第30卷 第16期 计 算 机 工 程 2004年8月
Vol.30 №16 Computer Engineering August 2004

·基金项目论文· 文章编号：1000—3428(2004)16—0025—02 文献标识码：A 中图分类号：TP18

蚁群算法求解问题时易产生的误区及对策

徐精明[1,2]，曹先彬[1]，王煦法[1]
(1. 中国科学技术大学计算机科学技术系，合肥 230026；2. 安徽技术师范学院计算机系，蚌埠 233100)

摘 要：蚁群算法是一种新型的模拟进化算法，具有智能搜索、全局优化、稳健性强、分布式计算等优点。是求解复杂的组合优化问题的有力工具。该文对蚁群算法的应用进行了研究，指出了应用该算法时易产生的几个误区，并提出了相应的对策。

关键词：蚁群算法；TSP；误区；对策

Misunderstandings and Counter Measures on Problem Solving by Ant Colony Algorithm

XU Jingming[1,2], CAO Xianbin[1], WANG Xufa[1]
(1.Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230026;
2. Department of Computer, Anhui Technology Normal University, Bengbu 233100)

【Abstract】 Ant colony algorithm is a novel simulated evolutionary algorithm ,which has many advantages ,such as intelligent search ,global optimization ,robusticity ,distributed computation .It provides a very useful tool to solve complex combinatorial optimization problems. In this paper, the applications of ant colony algorithm are discussed, and some misunderstandings frequently coming from the application of the algorithm are designated. Furthermore, the paper gives some corresponding counter measures.

【Key words】 Ant colony algorithm; TSP; Misunderstanding; Counter measure

蚁群算法(Ant Colony Algorithm)是一种基于种群的模拟进化、用于解决复杂优化问题的全新的启发式算法。它是在对自然界中真实蚁群的集体行为(蚂蚁依赖信息素进行通信而显示出的社会性行为)研究的基础上，于20世纪90年代由意大利学者Dorigo M[1,2]等首先提出。

蚁群算法本质上仍是一种随机搜索算法，它是通过对候选解组成的群体的进化来寻求最优解。算法由许多蚂蚁共同完成，每只蚂蚁在候选解的空间中独立地搜索解，并在所寻得的解上留下一定的信息素，蚂蚁倾向于朝着该物质浓度高的方向移动。因此，由大量蚂蚁组成的蚁群的集体行为便表现出一种信息正反馈现象：某一路径上走过的蚂蚁越多，后面的蚂蚁选择该路径的概率就越大。随着算法的推进，较优解上的信息素将逐渐增多，算法渐渐趋于收敛。

近年来，蚁群算法广泛地引起人们的兴趣，逐渐得到了应用。特别是成功地运用于解决组合优化问题，如TSP[3]、QAP[4]、JSP[5]等。虽然蚁群算法本身并不复杂，但在应用该算法求解实际问题时易产生几个误区。本文结合求解TSP，指出这些误区，并提出了相应的对策。

1 蚁群算法模型

蚁群算法通常用于求解复杂的组合优化问题。在对不同性质的问题求解时，蚁群算法模型的定义也有所不同。以平面上m个城市的TSP问题为例说明基本蚁群算法模型。m个城市的TSP问题就是寻找通过m个城市各一次且最后回到出发点的最短路径。

设n是蚁群中蚂蚁的数量，d_{ij} (i,j=1,2,…,m)表示城市i和城市j之间的距离，$\tau_{ij}(t)$ 表示t时刻在城市i与城市j连线上信息素的浓度。初始时刻，各条路径上信息素的浓度相同，设 $\tau_{ij}(0) = C$ (C为常数)。蚂蚁k(k=1,2,…,n)在运动过程中，根据各条路径上的信息素的浓度决定转移方向，$P_{ij}^{k}(t)$ 表示在t时刻蚂蚁k从城市i转移到城市j的概率，其计算公式见式(1)。

$$P_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t)\,\eta_{ij}^{\beta}(t)}{\sum_{s \in tabu_k} \tau_{is}^{\alpha}(t)\,\eta_{is}^{\beta}(t)} & \text{if } (j \notin tabu_k) \\ 0 & \text{else} \end{cases} \tag{1}$$

