PKU-04833060
算法设计与分析『实验班』

第14讲 在线算法

罗国杰

gluo@pku.edu.cn

2025年春季学期

与经典算法有何不同?

经典离线算法（offline algorithms）在问题求解前已经获得与问题相关的完全信息

在线算法则是在算法实行过程中不断给出相关信息

由于不具备完全信息，所以在线算法的解只是局部最优的而无法保证整体最优

经典案例：K服务问题

给定一个有 n 个顶点的图 G，其 n 个顶点均为服务对象，随时会提出服务要求。

现有 k 辆服务车按提出要求的先后顺序来往服务于 n 个顶点之间。

服务要求是在服务过程中一个一个给出的，每一时刻只知道之前的服务要求序列。

问如何调度 k 辆服务车使他们在服务过程中移动的距离最短?

K服务问题：问题分析

由于事先并不知道所有的服务，所以显然不能保证给出的算法能有最优解

只能给出近似解

求助于贪心算法

贪心策略导致的效率与离线算法的差异究竟多大？

K服务问题：贪心策略的选择

策略一：当顶点 i 提出服务要求时，找出距顶点 i 最近的服务车 j，派 j 前往 i 处。

策略二：设第 i 辆车已经移动的距离为 $d[i]$，当顶点 j 提出服务要求时选择第 t 辆车使得 $d[t] + dist(t, j) = \min\{d[i] + dist(i, j)\}$，其中 $1 \le i \le k$，$dist$ 表示两点间的距离。

K服务问题：考虑策略一

考虑当 $k=2$

G 为三个顶点的一条路的简单实例，其中顶点从左到右依次为 A，B，C

$|AB| = 1$，$|BC| = 2$，服务车开始停在 B，C 上，服务序列为 ABABABAB……

此时贪心策略会将停在 B 处的服务车来回移动

设服务序列长 m，则移动的总距离为 $m = O(m)$

K服务问题：考虑策略一（续）

对于 $m>2$ 时，最优的调度方案显然是将 B 处的车移到 A 处，再将 C 处的车移到 B 处，这样就能一劳永逸了，此时移动距离仅为 $3 = O(1)$。

可见贪心策略一与最优算法距离的比值可达到 $m/3$，比值可随 m 增大取得任意大的值，这样的算法显然是不好的。

K服务问题：考虑策略二

对于策略二，上面所描述的实例移动距离不超过 5

后面将会证明，该算法的移动距离不会超过最优值的 2 倍。

K服务问题：分析结果

从上面的分析过程可以看出，衡量一个在线算法的效率的一个重要标准就是看它和最优的离线算法的开销的比值，比值越小，说明算法的效率越高。

竞争比（Competitive Ratio）

设在线算法 A 的输入序列为 σ，代价为 $C_A(\sigma)$，最优离线算法 OPT 的代价为 $C_{OPT}(\sigma)$，如果存在非负整数 α 和 c（与序列长度 $|\sigma|$ 无关的常数），使得 $C_A(\sigma) \leq \alpha C_{OPT}(\sigma) + c$ 对任何输入序列 σ 都成立，则称算法 A 是 α-竞争的，常数 α 称为算法 A 的竞争比。

当算法 A 的竞争比不可能再改进时，称算法 A 是最优在线算法。

在线算法的优劣标准

如果对于某一个特殊的输入序列 σ，找不到一个非负常数 α 使得 $C_A(\sigma) \leq \alpha C_{OPT}(\sigma)$，这样的算法显然是不好的。即找不到竞争比的算法显然是不好的。

应该找一个尽量小的竞争比 α 使得 $C_A(\sigma) \leq \alpha C_{OPT}(\sigma)$ 成立。

基于竞争比的算法评价——页调度问题

内存按存取速度分为高速的片上缓存和相对低速的片外内存。

高速缓存可分为 k 个页面，其余页面在内存中。

页调度问题的输入是内存访问请求序列 $\sigma = \sigma(1)\ \sigma(2)\ \sigma(3)\ \ldots\ \sigma(m)$。

当页面 $\sigma(i)$ 不在缓存中时，需将其调入缓存中，同时缓存中的某一页面调回内存

设计算法使调换的总次数最小（某一时刻不知道后续的访问序列）。

常见的页调度算法

Last In First Out (LIFO):

