Online Algorithm(在线算法)

所属领域：理论计算机，运筹学
顶会（计算机领域）：focs(Foundations of Computer Science),
soda(Symposium on Discrete Algorithms), stoc(Symposium on Theory of Computing).
顶刊（运筹学）：Operational Research, Management Science.

领域研究目的

1. 在只知道部分信息的情况下，做出决策（知道完全信息——最优化问题）。
2. 算法有性能保证（类似于近似算法）
3. 可以用多项式复杂度的算法求得NP-hard问题的近似解

接下来这个例子说明前两点（部分内容来自这学期中国运筹学会理事长胡旭东的《运筹学》的PPT）。

Ski Problem(滑雪问题)

题设

这个冬天你想在雁西湖周围的滑雪场滑雪，你面临两种选择：1. 在每次滑雪前花 1 元租一套滑雪用具。2.花 T 元买一套滑雪用具，这样整个冬天可以一直用它。

离线最佳策略

你知道你会滑 L 天，那么 $L < T$，则每次都**租**；$L >= T$，那么第一次滑雪前就**买**。花费 $\min\{L,T\}$元。

如果不知道你会滑多少次怎么办？因为你只能有两种行为——买或者租。因此，你的方法只能是前 k 次租，然后要滑 $k+1$ 次时，再买。

在线算法

在线算法：前 $T-1$ 次租，要是还要继续滑，就买。

性能保证：此算法的花费最多是最佳策略的2倍。
证明：假设 $L < T$，那么最佳策略是每次都租，而此算法也是租，两者价格相等。反之，$L >= T$，那么最佳策略是第一次就买（花 T 元），而此算法花费 $T-1+T$ 元，是 T 的 $2-1/T$倍。即：此算法的花费最多是最优策略的2倍。

能不能更好？不能了，这就是最佳确定型（deterministic）的在线算法，并且是紧的（tight）。要证明这点，只要证明最佳 k 就是 $T-1$。

证明：对于任意一个确定型算法（假设是第 $k+1$ 次时买滑雪用具），我们都考虑一个特殊情形：前 k 天一直在租，而要滑 $k+1$ 次时，买了滑雪用具，但是接着腿摔断了。于是此算法的花费是 $k+T$ 元。
而最佳策略的花费是$\min\{k+1,T\}$。
分类讨论：

1. $k+1 \leq T$，则$\frac{k+T}{k+1} = 1 + \frac{T-1}{k+1}$，为了让这个值尽可能小，$k$ 取最大值$T-1$。
2. $k+1 \geq T$，则$\frac{k+T}{T} = 1 + \frac{k}{T}$，为了让这个值尽可能小，$k$ 取最小值$T-1$。

因此，所有确定型算法对应的比值都不可能小于 $2-1/T$，而我们已经证明了 $k=T-1$ 时的比值为 $2-1/T$。因此，最佳 k 就是 $T-1$。

性能保证指标：竞争比（competitive ratio）

对于最小化费用问题，我们要保证在线算法的费用不超过离线最佳策略的 c 倍$(c>1)$。我们就用这个 c 来衡量在线算法的性能。c 被称为此算法的竞争比。理想中的 c 应该是个常数（比如2）或者和常量相关（$2-1/T$），而不能和在线算法中涉及到的变量相关（比如 k）。

近似算法

近似算法是理论计算机中一个被更多人知道，研究更加深入的领域（也更难，现在还有很多公开问题留待人解决），是为了针对某个计算机难解的问题（通常是NP-hard问题，比如TSP旅行商问题）设计求得近似解的算法，我们需要保证这个近似解和最优解之间的比值在一定范围（性能保证指标叫做近似比，approximation ratio），同时要求此近似算法要具备比较小的复杂度（多项式时间）。

竞争比和近似比关注的都是算法在最不利情况（worst case）下的表现，而不是平均情况（average case）。

Ski Problem 随机算法

以 p_i 的概率在第 i 天购买滑雪用具（当 $i \le T$ 时，p_i 随 i 变大）

$$p_i = \begin{cases} \left(\dfrac{T-1}{T}\right)^{T-i} \dfrac{1}{T(1-(1-1/T)^T)}, & i \le T, \\ 0, & i > T. \end{cases}$$

该算法在期望意义下的竞争比为 $e/(e-1) = 1.58197$，且是紧的（没有能做到更好的随机算法）。随机算法往往是比确定型算法好的，因为它使用随机性来对抗环境的worst case。

The Marriage Problem(择偶问题)

