广告系统预算平滑

最近在做预算平滑（budget pacing），所以总结一些预算平滑的方法，总结下来预算平滑的好处如下

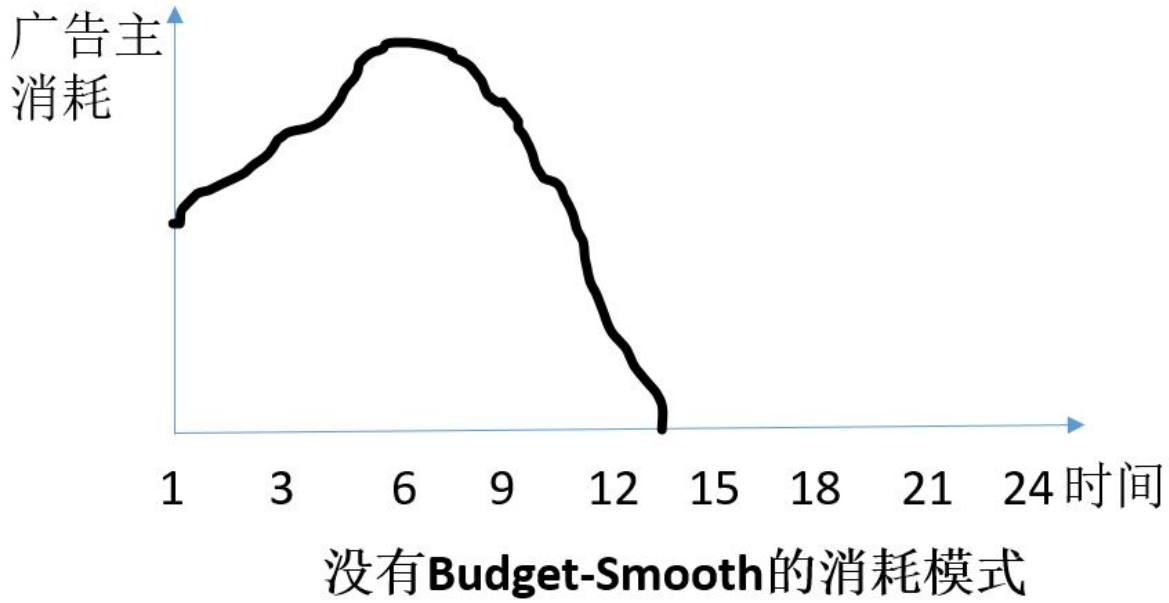
对于广告主：

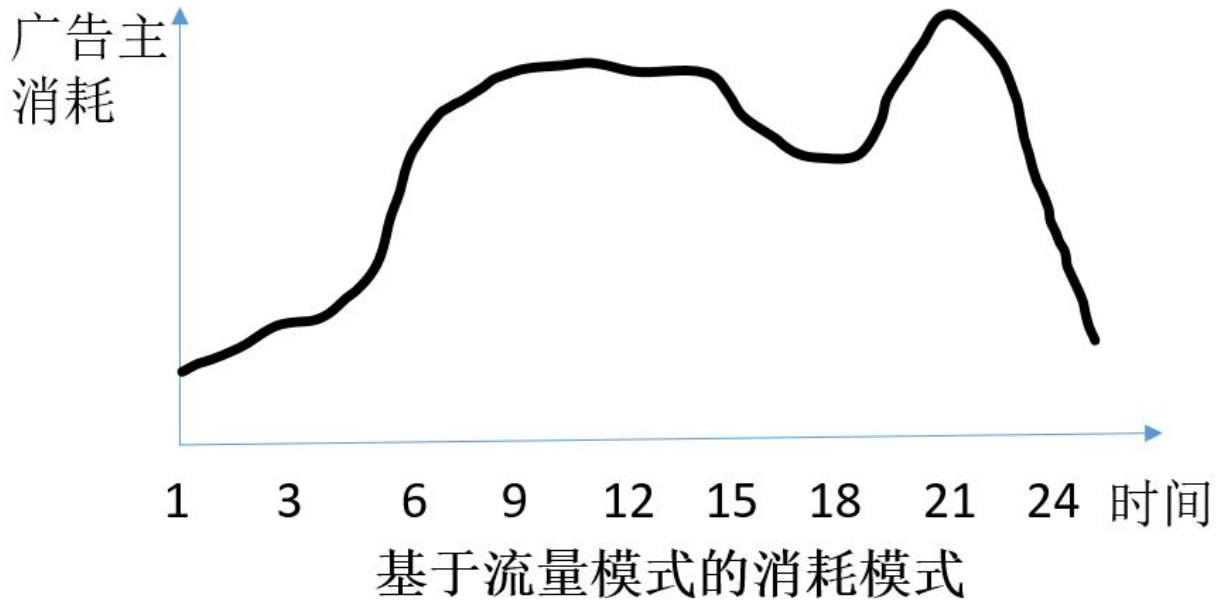
- 避免预算在每天初期耗尽，无法参加后期的竞价
- 帮助触达更多的用户
- 有机会提升效果

