第 24 卷第 10 期 计算机应用 Vol. 24 No. 10
2004 年 10 月 Computer Applications Oct. 2004

文章编号:1001 - 9081(2004)10 - 0136 - 03

蚁群算法不确定性分析

曾 洲,宋顺林
(江苏大学 计算机科学与通信工程学院,江苏 镇江 212013)
(bluecourse@ mail. china. com)

摘 要:蚁群算法作为一种开创性的生物仿真算法,因其具有并行性、鲁棒性等优良性质得到了广泛的应用。在对蚁群算法进行系统仿真的实验中,发现蚁群算法存在很多不确定因素。这些因素对蚁群算法的性能造成不同程度的影响,作为一种基于实验的研究性的探讨,本文对所发现的不确定因素做了分析,并根据分析结果对蚁群算法作了相应的改进。

关键词:蚁群算法
中图分类号: TP301.6 **文献标识码**:A

Uncertainty analysis of ant colony optimization algorithm

ZENG Zhou, SONG Shun-lin
(College of Computer Science and Communication, Jiangsu University, Zhenjiang Jiangsu 212013, China)

Abstract: Ant Colony Optimization algorithm(ACO) is a kind of innovative biology emulation algorithm. ACO has been used in many fields because it has some excellent qualities, such as parallelism and robustness. In the process of the system emulation experiment, many uncertain factors of ACO were found. These factors would influence qualities of ACO in various degrees. As a kind of learned disquisition based on experiment, the uncertain factors which were found in experiment were analyzed and ACO was improved based on the analysis result.

Key words: Ant Colony Optimization Algorithm(ACO)

1 蚁群算法原理概述

蚁群算法是对自然界蚂蚁寻径方式的模拟而得出的一种仿生算法。蚂蚁群在寻找食物时会在走过的路径上留下一种称为信息素的物质,释放的信息素浓度跟路径的长度有关,路径越长信息素浓度越低,相反,路径越短信息素浓度越高。其他蚂蚁在运动中会感知到这种物质,并且倾向于选择信息素浓度高的路径上走,同时信息素会随着时间的流逝逐渐减弱或消失,这样就形成了一种正反馈,某一路径上的信息素浓度越高,则后来的蚂蚁选择这条路径的概率就越大,最后,整个蚁群会倾向于走一条最优路径。

2 仿真系统说明

2.1 问题定义

系统没有使用 TSP 问题、中国邮递员问题等 NP 问题作为系统的问题界定,主要原因是实际应用中大多数复杂网络不可能满足不重复遍历节点这个要求,因此仿真系统解决的就是在复杂网络中寻找两节点间的最短路径,允许重复遍历某个节点。

2.2 路径选择机制

文献[1]算法中路径选择考虑了路径信息(启发信息)和信息素两个因素,依据两者的概率来选择。在仿真系统中,实验的目的是找出影响算法的因素,而信息素是蚁群算法的决定性因素,因此仿真系统中路径的选择只依赖于信息素。

3 不确定性分析

3.1 信息素的更新方式影响收敛效率

通常的信息素更新方式有两种:(1)逐步更新,蚂蚁每走一步更新一次;(2)等蚂蚁走完一遍路径之后,再将整条路径的值更新。

下面是测试数据列表,显示收敛时蚂蚁的遍历次数:

逐步更新:(格式 实验序号/遍历次数)

1/98 2/78 3/87 4/93 5/65 6/103 7/75 8/90 9/86 10/64 平均循环次数:84

遍历完后更新:(格式 实验序号/遍历次数)

1/54 2/58 3/54 4/60 5/63 6/56 7/71 8/83 9/46 10/64 平均循环次数:61

实验表明,不同的更新方式对蚁群算法的收敛效率影响很大。遍历之后再更新的特性就是可以提高收敛效率,但是很容易产生早熟现象,同时会影响结果的正确性。鉴于此,仿真系统采用了逐步更新方式。为了提高收敛效率,做了如下改进:设置变量 noSexNum,蚂蚁在前 noSexNum 遍查找路径时不考虑激素信息,路径完全是随机选择的,但是照样逐步更新信息素。蚂蚁在第 noSexNum 遍后再根据信息素查找路径时,网络上已经均匀的分布了一些信息,实验表明,这种方式是有效的,但是 noSexNum 的取值范围是一个值得考虑的问题。

3.2 路径的位置特征和随机数的随机程度影响搜索精度

1) 路径的位置特征

收稿日期:2004 - 04 - 07

作者简介:曾洲(1980 -),男,江苏常州人,硕士研究生,主要研究方向:办公自动化,企业信息化; 宋顺林(1947 -),男,江苏溧阳人,博士生导师,主要研究方向:计算机图形学.