- $\sigma(i)$ 发生缺失时，将最近调入缓存的页面与 $\sigma(i)$ 交换。

First In First Out (FIFO):

- 将最早调入缓存的页面与 $\sigma(i)$ 交换

Least Recently Used (LRU):

- 将上一次访问时间最早的页面与 $\sigma(i)$ 交换。

Least Frequently Used (LFU):

- 将访问次数最少的页面与 $\sigma(i)$ 交换。

Furthest in Future (FF):

- 最优的离线调度算法，见 第5讲“贪心算法 (1/2)”

LRU算法的竞争比分析（1/6）

设高速缓存可容纳 k 个页面，任意请求序列 σ,

- 在线算法 LRU 的代价为 $C_{LRU}(\sigma)$
- 最优离线算法 OPT 的代价为 $C_{OPT}(\sigma)$

下面证明在线算法 LRU 的竞争比为 k

- 即证对于任意的内存访问请求序列

  $\sigma = \sigma(1), \ldots, \sigma(m)$

  有

  $C_{LRU}(\sigma) \le k\, C_{OPT}(\sigma)$。

LRU算法的竞争比分析（2/6）

证明 LRU 的竞争比为 k 的主要思路：

根据 LRU 算法的结果将 σ 分为若干阶段 $P(0), P(1), \ldots$，使得 $P(0)$ 最多有 k 个页面缺失，而对其余的 $P(i)$，每个都有 k 个页面缺失。这样的阶段划分是可行的，只要从尾部开始扫描，遇到 k 个页面缺失就截取一段新的阶段。

下面证明最优离线算法 OPT 在每个阶段 $P(i)$ 至少产生 1 个页面缺失。

LRU算法的竞争比分析（3/6）

不失一般性假设初始状态下两种算法都有相同的高速缓存

当 $i=0$ 时，LRU 在 $P(0)$ 阶段产生第 1 个页面缺失时，因为初始状态相同，算法 OPT 也产生 1 个页面缺失。

对于 $i \geq 1$ 的阶段 $P(i)$，记 $\sigma(t_i)$ 是阶段 $P(i)$ 的第 1 个页面请求，则 $\sigma(t_{i+1}-1)$ 是阶段 $P(i)$ 的最后 1 个页面请求，且 $p=\sigma(t_i-1)$ 是 $P(i-1)$ 最后 1 个页面访问请求。

可以证明：$P(i)$ 中有 k 个不同于 $\sigma(t_i-1)$ 的互不相同的页面访问请求。

（分情况讨论）

- $P(i)$ 缺失页面互不相同，且包含 $\sigma(t_i-1)$
- $P(i)$ 缺失页面互不相同，且不含 $\sigma(t_i-1)$ √
- $P(i)$ 有某页面产生两次缺失

LRU算法的竞争比分析（4/6）

（$P(i)$ 的缺失页面互不相同，且包含 $\sigma(t_i-1)$ 的情况）

当算法在阶段 $P(i)$ 产生页面缺失的页面访问请求互不相同，但有 1 次在页面访问请求 $p=\sigma(t_i-1)$ 时产生页面缺失。

设在某 $t \geq t_i$ 时刻的请求 $\sigma(t)$ 时，页面 p 被调出高速缓存。

考虑子序列 $\sigma(t_i-1)...\sigma(t)$:

在 t 时刻页面 p 被调出缓存时，由于 p 是上次访问时间最早的页面，

可知 **缓存其余 $k-1$ 个页面** 在 $\sigma(t_i)...\sigma(t-1)$ 中曾被访问;