对于广告平台来说：

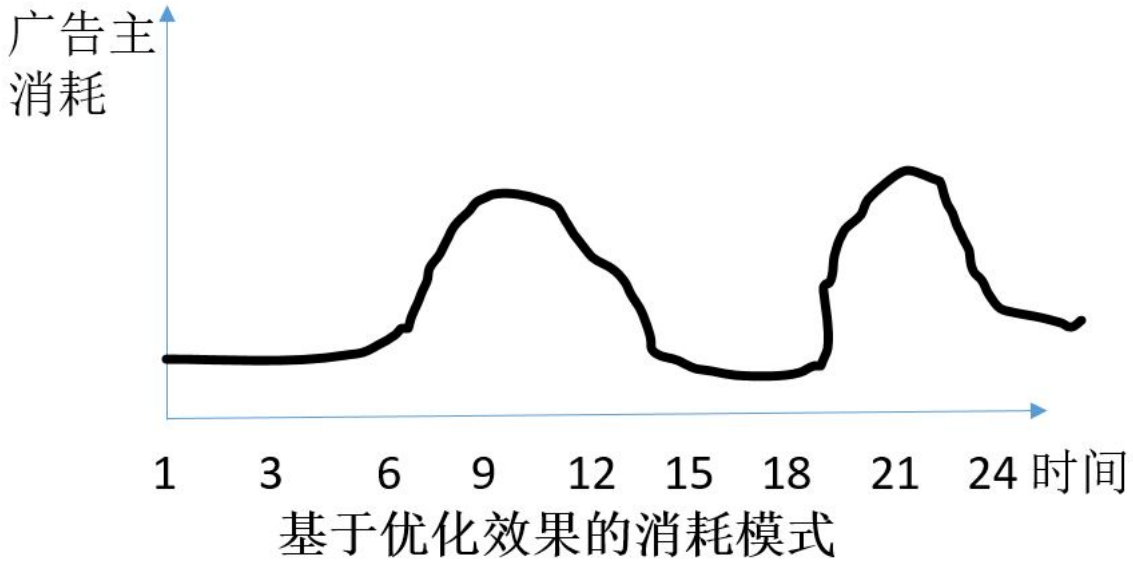
- 避免竞争都集中在每天的开始，整个系统更加稳定
- 为广告主提供更多的选项，优化效果