在实验中,发现有些节点间只能找到次优路径,如在查找节点(10,1)之间的路径时,找到次优路径的概率为 90%,最优路径仅为 5%。

例如,找节点 10 至节点 1 的最短路径:

次优路径:10——19——13——1

路径权值:[12][5][2]

节点的边:3 5 7 6

最优路径:10——2——7——19——13——1

路径权值:[6][2][2][5][2]

节点的边:3 8 12 5 7 6

次优路径的总权值为 ValueSecond = 12 + 5 + 2 = 19;最优路径的总权值为 ValueFirst = 6 + 2 + 2 + 5 + 2 = 17;可见最优路径与次优路径的差距不大。在前 noSexNum 遍蚂蚁随机选择的情况下,选择次优路径的概率为(1/3) * (1/5) * (1/7) = 1/105;选择最优路径的概率为(1/3) * (1/8) * (1/12) * (1/5) * (1/7) = 1/10080;可以看出,在初始阶段蚂蚁自由选择的情况下,选择次优路径的可能性远大于选择最优路径。因此,如果采用先随机选择路径使信息素均匀分布,然后再根据信息素查找路径这种方式时,如果路径满足如下位置特征,则很难找到最优路径:

(1)最优路径与次优路径的总权值相差不大;

(2)最优路径的随机选择概率远低于次优路径。

2) 随机数的随机程度

随机数要尽量服从均匀分布,在蚁群随机选择阶段,随机数的随机性能会严重影响结果的正确性。所谓随机性能不佳,最通常的情况是随机数在一定的时间间隔内连续产生相同的数据,在蚁群自由选择阶段,会使某条路径上的权值过于偏大,事实上这条路径未必是最优的,这就导致蚁群最终选择次优路径的可能性增大。

3.3 信息素的算法及其参数选择影响总体性能

3.3.1 经典信息素更新算法概述

信息素的更新算法是蚁群算法的核心内容,同时也是研究和改进的热点。因为蚁群的后期路径选择完全依赖于信息素,所以信息素更新算法的性能对蚁群算法的性能起决定作用。信息素的更新包含两个方面:增强刚走路径的信息素;削弱未走路径的信息素。文献[2]提出的更新算法:

(1)局部更新

蚂蚁每经过一个端点按照公式:

$\tau(j) = (1-\rho) \times \tau(j) + \rho \times \tau 0$

参数 ρ 为气味蒸发因子,$\tau 0$ 为局部更新常数。

(2) 全局更新

在一次迭代中,当所有蚂蚁都完成了自己的路径,应用全局更新规则,仅对所有已走过路径中最短的一条路径上的各端点上的气味进行更新,其他端点的气味只是进行衰减。公式:

$\tau(j) = (1-\rho) * \tau(j) + \rho \times \Delta\tau(j)$

其中对于最短路径 $\Delta\tau(j) = 1/L$,L 为最短路径的长度,其他路径 $\Delta\tau(j) = 0$。

3.3.2 仿真系统信息素更新算法

在仿真系统中,目的是找出影响信息素算法的不确定因素,因此对经典更新算法做了些修改,修改的目的不是为了提高或改进信息素更新算法,只是为了更好的分析信息素更新算法中的不确定因素。因此仅仅考虑了局部更新,全局更新是必要的,作为一种优化手段将在后面叙述。

以下是局部更新算法:

(1) 对所有路径的信息素进行削弱

$HI[i][j] = (1 - DiscountDown \times goLength) \times HI[i][j]$

(2) 对刚走过路径的信息素加强

$HI[i][j] = (1 - DiscountDown \times goLength) \times (HI[i][j] \times SEX)$

其中 $HI[i][j]$ 为信息素矩阵,*DiscountDown* 为信息素削弱消耗率,*DiscountUp* 为信息素增长消耗率,*goLength* 为刚走过的路径值,*SEX* 为信息素预定义量。经典算法中信息素削弱和增长时的挥发度是一样的均为 P,这不利于信息素在更大的程度上调整,因此对于削弱和增长设置了两个挥发度,*DiscountDown* 和 *DiscountUp*。因此整个算法的性能由 *SEX*、*DiscountDown* 和 *DiscountUp* 决定。

3.3.3 SEX 对整个算法的影响

在仿真系统中,将 *SEX* 的值分 10 次由 10 增加至 100,每次测试50组数据,下面是500组数据的实验值曲线图。由图上看出,从结果的正确性和收敛效率来讲,*SEX* 的取值对信息素更新算法基本上没有影响。所以更新算法的性能取决于 *DiscountDown* 和 *DiscountUp* 两个参数。

