计算机科学2002Vol. 29№. 1

随机算法的一般性原理*)

The General Principles of Randomized Algorithms

贺 红 马绍汉

(山东大学计算机科学系 济南250100)

Abstract The last decade has witnessed a tremendous growth in the area of randomized algorithms. During this period, randomized algorithms went from being a tool in computational number theory to finding widespread application in many types of algorithms. Two benefits of randomization have spearheaded this growth: simplicity and speed. For many applications, a randomized algorithm is the simplest algorithm available, or the fastest, or both. A handful of general principles lie at the heart of almost all randomized algorithms, despite the multitude of areas in which they find application. We briefly survey these here in order to draw about the method of studying randomized algorithms.

Keywords Randomized algorithms, General principals, Method of studying

最近十几年来,国际上对随机算法(randomized algorithm)的研究有了巨大进展。在此期间,随机算法从一个计数理论的工具发展到今天在许多类型的算法中都得到了广泛应用,显示了随机算法本身强大的生命力。

所谓随机算法,就是在执行过程中要做出随机选择的算法。随机算法有两种不同类型:其一是总能给出正确的解,两次运行之间唯一的区别是运行时间不同,我们把这种随机算法叫做 Las Vegas 算法;其二是有时会产生不正确解的算法,然而我们能够界定产生不正确解的概率,我们把这种随机算法叫做 Monte Carlo 算法。这两种算法哪种更好些,要看它应用于哪类问题,Las Vegas 算法可以看成是 Monte Carlo 算法当错误的概率为零的情况。

随机算法有两个优势:简单和快速。我们研究随机算法,除了它的研究带来新的方法和新的思想外,另一个重要原因是它与算法复杂性的关系。讨论算法的时间复杂性估计,对解答 P 类问题算法分析,已经取得若干令人信服的结果。但对于 NP 难解问题,仍然没有取得实质性进展。在这种背景下,研究解答 NP 难解问题的近似算法引起人们极大的兴趣。但是,对近似算法近似性能比的估计要求对所解答问题的所有实例都成立,就显得太苛刻。致使在 P≠NP 假设下,若干 NP 难解问题找不出其近似性能比有界的近似算法,例如货郎优化问题。即使有些 NP 难解问题存在近似性能比有界的算法,但这种上界也似乎太大了些,例如图着色问题。针对这种情况,Karp 试图预先假定问题实例在实例空间中服从某种概率分布,设计出解答若干 NP 难解问题的概率算法。概率算法有两种思路,其一是算法为确定型算法,解答问题的实例服从某种概率分布。可分析算法的期望时间复杂性或在概率为1(几乎处处)的条件下,给出解答问题的精确解或近似解。设计概率算法的另一种思路是,求解问题的实例空间是确定的,而将随机语句引入算法,这就是通常讲的随机算法。Karp 的思想引起了研究算法的人们对随机性的兴趣。

众所周知,许多 NP 问题它们的真正难度或许在于我们不能找出其构造中一致性的东西,换句话说,它们的构造太乱以至于我们不能清晰准确地对它们进行刻画,按照经典的基于清晰的离散结构的算法理论自然找不到解决问题的好算法。此时,随机算法退而求其次,不考虑最坏情况下的时间耗费,而考虑其平均时间耗费,寻找一个在大多数情况下是个好算法的解决途径,尽管它可能不是绝对好的,但是比对问题无法估计要好得多了。

在许多应用中,随机算法是能找到最简单的或者最快的算法,或者二者得兼。虽然随机算法已经应用于众多领域,但是,所有随机算法的核心遵循几个一般原则。我们假定读者了解算法和复杂性以及概率论方面的基本概念而不加重复。本文试图陈述随机算法遵循的一般原则,并且举例说明随机算法研究问题的思路及研究现状。