广告消耗的几种模式：



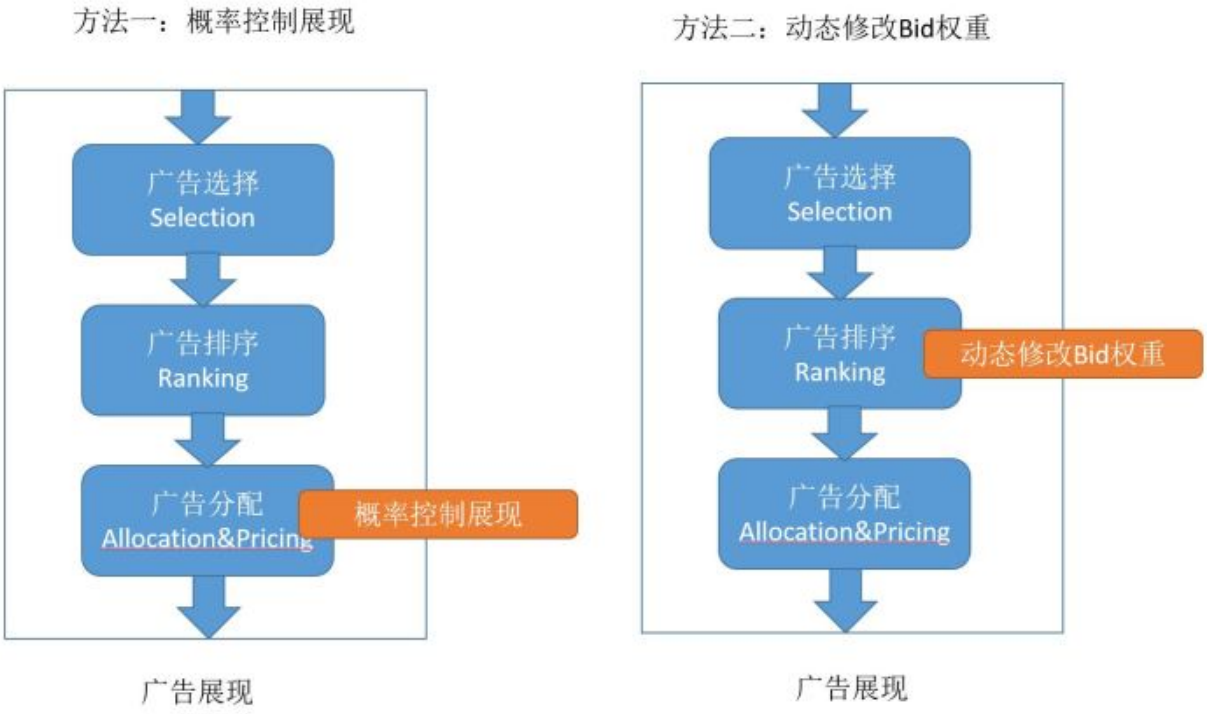


基于流量模式的消耗模式



基于优化效果的消耗模式

预算平滑的两种方式：



本文总结了三种smart pacing的方式分别来自于三篇论文

- 《Budget Pacing for Targeted Online Advertisements at LinkedIn》
- 《Smart Pacing for Effective Online Ad Campaign Optimization》
- 《Real time bid optimization with smooth budget delivery in online advertising》

Budget Pacing for Targeted Online Advertisements at LinkedIn

本篇论文是linkedin在2014年发表的预算控制的论文，策略也不复杂，很强的实践性和工程性，主要侧重点在于机制设计，并且是基于流量模式平滑，即怎么让广告在一天里平滑消耗。

原理

算法主要思想是令每一个campaign的消耗趋势与其曝光趋势基本保持一致，以天为时间单位，基于其曝光情况计算当前时间片的累积实际消耗和累积期望消耗的大小，如果实际消耗大于期望消耗则消耗过快，降速，否则消耗过慢则减速。

详细原理如下

将一天划分成T个时间窗口，对于某一个时间窗口t:

$s_{i,t}$ 表示到t窗口之前实际的累积消耗

$a_{i,t}$ 表示到t窗口之前期望的累积消耗

通过率PTR（pass through rate）计算如下：

$$p_{i,t} = \begin{cases} p_{i,t-1} * (1 + r_t) , if\ s_{i,t} \leq a_{i,t} \\ p_{i,t-1} * (1 - r_t), if\ s_{i,t} > a_{i,t} \end{cases}$$

