优化算法^[1-2], 该算法在解决著名的 NP 问题——旅行商问题(TSP)及一些复杂的优化组合问题上取得了很好的效果, 受到越来越多国内外学者的重视, 并提出一系列改进策略, 以进一步提高基算法的性能. 比较著名的有 Bullnheimer 等人提出的基于排序的蚂蚁系统(RAS)^[3], Stutzle 等人提出的最大最小蚂蚁系统(MMAS)^[4], Dorigo 和 Gambardella 提出的 Ant-Q 算法^[5], 以及 Ant-Q 的改进版本——蚁群系统(ACS). 其中, ACS 算法设计了信息素的全局更新和局部更新两种规则, 是性能最优的 ACO 算法之一^[6]. 在处理规模较大的问题时, 与其他计算

智能算法一样, ACS 算法也存在收敛较慢和容易陷入局部最优的问题^[7]. 通过合理设置 ACS 算法中的一系列控制参数, 可以缓解收敛速度与搜索停滞之间的矛盾. 然而, 目前对参数选择没有严格的理论依据, 根据经验和试验来选取合适的最佳取值通常是与具体问题相关的, 普适性不高^[8]. 因此, 设计出一种自适应参数调整的策略对提高算法收敛速度、求解精度以及增强算法的普适性有着重要的意义. 本文在 ACS 的框架下, 分析了信息素更新系数对算法寻优能力与收敛速度的关系, 定义了平均路径相似度(ATS)来表征寻优过程的成熟程度, 并据此自适应调整信息素更新系数, 提出了改进的自适应 ACS 算法. 经过与典型蚁群算法在多个旅行商问题测试用例上进行对比测试, 证明了新的算法具有较快收敛速度与较强的全局寻优能力.

收稿日期: 2009-06-08; 修回日期: 2009-07-08

基金项目: 国家自然科学基金项目(60533080); 教育部博士点基金项目(20050335096); 浙江省重大科技专项项目(2006c11235)

2 ACS算法描述与相关参数分析

2.1 ACS算法描述

以求解 TSP 问题为背景介绍 ACS 算法,具体步骤如下:

(1) 初始化:设置所有边上信息素的初始值,并将所有蚂蚁放置到随机选择的初始城市上.

(2) 构建路径:每只蚂蚁应用伪随机比例状态转移规则构建出一条路径.当蚂蚁 k 位于城市 r ,它将根据式(1)选择城市 s 作为下一个城市进行转移:

$$s = \begin{cases} \operatorname{argmax}_{u \in J_K(r)} \{ [\tau(r, u)] * [\mu(r, u)]^\beta \}, & \text{if } q \leq q_0 \\ S, & \text{otherwise} \end{cases} \quad (1)$$

式中, $\tau(r, u)$ 表示城市 r 和 u 之间路径上的信息素浓度; $\mu(r, u)$ 表示启发信息,即城市 r 和 u 之间距离的倒数; $J_K(r)$ 表示未被访问的城市集合; β 是平衡信息素和距离启发信息影响力的系数; q 是分布在 $[0, 1]$ 之间的随机数, q_0 为固定参数.若 $q \leq q_0$ 蚂蚁将选择相应边的信息素和启发信息乘积最大的城市进行转移;否则将按照式(2)选择城市转移.

$$R_K(r, s) = \begin{cases} \frac{[\tau(r, u)] * \mu(r, u)^\beta}{\sum_{u \in J_K(r)} [\tau(r, u)] * \mu(r, u)^\beta}, & \text{if } q \leq q_0 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

以上过程将重复执行直到所有蚂蚁都构建完成一条完整路径.

(3) 局部信息素更新:构建路径时,蚂蚁将按照式(3)对刚经过的边 (r, s) 上的信息素进行更新:

$$\tau(r, s) = (1 - \rho) * \tau(r, s) + \rho \tau_0 \quad (3)$$

式中, ρ 为局部信息素挥发比例系数, τ_0 是信息素的初始值.

(4) 全局信息素更新:当所有蚂蚁都构造出一条完整路径后,对最优路径上所有边进行全局信息素更新,如式(4)所示:

$$\tau(r, s) = (1 - \alpha) * \tau(r, s) + \alpha \Delta \tau(r, s) \quad (4)$$

式中, α 表示全局信息素挥发比例系数, $\Delta \tau$ 定义为

$$\Delta \tau(r, s) = \begin{cases} \frac{1}{L_{\text{best}}}, & \text{if } (r, s) \in \text{beset-tour} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

式中, L_{best} 表示最优路径的长度.