1 用对手作衬托

对于一个确定性算法,当对它进行绝对行为的复杂性分析无意义时,用经典的对手(Adversary)论证的方式可以建立该算法的运行时间的下界。例如,我们通常使用竞争性分析来分析联机算法(Online algorithm),一个对手能产生出一个任务序列,在该序列上一个给定的联机算法的费用增加而优化的脱机算法(Offline algorithm)的费用几乎不增加,以此确定竞争系数。常用的对手有忘却的对手(Oblivious adversary)和自适应对手(Adaptive adversary)。例如,Manasse, McGeoch 和 Sleator[1]证明了对于有 k+1个点的测量任务系统的联机算法 BALANCE 的竞争系数≤k, Irani 和 Rubinfeld[2]证明了对于平衡算法的两个服务器情况的特例竞争系数为10,等等。这种证明所用的方式是,借助不同的对手构造不同的输入,考察联机算法在这些不同输入上的性能表现,从而确定联机算法的竞争系数。实际上,一个随机算法可以看作在一个确定性算法的集合上的概率分布,而对手可以构造一个输入,用该输入来衬托从而改进在这个集合上的某个(或一小部分)确定性算法。设计一个输入能证明一个随机选择的算法不好是很困难的,正如举出反例证明命题不正确一样困难。但是这个方法重在强调"任意"随机算法的成功性。最直接的例子有游戏树评价[3],联机算法[4]等。

*)本文受国家自然科学基金资助(69873027)。贺 红 博士生,主要研究方向为网络优化,智能路由等。马绍汉 教授,博导,主要研究方向为图算法,NP 难解问题,网络优化等。

2 随机抽样

关于来自全域的随机样本是全域作为一个整体的代表的观点贯穿随机算法的始终。在几乎所有随机算法的文献中该观点无处不在。在选择算法[5]、数据结构[6]、图算法[7]以及近似计数[8]中，都有一个合适的随机抽样来代表被研究的总体。

3 提供丰富的证明方法

通常，我们要求一个算法能确定是否一个输入具有某种特征(例如"x 是素数吗?")，算法总是通过找到一个证明 x 确实具有该特征来确定。对于许多问题，确定性证明的困难在于要搜索一个巨大得无法穷尽的搜索空间。然而，通过建立一个包含大量证明的空间，可以从这个空间中随机地选择一个元素来满足需要。随机选择的项目本身就是一个证明，更进一步说，过程的独立重复减少了证明不存在于任何重复过程中的概率。最有说服力的例子是 M. O. Rabin 在这方面做出的开创性的工作。

Rabin 应用随机算法，令人信服地解答了合数判定问题：给定正整数 n，询问是否存在两个大于1的正整数 l，m，使得 n=l*m。这个问题的补问题为素数判定问题。根据文[9]的定理，Rabin 设计出判定 n 是否为合数的随机算法。

合数判定随机算法：

step1 For I 1 to m do
begin
step2 b Random[1,2,…,n−1]//从{1,2,…,n−1}中随机取一个数//
step3 If W(b) then return "YES"
end
step4 return "NO"

其中，n 为自然数，1≤b<n，W(b)表示下述条件之一：(a) $b^{n-1}\neq 1(\mathrm{mod}n)$；或者(b)存在 i[(n−i)/$2^i$=m，m 为整数且1<gcd($b^m$−1,n)<n]。

对于任给的 ε>0，合数判定随机算法以置信度(1−ε)解答合数判定问题。注意到当 n 为素数时，算法能以置信度1作出正确回答，只有当 n 为合数时，算法将 n 错误地判定为素数的概率仅为$1/2^m$。据报导[9]，Rabin 对这个合数判定随机算法作了实验。令 m=10，因而取 ε<10^{-3}，然后对小于等于500的素数 P 实验了2^P-1的素数性，与现有素数表对照，没有一个错误，而整个实验仅花了几分钟。Rabin 还采用这个随机算法去测试尚未确定是否为素数的任意数。他采用从2^{400}逐次减1的方法，在一分钟内，判定$2^{400}-593$为素数，重复次数 m=100，因而错判率小于10^{-30}。若用通常的循环除法判定这个数的素数性，大约需要进行10^{60}次除法，即使使用亿次计算机，也要进行3*10^{44}亿年才能完成这个实验。

虽然合数判定随机算法要比现有的确定型算法快得多，但很遗憾，至今无法证明它比所有确定型合数判定算法快。Rabin 等人的这些研究工作使人们对随机算法寄予了很大希望。

4 指纹和 Hash 技术

一个长串可以由一个使用随机映射得出的短"指纹"(Fingerprinting)代表。在模式匹配的某些应用中可以看到，如果两个长串的"指纹"相同那么这两个长串就几乎是相同的。对比短的指纹比对比原来两个长串本身要快得多[10]。这也是 Hash 技术隐含的观点，当一个大域中取出的元素组成一个小集合 S 被映射成一个较小的域，并且保证 S 中不同的元素有不同的象，这给出一个决定 S 中成员的有效方法，并且在产生伪随机数和复杂性理论中有进一步的应用。