一些details

- 更新频率，一分钟更新一次PTR表，另外会在7秒钟随机12%的campaign更新一次，上面算法是需要进行快更新。
- 初始的PTR设置成0.1
- fast finish，尽可能使得计划在22点花完所有预算，防止花不完

Real time bid optimization with smooth budget delivery in online advertising

本篇论文主要是2013年Kuang-Chih Lee等人发表的一篇预算平滑的论文，论文特点在于，不仅考虑traffic的平滑消耗，而且会考虑线上performance，论文主要有两个组成部分，一是怎么去计算pacing rate，二是线上如何去使用这个pacing rate。

pacing rate计算

通过某一个campaign我们可以找到以下的规律

$$s(t)=\sum_{j\in I_t}c_jx_j\propto imps(t)\propto reqs(t)\frac{bids(t)}{reqs(t)}\frac{imps(t)}{bids(t)}\propto reqs(t)\cdot pacing\_rate(t)\cdot win\_rate(t)$$

$s(t)$ 是t时间片的实际消耗，$reqs(t)$ 是满足计划受众定向的请求数，$bids(t)$ 是campaign的竞价数，$imps(t)$ 是计划的展现数。由上面可以推导出pacing_rate(t+1)如下

$$pacing\_rate(t+1)=pacing\_rate(t)\cdot\frac{s(t+1)}{s(t)}\cdot\frac{reqs(t)}{reqs(t+1)}\cdot\frac{win\_rate(t)}{win\_rate(t+1)}$$

其中$reqs(t+1)$和$win_rate(t+1)$可以通过一些历史数据去统计获取得，$s(t+1)$是下一个时间片slot的花费，但是在广告投放过程中可能会多或者少于我们期望，这时就需要去把多或少的部分折算到后面，这时就需要重新去调整下一时间slot的花费如下

$$b_{t+1}^u=(B-\sum_{m=1}^{t}s(m))\frac{L(t+1)}{\sum_{m=t+1}^{T}L(m)}$$

如果希望后面是均匀投放，则会$L(m)$就是后面余下的时间片长度，即

$$b_{t+1}^u=(B-\sum_{m=1}^{t}s(m))\frac{1}{T-t}$$

如果希望后面按照时间片内ctr去分配预算，则可以统计每一个campaign的每一个时间片内的历史ctr，按照这个历史ctr来投放，冷启动可以按照均匀投放，积累数据再按ctr投放

$$b_{t+1}^p=(B-\sum_{m=1}^{t}s(m))\frac{p_{t+1}\cdot L(t+1)}{\sum_{m=t+1}^{T}p_m\cdot L(m)}$$