(5) 返回(2)重复执行,直到算法满足终止条件.

2.2 相关参数分析

信息素更新的目的是加强较优的路径削弱较差的路径,以引导搜索快速向较优路径周围收敛.然而,在经过一定次数的迭代后,信息素将集中分布在较少的几条较优路径上,蚂蚁选择其他路径的概率大大降低,此时算法极易陷入局部最优.如何恰当地使用信息素更新规则,将成为实现算法快速收敛并跳出局部最优的关键.

对于 ACS 算法来说,全局信息素更新控制系数 α 取值越大,最优路径上的信息素积累越快,可以加快收敛速度,却容易陷入局部最优;而局部信息素更新控制系数 ρ 若取值较大,可提高搜索的随机性,但导向性不强,收敛速度较慢.

图1显示的是 α 和 ρ 分别取较大值和较小值时,求解 TSP 时所得的最优路径长度随迭代次数的变化情况,其中横轴代表迭代次数,纵轴代表每次迭代的最优路径长度.

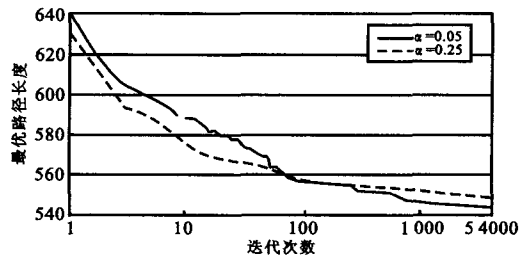
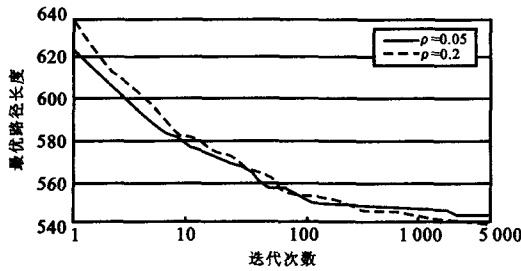


图1 α 取不同值时每次迭代所得的最优路径长度

从图1中可以看出,在前100次迭代中, α 取较大值0.25的收敛速度要快于取最小值0.05.然而,随着迭代次数增加, α 取最小值有更好的全局搜索能力,表现为在算法运行后期有更强的收敛能力,获得更优的解.

从图2中可以看出,在初始阶段, ρ 取较小值,信息素挥发的速率慢,有利于信息素在较短路径上的积累,因此收敛速度快;然而,在算法优化的成熟阶段, ρ 取较大值才能保证较好的全局搜索能力,因为此时需要以较快的速率对较短路径上的信息素进行挥发,保持路径的多样性,才能避免搜索陷入局部解而停滞.

在 ACS 算法优化的初始阶段,为了提高算法的收敛速度,必须在较短路径上释放较多的信息素,此阶段的取 α 值应该较大而的 ρ 取值应该较小.而当算法优化进入“成熟”阶段,即大量信息素集中于几条较短路径上时,较短路径上的信息素应该挥发减少,以避免搜索陷入局部最优解,开拓更多新的路径.因此,此阶段的 α 取值应较小而 ρ 取值应较大.

图2 ρ 取不同值时每次迭代所得的最优路径长度

3 优化的成熟度策略

由上文分析可知,算法需判断当前优化所处状态是否“成熟”,实现对 α 和 ρ 自适应调整.式(6)和式(7)分别表示对 α 和 ρ 的取值进行调整的一般形式:

$$\alpha = F_{\alpha}(\varphi) \quad (6)$$

$$\rho = F_{\rho}(\varphi) \quad (7)$$

式中, φ 是算法优化成熟度的指示器.

3.1 基于路径相似度的优化成熟度的判断

在算法的初始阶段,蚂蚁构建出来的路径的差异较大.随着全局信息素的更新,算法的优化过程逐渐进入成熟状态后,信息素的分布逐渐集中到较短的路径上,蚂蚁将向较短路径上进行转移,此时蚂蚁构建出来的路径相似度会很高.则可以根据所有路径与最短路径的相似程度来判断算法优化的成熟程度,我们定义“平均路径相似度(ATS)”来进行度量.ATS的计算步骤如下:

(1) 每只蚂蚁 k 分别计算自身构建出来的路径 T_k 与当前最优路径 T_{best} 之间的相似度 $s(T_k, T_{best})$ 如式(8)所示:

$$S(T_k, T_{best}) = |T_k \cap T_{best}| \quad (8)$$

其含义是 T_k 与 T_{best} 公共边的数目.