而 **请求页面 $\sigma(t)$** 是 t 时刻不在缓存中的页面。

因此，$P(i)$ 有 k 个不同于 $\sigma(t_i-1)$ 的互不相同的页面访问请求。

LRU算法的竞争比分析（5/6）

（$P(i)$ 的缺失页面互不相同，且不含 $\sigma(t_i-1)$ 的情况）

算法 LRU 在阶段 $P(i)$ 产生 k 个页面缺失的页面访问请求互不相同且不同于 $\sigma(t_i-1)$ 时，结论成立。

（$P(i)$ 有某页面产生两次缺失的情况）

若 LRU 在阶段 $P(i)$ 时对某页面访问请求 q 产生两次页面缺失：设 $\sigma(s_1)=\sigma(s_2)=q$ 产生页面缺失，且 $t_i \le s_1 < s_2 \le t_{i+1}-1$。说明 q 在 $s_1 < t < s_2$ 的某次访问请求 $\sigma(t)$ 被调出高速缓存。与上页的论证类似，子序列 $\sigma(s_1)\ldots\sigma(t)$ 中包含了 $k+1$ 个不同的页面访问请求，则有 k 个不同于 $\sigma(t_i-1)$ 的互不相同的页面访问请求。

LRU算法的竞争比分析（6/6）

通过上面讨论可知，$\sigma(t_i-1)$ 是阶段 $P(i-1)$ 的最后一个页面访问请求，因此，在阶段 $P(i)$ 开始时，页面 $\sigma(t_i-1)$ 在高速缓存中。而在阶段 $P(i)$ 中又有 k 个不同于 $\sigma(t_i-1)$ 的互不相同的页面访问请求。由此可见任何一个算法在阶段 $P(i)$ 都至少产生 1 个页面缺失，包括最优离线算法 OPT。

这就证明了 $C_{LRU}(\sigma) \le k\, C_{OPT}(\sigma)$，
即在线算法 LRU 的竞争比为 k。

LRU算法的最优性分析（1/2）

进一步分析表明，在线算法 LRU 是最优在线算法。换句话说，如果算法 A 是页调度的在线算法且竞争比为 α，则 $\alpha \geq k$。

设 $S = \{p_1, p_2, p_3, \ldots, p_{k+1}\}$ 是任意 $k+1$ 个访问页面的集合，在初始状态下算法 A 与 OPT 都有相同的高速缓存。访问的每次的页面请求都在 S 中。

LRU算法的最优性分析（2/2）

考虑这样一个访问序列 σ，它的每个访问都使该页面不在 A 的高速缓存中，则算法 A 每次都要产生一个页面缺失。

对于最优离线算法 OPT 来说，k 次连续的访问，最多产生 1 次页面缺失。

- Furthest in Future (FF) 算法，k 次连续访问前换出的页面再发生缺页，肯定是第 k 次或更晚发生

可见 $C_A(\sigma) \geq k\, C_{OPT}(\sigma)$

K服务问题的一般情况

距离空间 V 是一个点集以及定义在该点上的一个距离函数 d: $(V \times V) \rightarrow R$，且满足如下性质：

$d(u,v) \geq 0$，$\forall u,v \in V$;

$d(u,v) = 0 \Leftrightarrow u = v$;

$d(u,v) = d(v,u)$, $\forall u,v \in V$;

$d(u,w) + d(w,v) \geq d(u,v)$, $\forall u,v,w \in V$;

其余条件与开头的例子一样

页调度问题是K服务问题的特殊情形

高速缓存中的 k 个页面是 k 个服务。

页面缺失时，缓存中页面与内存中页面的交换看成是 1 次移动服务，其代价为 1.

