近似算法与随机算法的分析方法与应用实例

林舒

北京大学 信息科学技术学院
linshu@net.pku.edu.cn

2016 年 1 月 29 日
绵阳南山中学

大纲

大纲

真的要永远追求完美算法吗？

我们往往在追求各种“完美算法”

- 100% 正确
- 时间、空间复杂度低
- 代码短、优美

完美算法不完美！

然而现实是：

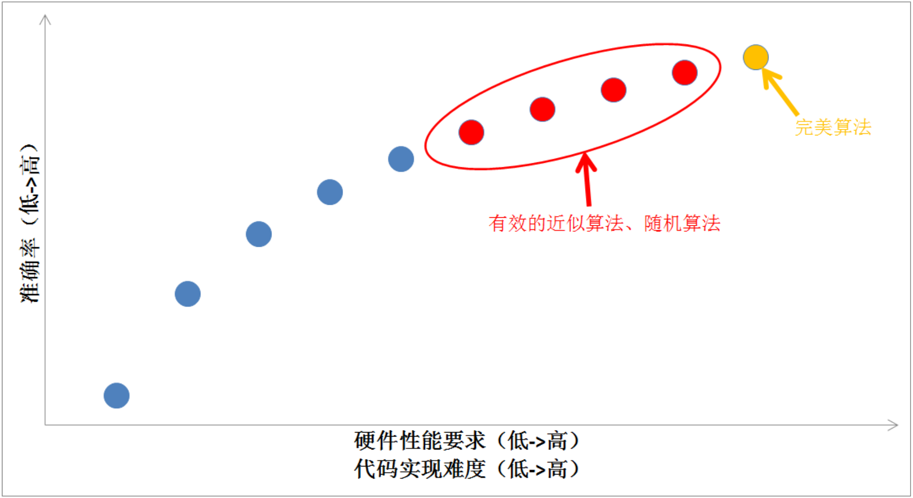
程序竞赛中

- 这题太难了实在想不出来！
- 终于想出来了可是来不及码了！
- 这题就没有最优解法！
- 写 AI 就是靠运气！
- ……

实际应用中

- 客户要求响应速度快！（搜索引擎、购物网站）
- 客户给的硬件太烂！（嵌入式开发）
- 研究问题参数太多太复杂！
- ……

利弊权衡



近似算法与随机算法的共同点

近似算法和随机算法的共同点是：

- 适当牺牲解的准确性，或者算法的稳定性
- 提高效率，或降低代码实现难度

主要内容

- 说明近似算法和随机算法的理论分析要点
- 用一些简单但经典的例子阐明分析方法
- 介绍一些目前学术界、工业界较热门的应用领域

上述内容可能对竞赛帮助不大，但在未来的科研或者工作中，或多或少都会有所接触。

大纲

近似算法和最优化算法

定义 (近似算法)

对组合优化问题 Π，A 是一个多项式时间算法。若对 Π 的每个实例（输入）I，算法 A 输出一个可行解 σ，则称 A 是 Π 的近似算法。

记 $\mathrm{A}(I) = c(\sigma)$，表示算法 A 对于实例 I 的输出的价值为 $c(\sigma)$。

定义 (最优化算法)

若对组合优化问题 Π 的每个实例 I，恒有 $\mathrm{A}(I) = \mathrm{OPT}(I)$，即 A 总能得到最优解，则 A 是 Π 的最优化算法。

近似比

定义 (近似比)

组合优化问题 Π 是最小化问题时，记

$$r_A = \frac{\mathrm{A}(I)}{\mathrm{OPT}(I)}$$

组合优化问题 Π 是最大化问题时，记

$$r_A = \frac{\mathrm{OPT}(I)}{\mathrm{A}(I)}$$

若对 Π 的每个实例 I，$r_A(I) \le r$，则称 *A* 的近似比为 r，又称 *A* 为 r-近似算法。r 为常数时，称 *A* 具有常数比。

可近似性

定义 (可近似性)

NP 难的组合优化问题，可近似性分为三类：

- **完全可近似** 对任意 $\epsilon > 0$，存在 $(1+\epsilon)$-近似算法。
- **可近似** 存在具有常数比的近似算法。
- **不可近似** 不存在具有常数比的近似算法。