这里不妨以验证多项式恒等为例。

欲证 p1(x)*p2(x)=p3(x)，在一个大域 S 上，随机取 t-S，计算 p1(t)，p2(t)，p3(t)，若有 p1(t)*p2(t)=p3(t)，则可判定 p1*p2=p3。这似乎有些缺乏说服力，但是反过来考虑，如果 p1*p2=p3不成立，那么，对随机取出的 t，使 p1(t)*p2(t)=p3(t)的可能性就太小了。

此例虽简单直观，却展示了随机性的强大作用。

5 随机重排序

在大量数据结构和计算几何问题中，随机算法的一个重要应用是对输入数据随机重排序[6]。在应用方面也有一些相对朴素的算法。

6 负载平衡

对于在多种资源之间进行选择的问题，例如多个处理器网络中的通讯链路，随机性可以用来在资源之间分散负载。这种思想在并行或分布环境中特别有效。在该环境中，资源利用由当地大量站点的状况决定而不考虑这些决定对整个网络的影响。

随机性在解决排列路由问题中的卓越表现最早是由 Valiant[11]提出的，他给出了一个忘却的随机算法(Oblivious randomized algorithm)。一个消息的路由依赖于它的源点、目的地和随机中介点的选择，而不是依赖于源点、目的地和对其它同时传送的消息的选择。在并行通讯模型中，我们总是假设一个节点在每一步能沿着它所有的链路传递消息。当节点度很大时这个假设是不现实的。Aleliunas[12]和 Upfal[13]给出了边界限定度网络来解决 Valiant 的路由问题。它以很大的概率对 N 个节点，每个节点出度为 d 的网络在 O(logN)步内完成对任何排列的路由。

7 快速混合 Markov 链

在运筹学与计算机科学中，某些算法以如下的方式进行：目标是要决定 N 个有序元素中的最优者，算法以其中一个元素开始，而后逐次移动到更好的元素，直到到达最优者为止。(最重要的例子可能是线性规划的单纯形算法，该算法试图要求一个受线性约束条件的线性函数的最大值，此时一个元素对应于可行区域的一个端点。)如果从"最差的情况"的角度考察算法的效率，那么一般都能构造出大致需 N-1步才到达最优元素的例子。在随机算法中，我们经常用到 Markov 链，因为它能从任何状态等可能地进入到任一更好的状态。

对于一些具有某种给定特性的对象的组合计数问题，我们有基于在给定抽样框上随机抽样的近似算法。这样的抽样经常是困难的，因为它可能需要计算样本空间的容量，当我们想通过抽样解决问题时这是个难点。在某些情况下，抽样可以通过定义在全域元素上的 Markov 链实现，使用 Markov 链得到的一个短的随机步长也能均匀地代表全域[14]。

8 隔离和对称切断

在并行计算中，当解决一个有许多可行解法的问题时，确保不同的处理器都在协同努力寻找同一个问题的解是十分重

要的。这就需要在不考虑任何解空间的单个元素的前提下把一个特殊的解从所有可行解中隔离出来。一个好的随机策略应达到"隔离",通过绝对地在可行解上选择一个随机次序,然后要求所有处理器集中力量寻找阶最低的解。在分布式计算中,某些处理器经常需要打破"死锁"状态,从而达到一致。随机算法还是一个避免死锁的强大工具。

我们以选择协作(Choice Coordination)问题为例。在并行和分布计算中经常出现的问题之一是要打破一个可行解集合的对称性,这可以通过随机算法来实现。我们使用下述处理器之间通讯的简单模型:有 m 个登记表可被这 n 个处理器读写,几个处理器可能几乎同时地试图读写或修改一个登记表。为解决这种冲突,我们假设处理器使用一个加锁机制,当某几个处理器试图存取登记表而有一个处理器唯一地获得了存取权限时它加锁,其它的处理器要等到锁被释放后才能再次为存取登记表而竞争。所有这些处理器要运行一个策略,以便在 m 个表中作出一个选择,策略运行到最后 m 个登记表的每一个含有一个特殊符号。一个选择协作问题的复杂性可用 n 个处理器的总的读写步数来衡量。为了显示随机算法的重要贡献,我们给出一个随机策略,对任意 c>0,它将以成功的概率至少$1-2^{-n^{(c)}}$在 c 步内解决冲突[15]。