(2) 计算平均路径相似度 ATS,如式(9)所示:

$$ATS = \frac{1}{m * n} \sum_{k=1}^m s(T_k, T_{best}) \quad (9)$$

式中, m 和 n 分别是蚂蚁数量和所有城市的数量.

3.2 自适应调整的实现

用 ATS 量化算法优化成熟程度,对 α 和 ρ 的取值进行调整,具体实现方法如式(10)和式(11):

$$\alpha = \begin{cases} \alpha_{\min}, & \text{if } a_{\alpha} * ATS + b_{\alpha} < \alpha_{\min} \\ a_{\alpha} * ATS + b_{\alpha}, & \text{if } \alpha_{\min} \leq a_{\alpha} * ATS + b_{\alpha} \leq \alpha_{\max} \\ \alpha_{\max}, & \text{if } a_{\alpha} * ATS + b_{\alpha} \geq \alpha_{\max} \end{cases} \quad (10)$$

$$\rho = \begin{cases} \rho_{\min}, & \text{if } a_{\rho} * ATS + b_{\rho} < \rho_{\min} \\ a_{\rho} * ATS + b_{\rho}, & \text{if } \rho_{\min} \leq a_{\rho} * ATS + b_{\rho} \leq \rho_{\max} \\ \rho_{\max}, & \text{if } a_{\rho} * ATS + b_{\rho} \geq \rho_{\max} \end{cases} \quad (11)$$

在优化的初始阶段,ATS值较小,要求 α 值较大 ρ 值较小来加速收敛.当ATS逐渐增大,优化进入了成熟阶段,此时要求 α 值较小而 ρ 值较大以削弱较短路径对蚂蚁的吸引力.因此, a_{α} 取值应该为负,而 a_{ρ} 应该为正.这两个系数的大小还决定了 α 和 ρ 取值作线性变换的斜率.

综上所述,在 ACS 算法框架下提出改进的基于优化成熟度的自适应蚁群算法(AACS),流程图如图3所示.

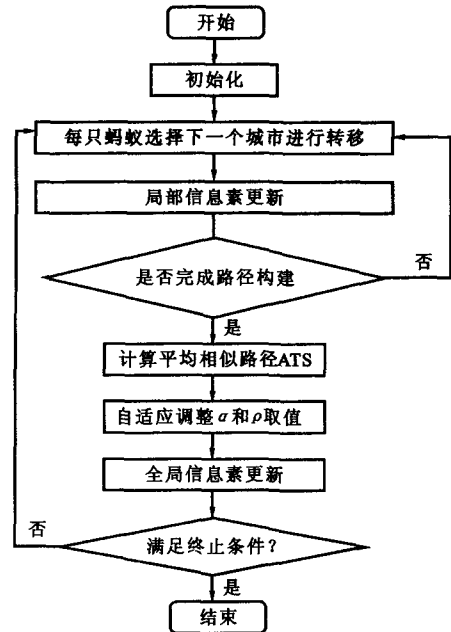


图3 AACS算法流程图

4 仿真实验比较与分析

为了检验 AACS 算法的普适性和有效性,使用传统 ACS 算法与之在 TSPLIB 中的四个基准实例进行对比测试,如表 1 所示.试验设置在每个测试实例上运行 100 次,每次以 5 000 次迭代作为算法的终止条件.

表1 基准 TSP 测试实例的相关数据

TSP 问题实例	TSP 城市数目	已知最短路径长度
oliver30	30	421
eil51	51	426
eil76	76	538
eil101	101	629

4.1 解路径质量比较

由表 2 可以看出,AACS 的解全面优于 ACS.在规模最小的 oliver30 实例中,AACS 在 100 次运行中均找到全局最优解,ACS 也得到了不错的结果.但随着问题规模的增大,AACS 的优越性越来越明显,在 eil51 和 eil76 实例上找到全局最优解的次数远远高于 ACS,而且 AACS 求解的标准方差更小,即稳定性更高.值得注意的是,在 eil101 实例中,ACS 已经难以求得已知最优解,而 AACS 依然保持较好的性能,这得益于在算法成熟阶段对参数和的自适应调整,使得搜索能及时跳出局部最优解,保持开拓新路径的能力.