最小顶点覆盖集

Π：给定无向图 $G = \langle V, E\rangle$，求最小的顶点覆盖集。

$V' \subseteq V$ 是顶点覆盖集当且仅当：对任意 $e \in E$，e 至少有一个顶点 $\in V'$。

最小顶点覆盖集：MVC 算法

算法 MVC

$V' \leftarrow \emptyset$
while $E \neq \emptyset$
  do 任取 $e = (u, v) \in E$
    $V' \leftarrow V' \cup \{u, v\}$
    $E \leftarrow E - \{e | e$ 与 u 或 v 关联 $\}$
return V'

最小顶点覆盖集：MVC 算法分析

- 假设选取了 k 次边，则 $\mathrm{MVC}(I) = |V'| = 2k$
  而由于这 k 条边互不关联，有 $\mathrm{OPT}(I) \geq k$
  于是，

$$\frac{\mathrm{MVC}(I)}{\mathrm{OPT}(I)} \leq \frac{2k}{k} = 2$$

  故 MVC 是 2-近似算法
- 若实例 I 中，图 G 由 k 条互不关联的边组成，则
  $\mathrm{MVC(I)} = 2k$，$\mathrm{OPT}(I) = k$，$\frac{\mathrm{MVC}(I)}{\mathrm{OPT}(I)} = 2$
  因此近似比至少为 2

多机调度问题

Π：已知有穷作业集 A 中每个作业 a 的处理时间 $t(a)$，要求将作业分配给 m 台相同的机器，使得最晚完成时间最短。

即，将 A 划分为 m 个不相交的子集 $\{A_1, A_2, \cdots, A_m\}$，使得

$$\max \left\{ \sum_{a \in A_i} t(a) \,\middle|\, i = 1, 2, \cdots, m \right\}$$

最小。

多机调度问题：GMPS 算法分析

算法 GMPS：将每个作业分配给当前时间和最少的机器

- 假设最终机器 M_j 时间和最多，a_k 是 M_j 的最后一个作业
  则 M_j 去掉 a_k 后一定是时间和最少的
  记所有作业总时间和为 $T = \sum\limits_{a \in A} t(a)$

$$
\begin{aligned}
\mathrm{GMPS}(I) &= \frac{1}{m}[T - t(a_k)] + t(a_k) \\
&= \frac{1}{m}T + (1 - \frac{1}{m})t(a_k) \\
&\leq \mathrm{OPT}(I) + (1 - \frac{1}{m})\mathrm{OPT}(I) \\
&= (2 - \frac{1}{m})\mathrm{OPT}(I)
\end{aligned}
$$

故 GMPS 是 2-近似算法

算法 DGMPS：从大到小将每个作业分配给当前时间和最少的机器

- 假设最终机器 M_j 时间和最多，a_k 是 M_j 的最后一个作业
  则 M_j 去掉 a_k 后一定是时间和最少的
  记所有作业总时间和为 $T = \sum_{a \in A} t(a)$，若 M_j 不止一个作业

$$
\begin{aligned}
\mathrm{DGMPS}(I) &= \frac{1}{m}[T - t(a_k)] + t(a_k) \\
&= \frac{1}{m}T + (1 - \frac{1}{m})t(a_k) \\
&\leq \mathrm{OPT}(I) + \frac{1}{2}(1 - \frac{1}{m})\mathrm{OPT}(I) \\
&= (\frac{3}{2} - \frac{1}{2m})\mathrm{OPT}(I)
\end{aligned}
$$

故 DGMPS 是 $\frac{3}{2}$-近似算法

满足三角不等式的货郎问题

Π：给定满足三角不等式的带权无向完全图 G，求最短的哈密顿回路。

最小生成树法 MST

1. 求图 G 的最小生成树 T
2. 在 T 上求经过每条边恰好两次的欧拉回路。
3. 在上述欧拉回路中删去中间重复经过的点，形成原图的哈密顿回路。

- 有：

$$\begin{aligned}\mathrm{MST}(I) &= \\ &\leq \\ &= 2\cdot T \\ &\leq 2\mathrm{OPT}(I)\end{aligned}$$

故 MST 是 2-近似算法

最小权匹配法 MM

1. 求图 G 的最小生成树 T
2. 对于 T 的奇度点，求它们在原图中的最小权匹配 M
3. 把 M 加入 T，并在新图上求欧拉回路
4. 在上述欧拉回路中删去中间重复经过的点，形成原图的哈密顿回路。