因此，页调度问题是 k 服务问题中所有不同点对间距离均为 1 的特殊情形。

- i.e.，边距离均为 1 的完全图上的 k 服务问题

K服务问题：竞争比的下界

由前面页调度问题的下界可推广到 k 服务问题。
即竞争比 $\alpha \geq k$。

下面针对在线算法 A 构造一个特殊的服务请求序列 σ 以及另外 k 个算法 $A_1, A_2, A_3, \ldots, A_k$ 使得 $C_A(\sigma) \geq \Sigma C_{Ai}(\sigma)$。

由此推出存在算法 A_i 使得
$C_A(\sigma) \geq k\, C_{Ai}(\sigma) \geq k\, C_{OPT}(\sigma)$，从而 $\alpha \geq k$。

K服务问题：竞争比下界的证明（1/4）

设 $|V| = k+1$，初始时：
k 个服务位于不同位置，另有 1 个空位置。

构造服务序列：$\sigma = \sigma(1)\sigma(2)\sigma(3),\ldots,\sigma(m)$，
其中每个服务请求 $\sigma(i)$ 都恰好发生在当时的空位置 h 处。

对于 $1 \le t \le m$，设 $\sigma(t) = x_t$，记 x_{m+1} 是最终的空位置，则有

$$C_A(\sigma) = \sum_{t=1}^{m} d(x_{t+1}, x_t)$$

- 根据 $t+1$ 时刻 x_{t+1} 是空位置可知，在 t 时刻发出服务请求 x_t 时，移动的是在 x_{t+1} 的服务
- 因此，t 时刻的服务距离是 $d(x_{t+1},x_t)$

K服务问题：竞争比下界的证明（2/4）

设 $y_1, y_2, \ldots, y_k$ 是初始时算法 A 的 k 个服务的位置。
构造算法 A_i 如下，其中 $1 \le i \le k$

- 初始状态 A_i 的 k 个服务占据 V 中除了 y_i 外的 k 个位置。
- 对于服务请求 $\sigma(t) = x_t$，如果没有服务处于位置 x_t，则算法就将位于 x_{t-1} 处的服务移动到 x_t 处；否则不做任何事情。
- 设 V_i 是算法 A_i 的 k 个服务占据的点的集合。
- 可以证明：在响应服务请求 $\sigma = \sigma(1)\sigma(2)\sigma(3),\ldots,\sigma(m)$ 的整个过程中，每个 V_i 互不相同。

K服务问题：竞争比下界的证明（3/4）

每个 V_i 互不相同，则对于任何服务请求 $\sigma(t) = x_t$，只有 1 个算法 A_i 需要响应服务请求。因此，有

$$\sum_{i=1}^{k} C_{A_i}(\sigma) = \sum_{t=2}^{m} d(x_{t-1}, x_t)$$

$$C_A(\sigma) = \sum_{t=1}^{m} d(x_{t+1}, x_t) \geq \sum_{i=1}^{k} C_{A_i}(\sigma)$$

下面用数学归纳法证明，
在响应服务请求序列的整个过程中 V_i 互不相同。

初始时结论显然成立。

K服务问题：竞争比下界的证明（4/4）

设在服务请求 $\sigma(t-1)$ 时成立，考察请求 $\sigma(t) = x_t$。

此时有 $x_{t-1} \in V_i$，$1 \le i \le k$。

对任意两个不同集合 V_j 和 V_l，$1 \le j, l \le k$。
由于 V_j 和 V_l 不同，则 x_t 不可能同时不属于 V_j 和 V_l。

1）如果 $x_t \in V_j$ 且 $x_t \in V_l$，则响应之后两个集合都没有变，从而仍然不同。

2）若 x_t 只属于其中 1 个集合（不妨假设为 V_j），
则响应之后 x_{t-1} 不属于 V_j，而 $x_{t-1} \in V_l$，从而有 V_j 不等于 V_l

由数学归纳法知所属结论成立。

K服务猜想（k-server Conjecture）

目前对于 k 服务问题的一些特殊情形找到了竞争比为 k 的在线算法。

一般情况下，很难找到竞争比为 k 的在线算法。

在线算法领域普遍猜测，距离空间中的 k 服务问题存在竞争比为 k 的在线算法。

这个猜测称为 k 服务猜想。

Mark S. Manasse, Lyle A. McGeoch, and Daniel Dominic Sleator. “Competitive algorithms for on-line problems.” STOC 1988.