其中，$tabu_k$ (k=1,2,…,n)为蚂蚁k已走过城市的集合。开始时 $tabu_k$ 中只有一个元素，即蚂蚁k的出发城市，随着进化的进行，$tabu_k$ 中的元素不断增加。随着时间的推移，以前留在各条路径上的信息素逐渐消逝，用参数 $1-\rho$ 表示信息素的挥发程度，所有蚂蚁完成一次循环，各路径上信息素的浓度要根据式(2)作调整。

$$\tau_{ij}(t+1) = \rho * \tau_{ij}(t) + \Delta\tau_{ij} \qquad \rho \in (0,1)$$
$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2}$$

$\Delta\tau_{ij}^{k}$ 表示第k只蚂蚁在本次循环中留在路径ij上的信息素的浓度，$\Delta\tau_{ij}$ 表示本次循环所有蚂蚁在路径ij上所释放的信息素浓度之和。

Dorigo M曾给出3种不同模型，分别称之为ant cycle system, ant quantity system, ant density system，它们的差别在于 $\Delta\tau_{ij}^{k}$ 的计算表达式不同。

在ant cycle system模型中，

$$\Delta\tau_{ij}^{k} = \begin{cases} Q / L_k & \text{if the kth ant uses edge (i,j) in its tour} \\ & \text{between t and t+n} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

基金项目：国家自然科学基金资助项目(60204009)

作者简介：徐精明(1964－)，男，副教授，主研方向为计算智能；曹先彬，博士、副教授；王煦法，教授、博导

收稿日期：2003-07-02 **E-mail**：xwy930@sohu.com

在ant quantity system和ant density system中，$\Delta\tau_{ij}^{k}$ 分别为

$$\Delta\tau_{ij}^{k}=\begin{cases}Q/d_{ij} & \text{if the kth ant uses edge (i,j) in its tour between t and t+1}\\ 0 & \text{otherwise}\end{cases}\quad(4)$$

$$\Delta\tau_{ij}^{k}=\begin{cases}Q & \text{if the kth ant uses edge (i,j) in its tour between t and t+1}\\ 0 & \text{otherwise}\end{cases}\quad(5)$$

上述3种模型中，后两者利用的是局部信息，而前者利用的是整体信息。所以，一般都采用ant quantity system作为基本模型。算法中参数 α，β，Q，ρ，可以用实验方法确定其最优组合，也可以用进化学习得到。停止条件可以用固定进化代数或者当进化趋势不明显时便停止计算。由算法复杂度分析理论可知，该算法复杂度为 $O(nc\cdot n^3)$，其中nc表示循环次数。

2 误区与对策

我们在应用蚁群算法求解实际问题时，发现了几个易产生的误区。下面以求解TSP为例，指出这些误区，并提出了相应的对策。

误区1 利用最大概率确定被选城市。

分析：在t时刻，蚂蚁k从城市i转移到城市j的概率 $P_{ij}^{k}(t)$，$j\notin tabu_k$ 由式(1)计算，并不意味着蚂蚁k到达城市i后，一定从这些 $P_{ij}^{k}(t)$ 中找出最大者，以其对应的城市作下一目标城市。否则该算法就失去了随机性，就有可能丢失某些较好解，以致最终找不到真正最优解。