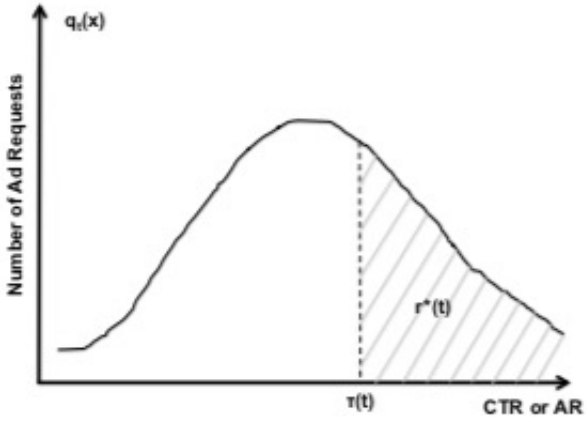
如果时间片长度一样可以简化成如下

$$b_{t+1}^p=(B-\sum_{m=1}^{t}s(m))\frac{p_{t+1}}{\sum_{m=t+1}^{T}p_m}$$

CPM campaign pacing rate线上使用

对于cpm广告，以固定的cpm投放，那么如果知道pacing rate，是可以计算出reqs(t)如下

$$reqs(t) = \frac{bids(t)}{pacing\_rate(t)}$$



现在就需要在时间片t里面找出$reqs(t)$的请求数来，怎么选择是个问题，假设以ctr为目标，就是选择ctr高的部分，不要ctr底的部分，所以就需要计算出一个ctr门槛值，这个threshold可以满足reqs(t)的数量，那现在就需要一个这样的分布$q_x(t)$表示对于ctr为x的reqs数量，那么ctr threshold计算如下

$$\tau(t) = \arg\min_x \left| \int_x^1 q_t(s)ds - \mathbf{reqs}^*(t) \right|$$

但是由于统计数据会存在波动，对于计算的ctr threshold会不准，那个就通过置信区间进控制置信度，如果置信区间的均值和方差为$u_\tau(t)$和$\sigma_\tau(t)$，可以进行增量更新如下：

$$\mu_\tau(t) = \mu_\tau(t-1) + \frac{1}{t}\left(\tau(t) - \mu_\tau(t-1)\right)$$
$$\sigma_\tau^2(t) = \frac{t-1}{t}\sigma_\tau^2(t-1) + \frac{1}{t}\left(\tau(t) - \mu_\tau(t-1)\right)\left(\tau(t) - \mu_\tau(t)\right)$$

如果$\tau(t)$服从高斯分布，那么$\tau(t)$可以表示为$\mu_\tau(t) + \gamma\frac{\sigma_\tau(t)}{\sqrt{d}}$和$\mu_\tau(t) - \gamma\frac{\sigma_\tau(t)}{\sqrt{d}}$ 其中d为统计数据的天数，这个上下界会在每一个时间slot上面更新，最后使用pacing_rate方式如下：

- 如果线上预测的值大于上界，那么直接按照给定的固定的cpm出价
- 如果线上预测的值在上界和下界之间，那么会以pacing rate的概率进行投放
- 如果线上预测的值小于下界，那么直接不出

这样的机制可能确定在每一个时间slot上面，会挑出高ctr的请求出广告，并且能满足预算需求。

非CPM campaign pacing rate线上使用

如果是动态出价的话，比如说是cpc广告或者是ocpc广告的话，就不能向cpm广告那样去卡门槛的方式去做，而是要去修改bid price来调整通过率达到预算限制需求。

以cpa广告，实际现在大部分的广告平台都是ocpc的投放方式了，那个bid price = cvr * target_cpa，现在给定一个pacing_rate，要去如何去调整这个bid price。

首先先定义三个档位$0 \leq \beta_1 < \beta_2 \leq 1$

- Safe region : $pacing\_rate(t) \leq \beta_1$

这时候pacing_rate很低，也就是通过率很底，需要打压bid price减小出价，论文中指出在二价系统下统计一下历史的计价出价比，选计价出价比直方图底部1%或2%的去打压bid price。

- Critical region : $\beta_1 \leq pacing\_rate(t) \leq \beta_2$

直接按照原来的bid price出价

- danger region : $\beta_2 \leq pacing\_rate(t)$

$$\rho^* = 1 + \frac{C/c^* - 1}{1 - \beta_2} \left(\mathbf{pacing\_rate}(t) - \beta_2\right).$$

按照ρ去提升bid price，其中C是指bid price cap，c^*是指历史平均的bid price

Smart Pacing for Effective Online Ad Campaign Optimization

本篇论文使用上文的pacing rate计算方式，但是在使用pacing rate上面做了很多改变，提出了分层的想法，即在线上将ctr分成L层，ctr高的在高优先级的层上，ctr低的在低优先级的层上面，每一层分配的预算不一样，这样预算分配机制可以使得整体系统的ctr得到提升。