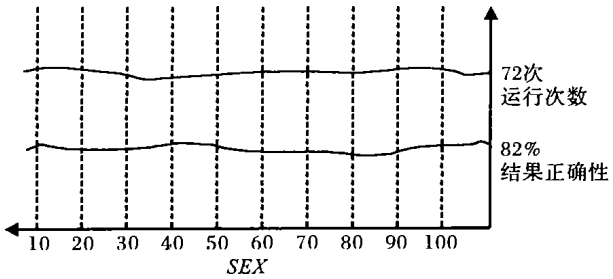

图 1　*SEX* 取值图

3.3.4 DiscountDown 和 DiscountUp 对算法性能的影响

测试 1:固定 *DiscountDown* 和 *DiscountUp* 值,变换不同的测试数据。实验前提:蚂蚁数量 = 40,*noSexNum* = 10,*DiscountDown* = 0.02,*DiscountUp* = 0.06;下面是测试 10 组数据,每组测 50 次。

表 1　固定 *DiscountDown* 和 *DiscountUp* 测试

路径	最优解概率	次优解概率	次优解 - 最优解	平均运行次数
8→7	79.2%	18.0%	1	72
1→19	100%	0	0	76
9→20	30.4%	56.4%	5	65
10→2	4.8%	66.8%	7	57
8→15	100%	0	0	44
3→6	22.4%	4.4%	3	38
16→3	28.8%	58.4%	1	51
17→1	2.8%	2%	1	35
5→1	92.4%	0.4%	1	60
7→1	90%	1.2%	1	63

测试 2:固定测试数据,变换 *DiscountDown* 和 *DiscountUp* 值。实验前提跟测试 1 一样。

测试 1 中,不同的测试数据得到的最优解概率相差很大,这说明对于不同的数据 *DiscountDown* 和 *DiscountUp* 值是要调整的,事实上,在手动调整了 *DiscountDown* 和 *DiscountUp* 值后,测试 1 中的数据均能达到 70% 以上的最优解概率,测试 1 反映了 *DiscountDown* 和 *DiscountUp* 对算法性能的影响。根据前面所定义的局部更新公式,*DiscountDown* 越大,则信息素的削弱程度就越大;*DiscountUp* 越大,则信息素的加强程度就越小。这两个参数对算法性能有很大的影响,因为它们决定了信

息素的削弱速度。信息素削弱的越快,收敛效率增大,也越容易产生早熟现象,找到最优解的可能性减少。但是如果信息素削弱速度过于缓慢,收敛效率会降低,也不利于找出最优解。因此调整 *DiscountDown* 和 *DiscountUp* 值设定一个合适的削弱速度是提高更新算法性能的关键。测试 2 中,将 *DiscountDown* 和 *DiscountUp* 逐渐减小,也就是使信息素削弱速度逐渐减小。第 1 次,增加一次信息素,要 4 次才能削弱为 0,第 10 次,增加一次信息素,要 21 次才能削弱为 0。从测试结果可知,尽管存在一定的波动性(如最后一组测试结果),但是总体上讲,总会存在一个最适合值,也就是此时的 *DiscountDown* 和 *DiscountUp* 组合值使算法性能最佳,在这个值之前和之后的算法性能都会下降。在测试 2 中,最佳值为 *DiscountDown* = 0.048;*DiscountUp* = 0.064。

表 2 变换 *DiscountDown* 和 *DiscountUp* 测试

2→14					
DiscountDown	*DiscountUp*	最优解概率	次优解概率	次优解 - 最优解	平均运行次数
0.06	0.08	78%	14%	15	89
0.054	0.072	74%	18%	15	82
0.048	0.064	82%	4%	12	75
0.042	0.056	76%	12%	15	71
0.036	0.048	54%	4%	12	68
0.03	0.04	64%	6%	12	73
0.024	0.032	74%	20%	15	82
0.018	0.024	66%	14%	15	77
0.012	0.016	60%	22%	15	89
0.006	0.008	70%	18%	15	80

3.3.5 算法改进

1) 信息素更新参数微调

如何在计算过程中跟踪并调整 *DiscountDown* 和 *DiscountUp* 值呢? 可以参照每只蚂蚁在某一时刻的遍历次数来决定。算法如下:

(1) 求 t 时刻所有蚂蚁运行次数的平均值 $EV(t)$:

$$EV(t) = \frac{1}{n}\sum_{i=1}^{h} ant(i,t)$$

(2) 统计运行次数大于 $EV(t)$ $\alpha 1$ 倍的蚂蚁个数 $\beta 1(t)$,其中 $\alpha 1$ 是自定义的。$\beta 1(t)$ 的初始值为 0,遍历所有的 $ant(i,t)$:

$$\beta 1(t) = \begin{cases} \beta 1(t) + 1 & ant(i,t) - EV(t) \times \alpha 1 \geq 0 \\ 0 & ant(i,t) - EV(t) \times \alpha 1 < 0 \end{cases}$$