对策：采用轮盘赌方法，将这些选择概率作累积概率统计，然后产生一个随机数，该随机数落入哪一个累积概率中，该累积概率对应的城市就作为下一个被选城市。

误区2：随机数的随机程度不高。

分析：在采用轮盘赌方法时，需产生一个随机数(如用rand()函数产生)。实际上计算机不可能产生完全随机的数字，所谓的随机数发生器都是通过一定的算法对事先选定的随机种子做复杂的运算，用产生的结果来近似地模拟完全随机数，这种随机数被称作伪随机数。伪随机数是以相同的概率从一组有限的数字中选取的。所选数字并不具有完全的随机性。如果随机种子一样，那么同一个随机数发生器产生的随机数也会一样，这就导致了每次运行时随机序列都一样，随机数并不随机。

对策：要解决这个问题，需要在每次产生随机序列前，先指定不同的种子，这样计算出来的随机序列就不会完全相同了。一般使用同系统时间有关的参数作为随机种子，在调用rand()函数之前，调用srand((unsigned)time(NULL))，以time函数值(即当前时间)作为种子数。但是，如果计算机的速度很快且计算规模不大，还不能完全解决问题，因为可能2次调用srand((unsigned)time(NULL))的时间差小于1s，所以最好用srand((unsigned)(time(NULL)×NC×i×j))，NC为进化代数，i,j为循环变量。这样就可以达到较高的随机性。

误区3：$\sum_{s\notin tabu_k}\tau_{is}^{\alpha}(t)\,\eta_{is}^{\beta}(t)=0$，导致出现 Floating point error:devided by 0 。

分析：在计算 $P_{ij}^{k}(t)$ 时，由式(1)知，需先计算 $\sum_{s\notin tabu_k}\tau_{is}^{\alpha}(t)\,\eta_{is}^{\beta}(t)$。虽然初始时刻，$\tau_{ij}(0)=C$ (C为不等于0的常数)，但在t+1时刻，$\tau_{ij}(t+1)=\rho*\tau_{ij}(t)+\Delta\tau_{ij}$，$\rho<1$，当某路径在多代内都没有蚂蚁经过时，$\tau_{ij}$ 就变得很小，再自乘α 次，以致在计算机的某精度范围内被作0看待，这是主要原因。另外 $\eta_{is}=1/d_{is}$ 一般也较小，再自乘β次，也可能在某精度范围内被作0看待。使得 $\sum_{s\notin tabu_k}\tau_{is}^{\alpha}(t)\eta_{is}^{\beta}(t)=0$，导致出现 Floating point error: devided by 0。

对策：修改式(1)

$$P_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)\,\eta_{ij}^{\beta}(t)}{\sum_{s\notin tabu_k}\tau_{is}^{\alpha}(t)\,\eta_{is}^{\beta}(t)} & \text{if } (j\notin tabu_k)\\ 0 & \text{else}\end{cases}$$

为

$$P_{ij}^{k}=\begin{cases}\dfrac{(\tau_{ij}(t)/M1)^{\alpha}(\eta_{ij}(t)/M2)^{\beta}}{\sum_{s\notin tabu_k}(\tau_{is}(t)/M1)^{\alpha}(\eta_{is}(t)/M2)^{\beta}} & \text{if } (j\notin tabu_k)\\ 0 & \text{else}\end{cases}$$

其中，M1是与τ 同一数量级的常数，M2是与η 同一数量级的常数。这样分子分母同乘一个常数，可保原分式值不变，同时可避免出现 $\sum_{s\notin tabu_k}\tau_{is}^{\alpha}(t)\,\eta_{is}^{\beta}(t)=0$，Floating point error: devided by 0。

误区4：累计概率为0。

分析：在采用轮盘赌方法时，需进行选择概率和累积概率统计。选择概率的计算按公式

$$P_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}(t)\,\eta_{ij}^{\beta}(t)}{\sum_{s\notin tabu_k}\tau_{is}^{\alpha}(t)\,\eta_{is}^{\beta}(t)} & \text{if } (j\notin tabu_k)\\ 0 & \text{else}\end{cases}$$