对称移动算法

竞争比为 k，适用于直线上的 k 服务问题。

响应服务请求 $\sigma(t) = x_t$ 时，对称移动算法 A 采用如下策略：

当 x_t 位于 2 个服务 s_i 和 s_j 之间时，s_i 和 s_j 同时向 x_t 移动距离 $d = \min\{|s_i - x_t|, |s_j - x_t|\}$。

当所有 k 个服务位于 x_t 同一侧时，选取距 x_t 最近的服务 s_i 向 x_t 移动距离 $|s_i - x_t|$。

对称移动算法的证明

利用势函数的方法证明，设对称移动算法 A 的 k 个服务在直线 L 上的位置（从左到右）为 $s_1,s_2,s_3,\ldots,s_k$，最优离线算法的 k 个服务在直线 L 上的位置为 $t_1,t_2,t_3,\ldots,t_k$。

进一步可设两个序列都是升序排列，否则可对服务重新编号。

对请求序列 σ，A 的代价为 $C_A(\sigma)$，最优离线算法的代价是 $C_{OPT}(\sigma)$。

具体的每个服务请求 $\sigma(i)$，A 的代价为 $C_A(i)$，OPT 的代价为 $C_{OPT}(i)$。

对称移动算法的证明（续1）

定义势函数：

$$\Phi = k\sum_{i=1}^{k}\left|t_i - s_i\right| + \sum_{i<j}(s_j - s_i)$$

不失一般性，对任意的服务请求 $\sigma(i)$，让 OPT 先响应，然后算法 A 再响应。

OPT 响应之前势函数的值为 Φ_{i-1}，OPT 响应服务请求之后的势函数值为 Ψ_i，算法 A 响应服务请求之后势函数的值为 Φ_i，$1 \le i \le m$。

算法 OPT 响应之后的增量 $\alpha_i = \Psi_i - \Phi_{i-1}$

算法 A 响应服务请求后势函数值的减量 $\beta_i = \Psi_i - \Phi_i$

算法 OPT 和算法 A 的每步响应后，势函数增量 $\alpha_i - \beta_i$

对称移动算法的证明（续2）

下面证明对于 $1 \le i \le m$，有：

$a_i \le k\, C_{OPT}(i)$

$\beta_i \ge C_A(i)$

设 $\Phi = \Omega + \Theta$，其中

$$\Omega = k \sum_{i=1}^{k} \left| t_i - s_i \right|$$

$$\Theta = \sum_{i<j} (s_j - s_i)$$

对称移动算法的证明（续3）

算法 OPT 响应服务请求 $\sigma(i) = y_i$ 时，服务 t_j 移动到 y_i，其代价 $C_{OPT}(i) = |y_i - t_j|$。

Ω 的值最多增加 $kC_{OPT}(i)$，而 Θ 的值不变，从而，$a_i \le k\,C_{OPT}(i)$。

- $|y_i - s_j| - |t_j - s_j| \le |y_i - t_j| = C_{OPT}(i) \quad \Rightarrow \quad a_i = \Delta\Omega \le kC_{OPT}(i)$

设算法 OPT 响应服务请求 $\sigma(i) = y_i$ 后，由算法 A 响应请求 $\sigma(i) = y_i$，下面分两种情况：

- 情况1：算法 A 的所有服务在 y_i 的同一侧
- 情况2：y_i 位于服务 s_r 和 s_{r+1} 之间

对称移动算法的证明：情况一

算法 A 的所有服务在 y_i 的同一侧，不妨设为右侧，且距离 y_i 最近的服务是 s_1。

显然有 $s_1 \geq t_j \geq t_1$，将 s_1 移动到 y_i，$C_A(i) = |s_1 - y_i|$。

Ω 的值减少了 $kC_A(i)$。

- s_1 与 t_1 的距离减少了 $C_A(i)$

Θ 的值增加了 $(k-1)C_A(i)$。

- s_1 与 $s_2, s_3, \ldots, s_k$ 的距离均增加了 $C_A(i)$

从而 $\beta_i = C_A(i)$

对称移动算法的证明：情况二

y_i 位于服务 s_r 和 s_{r+1} 之间，即 $s_r < y_i < s_{r+1}$，不妨设 s_r 距 y_i 较近。

算法 A 的代价为 $C_A(i) = 2|s_r - y_i|$。

Ω 中只有第 r 项和第 $r+1$ 项发生变化。

若 OPT 的服务 t_j 满足 $j \le r$，此时算法 A 响应前后均有 $t_r \ge s_r$，
则 Ω 第 r 项减少 $k|s_r - y_i|$，而第 $r+1$ 项最多增加 $k|s_r - y_i|$，
所以 Ω 值不增。