背景

有一次苏格拉底的弟子们问他什么是爱情，他没有直接回答，而是把弟子们带到了一片麦田上。他让两个弟子沿着一条田间小道一路走下去，捡一个最大的麦穗回来。他规定只能前进不许后退，且只允许捡一次。
一个弟子没走几步就捡起了一个很大的麦穗。然而继续往前走又看见了更大的麦穗，没有办法只好遗憾地走完了全程。 另一个弟子一路走一路看，每次看到大麦穗的时候，总觉得前面还会有更大的麦穗。直到最后已经快要走完整个麦田，他才发现最大的麦穗已经错过，只好空手而归。

题设

假设一位男孩儿有机会与 n 个女孩儿中的某一位结婚。在每一次恋爱过程中，若他最终选择与她结婚，则没有机会与后面的女孩儿结婚；若他选择放弃她，则没有机会再与她结婚。男孩儿怎么做出选择呢？

在线算法

我们还只是考虑确定型算法，一个经典的想法是：拒绝前 $r-1$ 个人，然后从第 r 个人开始，如果她比前面所有人都优秀，那就是她了；否则拒绝她。重复此过程直到选出了一位（有可能是最后一位）。

接下来我们要确定最优的r，使得上述方法选到最优的女孩的期望最大。
先给出结论，$r = \frac{1}{e}n$，然后选到最优女孩的期望是 $\frac{1}{e}$。

证明：拒绝前$r-1$对应的算法中，最优秀的女孩被选中的概率用P_r表示：

$$
\begin{aligned}
P_r &= \sum_{k=r}^{n} P(\text{第}k\text{位是最优的且被选中}) \\
&= \sum_{k=r}^{n} P(\text{第}k\text{位是最优})P(\text{第}k\text{位被选中}|\text{第}k\text{位是最优}) \\
&= \sum_{k=r}^{n} \frac{1}{n} P(\text{前}k-1\text{位中最优的出现在第}1\sim r-1\text{个位置}) \\
&= \sum_{k=r}^{n} \frac{1}{n}\frac{r-1}{k-1} \\
&= \frac{r-1}{n}\sum_{k=r}^{n}\frac{1}{k-1}
\end{aligned}
$$

P_r在 $r=1$时没有定义，但对应着的算法是拒绝前0个——直接选第一个就是最优的概率为$1/n$。
当 n 比较大的时候，$\sum_{k=r}^{n}\frac{1}{k-1}$约等于$\ln\frac{n}{r-1}$，则$\frac{r-1}{n}\ln\frac{n}{r-1}$最大值在$\frac{n}{r-1}$取$e$时得到（$f(x)=\frac{1}{x}\ln x$的最大值在$x=e$时取到且$f(e)=\frac{1}{e}$），此时$r^*=1+\frac{n}{e}$。
但我们发现，P_{r^*-1}约等于P_{r^*}：$P_r=\frac{r-1}{n}\sum_{k=r}^{n}\frac{1}{k-1}=\frac{r-1}{n}(\frac{1}{r-1}+\sum_{k=r+1}^{n}\frac{1}{k-1})=\frac{r-1}{n}(\frac{1}{r-1}+\sum_{k=r+1}^{n}\frac{1}{k-1})\sim\frac{r-1}{n}(\frac{1}{r-1}+\ln\frac{n}{r})$。
当$r=\frac{n}{e}$时，$P_r\sim\frac{r-1}{n}(\frac{1}{r-1}+1)=\frac{r}{n}=\frac{1}{e}$。
这个小问题是我自己推导时发现的，wikipedia上（Secretary problem）说的r^*就是$\frac{n}{e}$。

在线算法的主流方法

- primal-dual：对于最小化花费的原问题，找到一个对偶问题，同时更新原问题的变量和对偶变量，主要思路是$P^*\geq D^*\geq\frac{1}{c}P_D$。第一个不等式是因为弱对偶性，即原问题目标函数最小值会大于对偶问题目标函数的最大值（两个问题的极值点大概率不一样），第二个不等式就是我们设计算法的难点，其中P_D 就是D^* 对应在原问题上的解，这样我们就证明了竞争比为 c。这是应用最广泛的方法。
- Lyapunov Optimization：将长期约束解耦到每个时刻，每个时刻同时优化目标函数和约束条件，两者权重由V控制，可以实现目标函数以$O(\frac{1}{V})$的误差接近最优值，其副作用为平均的队列大小为$O(V)$，这意味着要花更长的时间，长期约束才能被满足。