- 有：

$$\begin{aligned}\mathrm{MM}(I) &= \text{哈密顿回路长度}\\ &\leq \text{欧拉回路长度}\\ &= T\text{的边权和} + M\text{的边权和}\\ &\leq \mathrm{OPT}(I) + \frac{1}{2}\mathrm{OPT}(I)\\ &= \frac{3}{2}OPT(I)\end{aligned}$$

故 MM 是 $\frac{3}{2}$-近似算法

0-1 背包问题

Π：给定

- n 个物品，物品 i 重量为 w_i（$\leq B$），价值为 v_i
- 背包重量限制为 B

求选择哪些物品才能在不超过重量限制的条件下价值最大。

即，求 $S \subseteq \{1, 2, \cdots, n\}$，满足

$$\sum_{i \in S} w_i \leq B$$

且使得 $\sum_{i \in S} v_i$ 最大。

贪心算法 GKK

1. 将物品按单位重量价值从大到小依次尽量放入背包，得到一个价值为 V 的可行解
2. 设物品 j 是所有物品中价值最大的，若 $v_j > V$，则最终背包只放物品 j；否则采用上一步得到的可行解

- 假设物品 k 是第一个未放入的物品，有

$$\mathrm{OPT}(I) < \mathrm{GKK}(I) + v_k \leq \mathrm{GKK}(I) + v_j \leq 2\mathrm{GKK}(I)$$

故 GKK 是 2-近似算法

算法 PTAS$_\epsilon$

1. 事先设定 $\epsilon > 0$，令 $m = \lceil 1/\epsilon \rceil$
2. 枚举不超过 m 个物品的物品集合，若总重量不超过 B，装入背包，并用 GKK 算法将剩余物品装入背包
3. 比较所有装法，选择价值最大的作为近似解

0-1 背包问题：PTAS$_\epsilon$ 算法分析

算法 PTAS$_\epsilon$

- 设最优解为 S^*，考虑 $|S^*| > m$ 的情况
- 当枚举时恰好选择了 S^* 中价值和最大的 m 个物品时
- 假设在 GKK 算法中，物品 k 是第一个 $\in S^*$ 但未放入的物品，有

$$
\begin{aligned}
\mathrm{OPT}(I) &< \mathrm{PTAS}_\epsilon(I) + v_k \\
&\leq \mathrm{PTAS}_\epsilon(I) + \frac{1}{m}\mathrm{PTAS}_\epsilon(I) \\
&\leq (1 + \frac{1}{m})\mathrm{PTAS}_\epsilon(I) \\
&\leq (1 + \epsilon)\mathrm{PTAS}_\epsilon(I)
\end{aligned}
$$

- 故 PTAS$_\epsilon$ 是 $(1+\epsilon)$-近似算法

算法 FPTAS$_\epsilon$，假设 $V = \max\{v_i | i = 1, 2, \cdots, n\}$

1. 事先设定 $\epsilon > 0$，令 $b = \max(\lfloor V/(1+\frac{1}{\epsilon})n \rfloor, 1)$
2. 将每个物品 i 的价值替换为 $v_i' = \lceil v_i/b \rceil$
3. 使用动态规划算法求解，得到物品集合 S，将 S 作为原问题的近似解

算法 FPTAS$_\epsilon$

- 设最优解为 S^*，若 $b = \max(\lfloor V/(1+\frac{1}{\epsilon})n \rfloor, 1) > 1$

$$
\begin{aligned}
& \mathrm{OPT}(I) - \mathrm{FPTAS}(I) \\
=& \sum_{i \in S^*} v_i - \sum_{i \in S} v_i \\
=& (\sum_{i \in S^*} v_i - b \sum_{i \in S^*} v_i') + (b \sum_{i \in S^*} v_i' - b \sum_{i \in S} v_i') + (b \sum_{i \in S} v_i' - \sum_{i \in S} v_i) \\
\le& b \sum_{i \in S} v_i' - \sum_{i \in S} v_i \\
<& bn \\
\le& V/(1+\frac{1}{\epsilon}) \\
\le& \mathrm{OPT}(I)/(1+\frac{1}{\epsilon})
\end{aligned}
$$