表 2 ACS 法与 AACS 所求得的路径质量的比较

TSP 实例	算法	平均值	最优值	最优值	方差	最优解次数
oliver30	ACS	421.17	421	438	1.70	999
	AACS	421	421	421	0	100
eil51	ACS	428.21	426	435	2.05	18
	AACS	426.51	426	430	0.89	73
eil76	ACS	541.27	538	551	2.77	15
	AACS	539.10	538	542	1.27	43
eil101	ACS	640.67	630	655	5.86	0
	AACS	635.89	629	647	4.75	8

4.2 收敛特性比较

图 4 展示了 ACS 与 AACS 在迭代过程中最优路径长度的变化情况,图中横轴代表迭代次数,纵轴代表最优路径长度.其中,图(a)、(b)、(c)和(d)分别是 oliver30、eil51、eil76 和 eil101 实例上的测试结果.可以看出,在 oliver30 和 eil51 实例中,AACS 在算法前期的收敛速度明显快于 ACS,这是因为 AACS 算法自适应调整得到较大的 α 值和较小的 ρ 值来强化最优路径,促进算法的快速收敛.这种效果在 eil76 和 eil101 中不是很明显,但 AACS 在优化过程后期相比 ACS 具有更强的收敛能力,这是由于 AACS 在算法成熟阶段依然能保持解的多样性.

4.3 全局搜索能力比较

图 5 展示了 AACS 与 ACS 运行 100 次(每次迭代次数为 5 000),寻找到全局最优次数的累积分布图.其中横轴代表迭代次数,纵轴代表在优化到当前迭代次数时,100 次试验累计能够找到全局最优的次数.其中,图(a)、(b)、(c)和(d)分别是 oliver30、eil51、eil76 和 eil101 实例上的测试结果.可以看出,在四个实例中 AACS 均以更快的速度找到更多的次

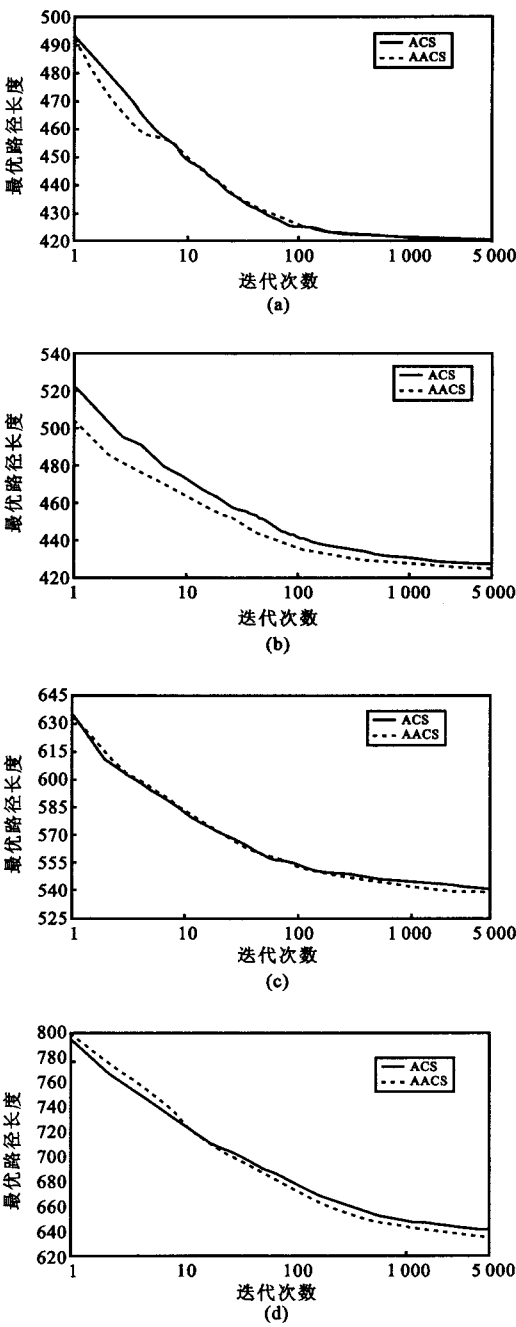


图 4 ACS 与 AACS 算法传统收敛特性比较

数的全局最优解.由于 oliver30 实例的规模较小,因此两种算法的曲线比较接近,但随着实例规模的增大,AACS 的全局搜索能力相比 ACS 越来越突出.在 eil51 和 eil76 实例中,ACS 的曲线在算法后期显得很平缓,这说明 ACS 算法所发现的全局最优解大多数是在算法前期发现的,这是由于 ACS 算法在优化后期容易陷入局部最优解的固有缺点所导致的.AACS 算法则不同,在整个优化过程曲线都保持快速上升的趋势,这说明无论在优化前期还是后期,