由于 $\tau_{ij}^{\alpha}(t),\eta_{ij}^{\beta}(t)$ 可能都很小，以致在某精度范围内，分子 $\tau_{ij}^{\alpha}(t)*\eta_{ij}^{\beta}(t)=0$，从而使得选择概率和累积概率为0，进而导致蚂蚁k到达城市i后无法正确选择下一城市，使算法失败。

对策：(同解决误区3)

误区5：Q值的影响不大。

分析：Q值是反映蚂蚁留在所经路径上的轨迹强度常数，在计算 $\Delta\tau_{ij}^{k}$ 时有用，

$$\Delta\tau_{ij}^{k}=\begin{cases}Q/L_k & \text{if the kth ant uses edge (i, j) in its tour between t and t+n}\\ 0 & \text{otherwise}\end{cases}$$

文献[6]中认为“Q值对算法影响不大，$1\leqslant Q\leqslant 10000$”。我们在应用蚁群算法时发现，Q值的大小对算法也有较大影响。Q值过小，会影响算法的收敛速度；Q值过大，可能导致算法不能收敛到较优解。

对策：关于Q等参数值的设定，目前还无理论上的依据。Q值的设定要根据具体问题及结合参数 α，β，ρ 的值而定。一般可以用实验方法确定其最优组合，或用进化学习得到。我们通常用下述误区6中的方法而定。(下转第90页)

消息，表明不再发送其他消息，因为MPI保证消息的接收是严格按照发送次序的，那么当收到所有其他进程的RM消息时，说明网络中没有以自己为目的地的消息了，这时可以看作是孤立的进程，独立设置检查点即可。图4显示了RM消息的效果，所有RM之前的消息都被接受方接收，进程间的通信通道是空的。

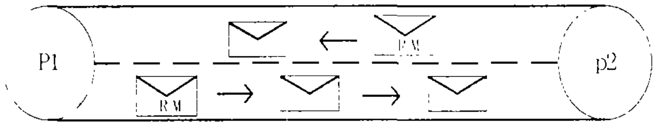

图4 进程间无消息的保证

具体设置检查点的过程如下：

(1)Manager进程。向所有进程发出通知。

(2)MPI进程。收到设置检查点的通知后，发送RM消息给其他所有进程。

(3)MPI进程。接收进来的消息直至所有其它进程的RM消息都被接收，不中止计算，如需和其他进程进行通信则暂停，并对RM进行计数。

(4)MPI进程。断开和并行环境的联系，产生检查点文件。

这样设置检查点时，确保没有任何交叉消息，这样检查点的设置保证了全局一致性，在出现故障进行恢复时，只需回到各进程当前的检查点文件记录的状态即可。

3.3 检查点文件镜象和故障恢复

将检查点文件保存在本地磁盘上只能容忍瞬时故障和间歇故障，当遇到节点宕机时，仍就不能从故障中恢复过来。通过检查点镜像就可以灵活选择检查点文件的存放位置，采用比较通用的平均存储策略，各个进程将保存在本地磁盘的检查点文件信息均衡地影像到其他节点的磁盘上，这样，如果某个节点发生永久性故障，在该节点上运行的MPI进程的检查点信息也可以在其他正常的节点上获得，并从检查点处重新加载应用程序运行。若要容忍m(m>1)各节点的永久性故障，只需将每个MPI进程的检查点信息镜像到其他m个节点的磁盘上即可。

当集群系统发现某个MPI进程出现故障时，通知同一应用程序中的其他进程中断，根据负载机检查点存储位置选择合适的机子重新构架进程，并通知Manager管理进程使各个MPI进程退回到检查点，为保持系统的状态一致性，此时也需要进程间的同步，其同步过程与设置检查点时的过程基本相似，这其中包括将修改过的进程表发送到各个进程。