- 故 FPTAS$_\epsilon$ 是 $(1+\epsilon)$-近似算法

大纲

随机算法分类

- 拉斯维加斯型
  - 特点：若能得到结果，总是正确
  - 目的：通过将某个确定型算法的一些选择改为随机选择，改进平均情况下的时间复杂度
  - 关注点：期望时间复杂度
  - 有效性：期望时间复杂度是多项式的，且总能给出正确答案
- 蒙特卡洛型
  - 特点：有时会给出错误答案
  - 目的：通过适当牺牲正确性，提高效率
  - 关注点：出错概率
  - 有效性：多项式时间复杂度，出错概率不高于 $1/3$

随机快速排序

随机快速排序算法

if 元素个数 ≤ 1 **return** 原数组
随机选取任一元素 p
数组 $A \leftarrow$ 所有小于 p 的元素
数组 $B \leftarrow$ 所有等于 p 的元素
数组 $C \leftarrow$ 所有大于 p 的元素
递归处理 A 和 C
return A, B, C

随机快速排序的期望比较次数

定理

若数组元素个数为 n，则随机快速排序算法的期望比较次数 $T(N) \leq 2n \ln n$

随机快速排序的期望比较次数证明一

证明一：

- 递推式

$$T(n) = (n-1) + \frac{1}{n}\sum_{i=0}^{n-1}[T(i) + T(n-i-1)]$$

- 使用数学归纳法证明，重要推导如下：

$$\begin{aligned}
T(n) &\le (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} 2i\ln i \\
&\le (n-1) + \frac{2}{n}\int_{1}^{n} 2i\ln i di \\
&\le (n-1) + \frac{2}{n}(n^2\ln n - \frac{n^2}{2} + \frac{1}{2}) \\
&\le 2n\ln n
\end{aligned}$$

证明二:

- 定义随机变量 $X_{ij} = \begin{cases} 1 & \text{算法比较了第 } i \text{ 小元素和第 } j \text{ 小元素} \\ 0 & \text{其他} \end{cases}$

$$
\begin{aligned}
T(n) &= E(\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X_{ij}) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E(X_{ij}) \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P(X_{ij}=1) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= 2\sum_{i=1}^{n-1}(\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n-i+1}) \\
&< 2n(\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}) < 2n\ln n
\end{aligned}
$$

串相等测试

- 在两台机器 A 和 B 上分别存放两个长度相等的二进制长串 x 和 y
- 要求尽量减少通信的情况下，判断 x 和 y 是否相等

串相等测试：随机算法

随机算法

1. A 事先设定通信量限制 M
2. A 随机选取小于 M 的素数 p
3. A 将 p 和 $(x \ mod \ p)$ 发送给 B
4. B 测试 $(y \ mod \ p)$ 与 $(x \ mod \ p)$ 是否相等

出错条件：

$$p|(x-y)$$

定理 (素数定理)

小于 n 的素数个数

$$\pi(n) \approx \frac{t}{\ln t}$$

定理

若 $k < 2^n$，当 n 较大时，整除 k 的不同素数个数小于 $\pi(n)$。

串相等测试：随机算法分析

- 假设 |x|=|y|=L，若选择 $M \geq 2L^2$
- 出错概率约为

$$P_{error} < \frac{\pi(L)}{\pi(M)} \approx \frac{L/\ln L}{2L^2/\ln(2L^2)}$$
$$\approx \frac{L/\ln L}{2L^2/2\ln L} \leq \frac{1}{L}$$

- 传输位数约为

$$p\text{的位数} + (x\ mod\ p)\text{的位数}$$
$$\leq 2(\lfloor \log M \rfloor + 1) \approx 4\log L$$

给定两个二进制串 x 和 y，求 x 在 y 中第一次出现的位置。
其中 $|x| = m \leq n = |y|$。

经典算法：KMP，时间复杂度 $O(m + n)$。

随机算法

1. 令 $M = 2mn^2$，随机选取小于 M 的素数 p
2. 计算 $x \ mod \ p$
3. 滑动计算 $y[i..(i+m-1)] \ mod \ p$，测试是否与 $x \ mod \ p$ 相等

- 出错率不超过 $\frac{1}{n}$
- 时间复杂度 $O(m+n)$

大纲

热门应用

- 搜索引擎
- 聚类
- 特征模式匹配
- 视频缓存
- 自然语言处理
- 机器学习
- 计算机博弈

总结

- 近似算法和随机算法都是“不完美”的算法
- 近似算法和随机算法的有效性需要经过严密的分析
- 无论在学术科研，还是在商业应用上，近似算法和随机算法都发挥着十分重要的作用

谢谢！