具体算法如下：

Algorithm 1 AdjustWithoutPerformanceGoal

Input: $\mathbf{c}^{(t-1)}$, $\mathbf{r}^{(t-1)}$, R
Output: $\mathbf{r}^{(t)}$

1: **if** $R == 0$ **then**
2: return $\mathbf{r}^{(t)} = \mathbf{r}^{(t-1)}$
3: **else if** $R > 0$ **then**
4: **for** each layer l in $(L, \ldots, l^{'})$ **do**
5: $r_l^{(t)} = min(1.0, r_l^{(t-1)} \times \frac{c_l^{(t-1)} + R}{c_l^{(t-1)}})$
6: $R = R - c_l^{(t-1)} \times \frac{r_l^{(t)} - r_l^{(t-1)}}{r_l^{(t-1)}}$
7: **end for**
8: $r_{l^{'}-1}^{(t)} = trial\ rate$ **if** $l^{'} \neq 1$ **and** $r_{l^{'}}^{(t)} > trial\ rate$
9: **else**
10: **for** each layer l in $(l^{'}, \ldots, L)$ **do**
11: $r_l^{(t)} = max(0.0, r_l^{(t-1)} \times \frac{c_l^{(t-1)} + R}{c_l^{(t-1)}})$
12: $R = R - c_l^{(t-1)} \times \frac{r_l^{(t)} - r_l^{(t-1)}}{r_l^{(t-1)}}$
13: **if** $R \geq 0$ **then**
14: $r_{l-1}^{(t)} = trial\ rate$ **if** $l \neq 1$ **and** $r_l^{(t)} > trial\ rate$
15: **break**
16: **end if**
17: **end for**
18: **end if**
19: **return** $\mathbf{r}^{(t)} = (r_1^{(t)}, \ldots, r_L^{(t)})$

这里 c^{t-1}, r^{t-1} 是计划在t-1的time slot上面的消费和pacing rate，而R是指residual，代表是期望消费和实际消费之差，期望消费的计算方式和上一篇论文一样

$$\widehat{C}^{(t)} = B^{(t)} + \frac{B_m - \sum_{t=m+1}^{K} B^{(t)}}{K - m}$$

$$R = \widehat{C}^{(t)} - C^{(t-1)}$$

B^t 为t时间预算，加号后面部分是指到上一次实际消费后，误差会均匀的分配到后面的期望消费上面，所以这里R就是实际产生的消费和我预期的差距，所以这里有三种情况：

- 如果R大于0则表示实际消费少了，需要进行加速消费，即pacing rate要调大，算法第5行就是在调整下一个时间 slot的pacing rate（调整方式和上一篇一样，只是这里只假设了消费和pacing rate成正比的情况），可以看到L是最高层，也就是pacing rate最大的层次，意思是如果有额外的消费会优先给ctr高的流量层，pacing rate计算很简单了，现在有R的预算要分配，那个会先填满第一层，pacing rate最高也只有1.0,所以有可能填不满，算法第6行会计算一下还剩多少，继续向下层进行分配。如果到了最后一层，但是已经分配完成，所以最后一层是不是应该为0呢，但是如果为0的话下一次就没有办法再去调整了，论文中设置了一个trial rate，即试验概率设放，这个trial rate会很小，论文中给出的经验值是0.001
- 如果R小于0则表示实际消费多了，那么需要进行降速，即pacing rate减小，和pacing rate增加不

同的是，他是从优先级低的层开始调整，然后当R>=0的时候结束，并将当前层的pacing rate设置成trial rate，所以这里面低优先级的层有可能小于trail rate，觉得过小会导致的问题是没有流量，下一个桶的消费过少而不置信，但是论文没有给于解释。