4 结束语

本文分析了MPI标准在容错方面的处理，介绍了在MPICH环境下利用检查点设置和卷回恢复技术进行容错的设计思想和实现技术，并利用检查点镜像的方法使失败任务可以在非源节点上重启，实现了N-节点容错，是将来实现进程迁移的前提工作。定期设置检查点的时间间隔对于协调设置检查点是一个难点， 不仅要考虑单个节点的平均出错率(MTF)，还要将各个节点的MTF综合考虑，通过实验进行性能评测，选择最佳的时间间隔，此外，针对协调设置检查点协议在自强2000上进行客观的性能评测是下一步的工作。

参考文献

1 Butler R M,Lusk E L.Monitors, Messages and Clusters:The P4 Parallel Programming System. Parallel Computing,1994,20(4):547-564

2 The MPI Forum. The MPI Message-passing Interface Standard.http://www.mcs.anl.gov/mpi/standard .html,1995-05

3 Stellner G. CoCheck:Checkpointing and Process Migration for MPI. In 10th Intl. Par. Proc. Symp.,1996-04

4 Gropp W, Lusk E. Fault Tolerance in MPI Programs. http://www-unix.mcs.anl.gov/~gropp/bib/papers/2002/mpi-fault.pdf

5 Chandy K M,Lamport L.Distributed Snapshots:Determining Global States in Distributed Systems.ACM Trans.Computer Systems,1998,(3):63-75

(上接第26页)

误区6：参数组合。

分析：算法中的主要参数有：Q为蚂蚁留在所经路径上的轨迹强度常数，α 为残留信息的相对重要程度，β 为期望值的相对重要程度，ρ 为残留信息的保留率，这些参数值的设定对本算法性能也有很大影响。Q值过小，追加信息不明显，会影响算法的收敛速度；Q值过大，可能导致算法不能收敛到较优解。α 值的大小表明留在所经路径上的信息量受重视程度，α 值越大，蚂蚁选择以前走过的路径的可能性大，但过大又会使搜索过早陷入局部最小值。β 值越大，蚂蚁重视局部信息的程度大，选择以前走过的路径的可能性小；β 值过小，又会使搜索过早陷入局部最优解。ρ 值太大，挥发很小，以前积累的信息比例太大，不易更新；ρ 值太大，挥发很大，积累的信息比例太小，不能很好地在蚂蚁间传递信息。所以能否选择确定一组合适的参数，也是本算法能否成功的关键之一。

对策：目前还无理论依据来设定这些参数，我们的做法是从实验中观察、比较，最后确定一种较优的参数组合。

因为α 、β 是残留信息与期望值相对重要程度，所以不妨固定β ，让α 变化。这样，只需将Q、α 、ρ 设为循环变量，在每种情况下，观察平均解的下降率、最优解的变化趋势、最优解值及所需进化代数，来选定一组较优参数值。

3 结束语

本文在分析现有蚁群算法的基础上，指出了应用该算法求解实际问题时易产生几个误区，分析了产生这些误区的原因，并提出了相应的对策。

蚁群算法的理论基础目前尚未奠定，许多工作还有待深入展开。如何从理论上对算法进行有效分析、如何选定最优参数组合，将是我们进一步研究的内容。

参考文献

1 Dorigo M, Maniezzo V, Colorni A. Ant System: Optimization by A Colony of Cooperating Agents [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B, 1996, 26(1): 29-41

2 Dorigo M,Gambardella L M. Ant Colony System: a Cooperative Learning Approach to the Traveling Salesman Problem [J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53-66

3 杜端甫. 运筹图论. 北京：北京航空航天大学出版社，1990:240

4 Gambardella L M,Taillard E D, Dorigo M.Ant Colonies for the Quadratic Assignment Problem[J]. Journal of the Operational Research Society, 1999, 50:167-176

5 Colorni A,Dorigo M, Maniezzo V et al. Ant System for Job-shop Scheduling[J]. Belgian J. of Operations Research Statistics and Computer Science,1994,34(1): 39-53

6 魏 平.用于一般函数优化的蚁群算法. 宁波大学学报，2001,14(4): 52-55