在线算法——设计与分析

北京大学信息科学技术学院

与经典算法有何不同?

- 基本的离线算法（off line algorithms）在问题求解之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，所以在线算法找到的解只是局部最优的而无法保证整体最优

经典实例：*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？

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？

贪心策略的选择

- 策略一：

当顶点i提出服务要求时，找出据顶点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。

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

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。
- 可见贪心策略一与最优算法效率比可达到$m/3$，当m无限大时，这样的算法显然是不好的。

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。

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

- 设在线算法A的输入序列为σ，耗费为$C_A(\sigma)$，最优离线算法OPT的耗费为$C_{OPT}(\sigma)$，如果存在非负整数α和c，使得$C_A(\sigma) \le \alpha\, C_{OPT}(\sigma) + c$ 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

- 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$C_A(\sigma) \le \alpha\, C_{OPT}(\sigma)$，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$C_A(\sigma) \le \alpha\, C_{OPT}(\sigma)$ 成立。

基于竞争比的算法评价——页调度问题

- 问题描述：
  - 内存按存取速度分为高速缓存和低速内存。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

- **LIFO (last in first out):**
  - $\sigma(i)$发生缺失时，将最近调入缓存的页面与$\sigma(i)$交换。
- **FIFO(first in first out):**
  - 将最早调入缓存的页面与$\sigma(i)$交换
- **LRU(least recently uesd):**
  - 将最近访问时间最早的页面与$\sigma(i)$交换。
- **LFU(least frequently used):**
  - 将访问次数最少的页面与$\sigma(i)$交换。

LRU算法的分析

- 设高速缓存可容纳k个页面，任意请求序列σ，
  - 在线算法LRU的耗费为$C_{LRU}(\sigma)$
  - 最优离线算法OPT的耗费为$C_{OPT}(\sigma)$
- 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

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

LRU算法的分析(续)

- 根据**LRU**算法的结果将σ分为若干阶段$P(0)$, $P(1)$,…，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算法的分析(续)

- 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。
- 对于阶段$P(i)$，$i \geq 1$，设$\sigma(t_i)$是阶段$P(i)$的第1个页面请求，$\sigma(t_{i+1}-1)$是阶段$P(i)$的最后1个页面请求，且p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p的互不相同的页面访问请求。

LRU算法的分析(续)

第一种情况：

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p时，结论成立。

第二种情况：

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$s_1$之后$s_2$之前的某次访问请求$\sigma(t)$被调出高速缓存，当$q$被调出时，它是当前缓存中最近访问时间最早的页面，这说明子序列$\sigma(s_1)$到$\sigma(t)$中包含了$k+1$个不同的页面访问请求，则有k个不同于p的互不相同的页面访问请求。

LRU算法的分析(续)

- 第三种情况
  - 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产生页面缺失时。
  - 设在请求$\sigma(t) = p$且$t \geq t_i$时页面p被调出高速缓存。与前面的论证类似，即有k个不同于p的互不相同的页面访问请求

LRU算法的分析(续)

- 通过上面讨论可知，p是阶段$P(i\text{-}1)$的最后一个页面访问请求，因此，在阶段$P(i)$开始时，页面p在高速缓存中。而在阶段$P(i)$中又有k个不同于p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$C_{\mathrm{LRU}}(\sigma) \le k\, C_{\mathrm{OPT}}(\sigma)$，即在线算法**LRU**的竞争比为$k$。

LRU算法的分析(续)

- 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算法的分析(续)

- 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。
- 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，其耗费为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。

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$C_A(\sigma) \geq \sum C_{Ai}(\sigma)$。
- 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$\alpha \geq k$。

下界的证明

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

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

下界证明（续）

- 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。

下界的证明（续）

- 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

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

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

- 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。

下界的证明(续)

- 设在服务请求$\sigma(t\text{-}1)$时成立，考察请求$\sigma(t) = x_t$。
- 此时有$x_{t\text{-}1} \in V_i$，$1 \le i \le k$。
- 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。
- 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。
- 若x_t只属于其中1个集合（不妨假设为V_l），
  则响应之后$x_{t\text{-}1}$不属于V_j，
  而$x_{t\text{-}1} \in V_l$，
  从而有V_j不等于V_l
- 由数学归纳法知所属结论成立。

K服务猜测

- 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。
- 计算机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*服务猜测。

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**的耗费为$C_A(\sigma)$，最优离线算法的耗费是$C_{OPT}(\sigma)$。具体的每个服务请求$\sigma(i)$，**A**的耗费为$C_A(i)$，**OPT**的耗费为$C_{OPT}(i)$。

对称移动算法的证明（续）

- 定义势函数：$$\Phi = k\sum_{i=1}^{k}\left|t_i - s_i\right| + \sum_{i<j}\left(s_j - s_i\right)$$
- 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$\beta_i = \Psi_i - \Phi_i$

对称移动算法的证明（续）

- 下面证明对于$1 \le i \le m$，有：
- $\alpha_i \le k\, C_{\mathrm{OPT}}(i)$
- $\beta_i \ge C_{\mathrm{A}}(i)$
- 设$\Phi = \Omega + \Theta$，其中

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

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

对称移动算法的证明（续）

- 算法**OPT**响应服务请求$\sigma(i) = y_i$时，服务t_j移动到y_i，其耗费$C_{OPT}(i) = |t_j - y_i|$。 Ω的值最多增加$kC_{OPT}(i)$，而Θ的值不变，从而， $\alpha_i \le k\, C_{OPT}(i)$ 。
- 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$kC_A(i)$。
- Θ的值增加了$(k - 1)C_A(i)$。
- 从而$\beta_i = C_A(i)$

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的耗费为$C_A(i) = 2|s_r - y_i|$。
- 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。

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

$$
\begin{aligned}
&\left|s_r - y_i\right|\left[-\left(k-r\right)+\left(r-1\right)-r+\left(k-\left(r+1\right)\right)\right] \\
&= -2\left|s_r - y_i\right| = -C_A\left(i\right)
\end{aligned}
$$

- 从而有 $\beta_i \geq C_A(i)$

对称移动算法的证明（续）

- $\alpha_i \leq k\, C_{\mathrm{OPT}}(i)$
- $\beta_i \geq C_{\mathrm{A}}(i)$
- 将上述两式从1到m叠加，可得

$$\Phi_m - \Phi_0 \leq k \sum_{i=1}^{m} C_{OPT}(i) - \sum_{I=1}^{M} C_A(i), \Phi_m \geq 0$$

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

- 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

- 设k个服务在树T中的位置为$s_1s_2s_3\ldots s_k$。
- 最优离线算法OPT的k个服务的位置为$t_1t_2t_3\ldots t_k$。
- 定义一个带权二分图G，满足$s_1s_2s_3\ldots s_k$图中顶点$v_1v_2v_3\ldots v_k$；对应图中顶点$u_1u_2u_3\ldots u_k$。
- 边(v_i,u_j)的权为$d(s_i,t_j),1\leq i,j\leq k$。
- 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（续）

- 定义势函数：

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

- 与直线情形类似可证$\alpha_i \le k\, C_{\mathrm{OPT}}(i)$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

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
- 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，不容易掌握
- 在线算法具有很广泛的应用性和实用性