- 如果R等于0的话就不调整

上面介绍的是without performance的方法，下面介绍如果有performance的要求该怎么去调整，论文是给的performance goal是以ecpc为例，假设给定上一个时间slot各层的ecpc和pacing rate，如果去估算下一层的ecpc值

$$\mathrm{ExpPerf}(\mathbf{c}^{(t-1)}, \mathbf{r}^{(t-1)}, \mathbf{r}^{(t)}, \mathbf{e}, i) = \frac{\sum_{j=i}^{L} \frac{c_j^{(t-1)} \times r_j^{(t)}}{r_j^{(t-1)}}}{\sum_{j=i}^{L} \frac{c_j^{(t-1)} \times r_j^{(t)}}{r_j^{(t-1)} \times e_j}}$$

首先得知道ecpc的计算方式，实际的ecpc计算是实际消费cost/实际点击数click，那么上式的分子

$$\sum_{j=i}^{L} \frac{c_j^{(t-1)} \times r_j^{(t)}}{r_j^{(t-1)}}$$

累加各层的预估消费，这里假设是pacing rate和消费正相关，分母多了一个e_j 表示预估累加各层的预估点击数，所以整体计算下面就是时间t的ecpc了，那么广告主希望的是ecpc要够小，即目标为goal，那么

Algorithm 2 AdjustWithPerformanceGoal

Input: $\mathbf{c}^{(t-1)}$, $\mathbf{r}^{(t-1)}$, R, $\mathbf{e}$, $goal$
Output: $\mathbf{r}^{(t)}$

1: $\mathbf{r}^{(t)}$=AdjustWithoutPerformanceGoal($\mathbf{c}^{(t-1)}$, $\mathbf{r}^{(t-1)}$, R)
2: **if** ExpPerf($\mathbf{c}^{(t-1)}, \mathbf{r}^{(t-1)}, \mathbf{r}^{(t)}, \mathbf{e}, 1) > goal$ **then**
3: **for** each layer l in $(1, \ldots, L)$ **do**
4: **if** ExpPerf($\mathbf{c}^{(t-1)}, \mathbf{r}^{(t-1)}, \mathbf{r}^{(t)}, \mathbf{e}, l+1) > goal$ **then**
5: $r_l^{(t)} = 0.0$
6: **else**
7: $r_l^{(t)} = r_l^{(t-1)} \times \frac{\sum_{i=l+1,\ldots,L} c_i^{(t-1)} \times (\frac{goal}{e_i} - 1)}{c_l^{(t-1)} \times (1 - \frac{goal}{e_l})}$
8: **if** $l \neq 1$ **then**
9: $r_{l-1}^{(t)} = trial\ rate$
10: **end if**
11: **break**
12: **end if**
13: **end for**
14: **end if**
15: **return** $\mathbf{r}^{(t)} = (r_1^{(t)}, \ldots, r_L^{(t)})$

算法整体流程：

> 第一行：按照无performance的方式去计算time slot t的各层的pacing rate
>
> 第二行：计算一下time slot t的expected performance即ecpc，如果没有达到目标goal进行第四行调整算法
>
> 第三行：从第一层到最后一层进行调整
>
> 第四行：先再检查一下ecpc，
>
> 第五行：如果没有达到goal，从低层开始直接将pacing rate设置成0
>
> 第七行：如果遇到小于goal的层，直接将之前pacing rate设置成0的层的消费匀到这一层来。如果l不是第一层，那个t-1这一层的pacing rate直接设置成trial rate去试量，以便后面有调整的余地。

解释一下算法第7行做的事情：

$1 - goal/e_l$是指l层离目标差多少，因为l+1层ecpc小于goal，并且l层肯定是大于goal，那么比l+1层高的层的ecpc肯定是小于goal，因为高层的ctr高，所以同样的消费点击数应该多，即ecpc低，论文最终的目标是要让整体的ecpc达到goal，所以就要把高层高于goal的ecpc增量分趟到l层，因为l层是最后一个低于goal，所以第七行的分子是让计算多出多少比例，因为消费和pacing rate成正比，所以会通过消费比去计算调整幅度，最终达到整体ecpc稳定在goal值。