$j \ge r+1$ 时类似。

对称移动算法的证明：情况二（续）

考察 Θ 的变化， s_r 和 s_{r+1} 均产生移动，使 Θ 值增加了

$$\left|s_r - y_i\right|[-(k-r)+(r-1)-r+(k-(r+1))]$$
$$= -2\left|s_r - y_i\right| = -C_A(i)$$

从而有 $\beta_i \geq C_A(i)$

对称移动算法的证明（续4）

$a_i \le k\, C_{OPT}(i)$

$\beta_i \ge C_A(i)$ （即 $-\beta_i \le -C_A(i)$）

将上述两式从 1 到 m 叠加，可得

$$\Phi_m - \Phi_0 \le k\sum_{i=1}^{m} C_{OPT}(i) - \sum_{l=1}^{M} C_A(i),\ \Phi_m \ge 0$$

$$\Rightarrow \sum_{l=1}^{M} C_A(i) \le k\sum_{i=1}^{m} C_{OPT}(i) + \Phi_0$$

由服务请求序列的任意性可知，算法 A 的竞争比为 k。

对称移动算法的推广

距离空间为树，任意两点 x，y 间的距离是树中连接 x 和 y 的简单路的长度，记为 $d(x,y)$。

用 $s_1s_2s_3\ldots s_k$ 表示 k 个服务在树 T 中的位置。在相应服务请求 $\sigma(t) = x_t$ 时，如果连接服务 s_i 和 x_t 的简单路上没有别的服务，则称服务 s_i 为有效服务；否则称为无效服务。

对称移动算法的推广（续）

移动策略：

所有有效服务 s_i 以相同的速度向 x_t 移动。

S_i 停止移动的条件是：

由于其它的有效服务的移动使自身变为无效服务。

或者已经有服务到达 x_t。

推广算法的证明思路

设 k 个服务在树 T 中的位置为 $s_1 s_2 s_3 \ldots s_k$。

最优离线算法 OPT 的 k 个服务的位置为 $t_1 t_2 t_3 \ldots t_k$。

定义一个带权二分图 G，满足 $s_1 s_2 s_3 \ldots s_k$ 图中顶点 $v_1 v_2 v_3 \ldots v_k$；对应图中顶点 $u_1 u_2 u_3 \ldots u_k$。

边 (v_i, u_j) 的权为 $d(s_i, t_j), 1 \le i, j \le k$。

设 M_{min} 是图 G 的 1 个最小权匹配，$|M_{min}|$为其权值。

推广算法的证明思路（续）

定义势函数：

$$\Phi = k\left|M_{\min}\right| + \sum_{i<j} d(s_i, s_j)$$

与直线情形类似可证 $\alpha_i \le k\, C_{OPT}(i)$ 和 $\beta_i \ge C_A(i)$。

从而该算法的竞争比也为 k。

其它在线算法

在线 Steiner 树问题

在线任务调度问题

负载平衡问题

本讲小结

简要介绍了在线算法的基本概念和与离线算法的区别

引入竞争比为工具进行经典的K服务问题的分析和推广

- 页调度问题（特殊k服务问题）的LRU算法与竞争比证明
- 一般K服务问题的“竞争比”下界
- 直线上K服务问题的k竞争比算法（对称移动）+ 证明
- 树上K服务问题的K竞争比算法 + 证明思路

在线算法的设计与证明都必须用到严谨而灵活的数学推理

在线算法具有很广泛的应用性和实用性