统计运行次数小于 $EV(t)$ $\alpha 2$ 倍的蚂蚁个数 $\beta 2(t)$,其中 $\alpha 2$ 是自定义的。$\beta 2(t)$ 的初始值为 0,遍历所有的 $ant(i,t)$:

$$\beta 2(t) = \begin{cases} \beta 2(t) + 1 & ant(i,t) - EV(t) \times \alpha 2 \geq 0 \\ 0 & ant(i,t) - EV(t) \times \alpha 2 < 0 \end{cases}$$

(3) 分别计算比率

$\gamma 1 = (\beta 1(t)/n \times 100\%$

$\gamma 2 = (\beta 2(t)/n \times 100\%$

(4) 如果 $\gamma 1$ 大于设定的上限值,则说明部分路径上的信息素偏高,为了均衡信息素,需要缓减信息素削弱速度,将 *DiscountDown* 和 *DiscountUp* 值减小。如果 $\gamma 2$ 小于设定的下限值,则说明信息素在路径上的分布过于均衡,应该加大差距,增加削弱速度,需要将 *DiscountDown* 和 *DiscountUp* 值增大。

需要注意的是,这种改进方式是有时段限制的,特别适用于所有蚂蚁自由选择完路径之后到有近三分之一的蚂蚁找到相同路径这段时间内调整。因为再往后调整将严重影响收敛效率。

2) 全局调整

类似于 M Dorigo 信息素更新算法中的全局调整。设置一个变量 *recordShort*,记录所有蚂蚁到目前为止所能找到的最短路径。自定义时间序列 $t[i]$。*recordShort* 是随时更新的,每当一个蚂蚁找到路径时,就和 *recordShort* 比较,若更小,则更新 *recordShort* 值,同时根据时间序列 $t[i]$,在相应时间点加强 *recordShort* 所记录路径的信息素。实验表明,这种方式能够提高蚁群算法的正确性。

3) 信息素值微调

蚁群算法中一个常见的问题是容易产生早熟现象。即在某些时刻,部分路径上的信息素值与其他路径上的信息素值偏差太大,会影响结果的正确性。因此对偏差太大的值进行调整是必要的,比较著名的方法是设置 MAX 和 MIN 值,超过上下限值的信息素都被强制限制为 MAX 和 MIN 值,算法如下:

(1) 求所有路径(信息素 $\neq 0$) 上的信息素的平均值 EV;

(2) 将所有大于 EV a 倍的信息素相对于 EV 减半(其中 a 是自定义的)。

$$sex[i] = sex[i] - (sex[i] - EV) \div 2(sex[i] \geq EV \times \alpha)$$

这样做的优点是偏差最大的值调整幅度最大。

3.4 蚂蚁的数量影响算法的性能

实验前提: 设定 *SEX* = 25,*DiscountDown* = 0.048, *DiscountUp* = 0.064 *noSexNum* = 10,测试 8 组数据。

表 3 蚂蚁数量对算法性能的影响

2→14				
蚂蚁数量	最优解概率	次优解概率	次优解 - 最优解	平均运行次数
10	40%	2%	12	36
20	54%	12%	15	49
30	70%	18%	15	66
40	70%	4%	12	74
50	84%	12%	15	75
60	88%	2%	12	89
70	94%	2%	12	96
80	96%	4%	15	98

由实验结果可知,算法的性能随着蚁群数量的增加有很大提高,但是运行时间也延长了不少。所以,选择蚁群数量取决于应用场合对算法的要求。

蚁群算法存在很多有待改进的地方。文中只是对通过实验发现的不稳定因素做了简要的分析,因为实验是有偶然性的,所以发现的问题和对问题的分析只是一种探讨性的研究。蚁群算法中如何调整信息素将是一个难题,同时也是以后研究的重点。

参考文献:

[1] DORIGO M, GIANNI DI CARO, STUTZLE T. Ant algorithms[J]. Future Generation Computer System, 2000, 16: 5 - 7.

[2] LUCA DM, GAMBERDELLA M. Ant colony for the traveling salesman problem[R]. TR/IRIDIA/1996 - 5, IRIDIA, Universite Libre de Bruxelles, 1997.

[3] STUTZLE T, HOOS HH. MAX-MIN Ant System[J]. Future Generation Computer System, 2000, 16: 889 - 914.