9 概率方法和存在性证明

通过论证一个随机选择的对象在正概率下有某些特性,可以建立对象具有该特性的存在性证明。这样的论证虽然直观易懂,却并不能给出如何寻找这样一个对象。有时,我们用这种方法来说明解决某个难题的算法确实存在,尽管知道算法存在,但不知算法是什么样的,也不知道如何去构造算法,这就引出算法中非一致性(non-uniformity)的观点。

综上所述,随机算法是从概率的观点而不是分析的观点去考察处理问题的过程,这种新工具的引入使得算法设计与分析重新呈现出勃勃生机。

早期的算法研究多半浅显易懂,富于技巧,以至于有些算法设计可作为中学的数学竞赛题。随机算法带来了一些意义深远的转变,这些转变的标志是一些抽象方法的引进和一些完全不平庸结果的出现。我们不必为这一转变担忧而应该为其喝彩,因为这不是经典算法分析丧失其地盘而是它因新工具的注入而正变得更加强壮。随机算法带来的好的结果揭示了好的算法分析与设计不仅存在而且比人们想象得要复杂得多,算法分析本身就是在设计算法。换个角度来看,把随机性引入算法,实质上是把经典数学应用于离散数学。这种各分支的相互交叉,可能代表着数学发展的一个最重要趋势,计算机科学界可以希望通过获取诸多数学分支中的有力工具而使算法领域变得更加壮大。

参 考 文 献

1 Hochbaum D S. Approximation algorithms for NP-hard problems. PWS, 1997

2 Irani S, Karlin A R. Online computation, in[1], 521～564

3 Snir M. Lower bounds on probabilistic linear decision trees. Theoretical Computer Science, 1985, 38: 69～82

4 Sleator D D, Tarjan R E. Amortized efficiency of list update and paging rules. Communications of the ACM, 1985, 28: 202～208

5 Feller W. An Introduction to Probability Theory and Its Applications, Volume Ⅰ, Ⅱ, John Wiley, New York, 1968

6 Aragon C R, Seidel R G. Randomized search trees. In: Proc. of the 30th Annual IEEE Symposium on Foundations of Computer Science, 1989. 540～545

7 Seidel R G. On the all-pairs-shortest-path problem. In: Proc. of the 24th Annual ACM Symposium on Theory of Computing, 1992. 745～749

8 Harary F, Palmer E M. Graphical Enumeration, Academic Press, New York, 1973

9 Rabin M O. Probabilistic Algorithms, in Algorithms and Complexity. edited by J. Traub, Academic press, 1976

10 Blum M, Kannan S. Designing programs that check their work. In: Proc. of the 21st Annual ACM Symposium on Theory of Computing, 1989. 86～97

11 Valiant L G. The complexity of enumeration and reliability problems. SIAM Journal on Computing, 1979, 8: 410～421

12 Aleliunas R. Randomized parallel communication. In ACM-SIGOPS Symposium on Principles of Distributed Systems, 1982. 60～72

13 Upfal E. Efficient schemes for parallel communication. Journal of the ACM, 1984, 31: 507～517

14 Karp R M, Luby M. Monte Carlo algorithms for the planar multiterminal network reliability problem. Journal of Complexity, 1985, 1: 45～64

15 Cook S A. A taxonomy of problems with fast parallel algorithms. Information and CONTROL, 1985, 64(1-3): 2～22

(上接第98页)

7 Liu J W S, et al. Imprecise Computations. Proceedings of the IEEE, 1994, 82(1): 83～93

8 Dey J K, Kurose J, Towsley D. On-line Scheduling Policies for a Class of IRIS Real-Time Tasks. IEEE Transactions on Computers, 1996, 45(7): 802～813

9 Hamdaoui M, Ramanathan P. A Dynamic Priority Assignment Technique for Streams with (m, k)-Firm Deadlines. IEEE Transactions on Computers, 1995, 44(12): 1443～1451

10 Aydin H, et al. Optimal Reward-Based Scheduling of Periodic Real-Time Tasks. In: Proc. of 20th IEEE Real-Time Systems Symposium, Dec. 1999

11 Aydin H, Melhem R, Mossé D. Incorporating Error Recovery into the Imprecise Computation Model. In: The Sixth Intl. Conf. on Real-Time Computing Systems and Applications, Dec. 1999

12 Aydin H, Melhem R, Mossé D. Tolerating Faults while Maximizing Reward. In: Proc. of the Twelfth Euromicro Conf. on Real-Time Systems, June 2000