AACS都保持着良好的全局搜索能力,不易进入停滞状态,求解的精度更高。

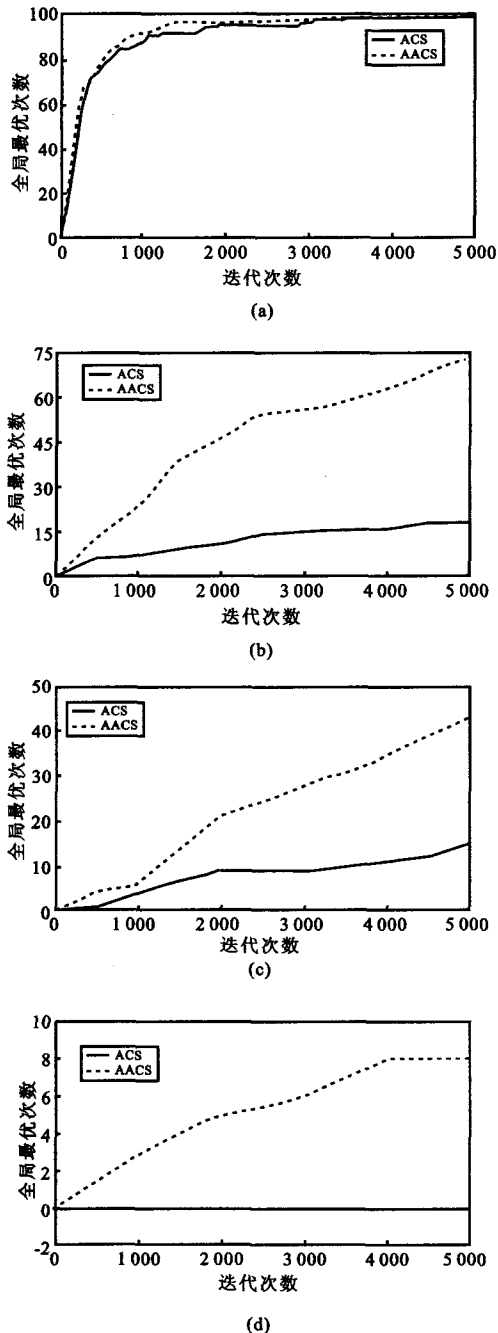


图5 ACS与AACS全局寻优能力比较

5 结束语

本文在对通过对信息素挥发系数和在算法不同优化过程对算法性能的影响的分析,制定了参数的自适应调策略,并利用平均路径相似度来表征优化成熟度,提出一种自适应的ACS(AACS)算法,以解决传统ACS算法中收敛速度与停滞现象之间的矛

盾,并提高算法的普适性.本文提出的AACS算法相比传统的ACS算法,无论在求解质量、精度方面,还是在收敛速度和算法稳定性方面,都具有更优秀的性能.然而,本文所做算法参数选择研究以及提出来的自适应算法都是以最具代表性的组合优化问题TSP作为应用背景,下一步我们希望能将参数分析以及提出的自适应算法扩展到更多组合优化问题上,进一步证明本文结论以及改进算法的有效性。

参考文献:

- [1] Colorini A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies[C]// Proceedings of the First European Conference on Artificial Life. Paris, France, 1991: 134-142.
- [2] Dorigo M, Maniezzo C, Colorni A. Ant system: optimization by a colony of cooperating agents[J]. IEEE Transaction on Systems, Man, and Cybernetics - Part B, 1996, 26(1): 29-41.
- [3] Bullnheimer B, Hart R F, Strauss C. A new rank based version of the ant system - a computational study[J]. Central European Journal of operations research and economic, 1999, 7(1): 25-38.
- [4] Stützle T, Hoos H. MAX-MIN ant system and local search for the traveling salesman problem[C]// Proceedings of the 1997 IEEE International Conference on Evolutionary Computation. USA, Indianapolis, 1997: 309-314.
- [5] Gambardella L M, Dorigo M. Ant-Q: a reinforcement learning approach to the traveling salesman problem[C]// Proceedings of ML-95, Twelfth International Conference on Machine Learning. Boston: Morgan Kaufmann, 1995: 252-260.
- [6] Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem [J]. IEEE Transaction on Evolutionary Computation, 1997, 1(1): 53-66.
- [7] 陈峻,沈洁,秦玲,等. 基于分布均匀度的自适应蚁群算法[J]. 软件学报, 2003, 14(8): 1379-1387.
- [8] 蒋玲艳,张军,钟树鸿. 蚁群算法的参数分析[J]. 计算机工程与应用, 2007, 43(20): 31-36.

作者简介:

王海宁 男, (1984-), 博士研究生. 研究方向为情感计算、智能计算。