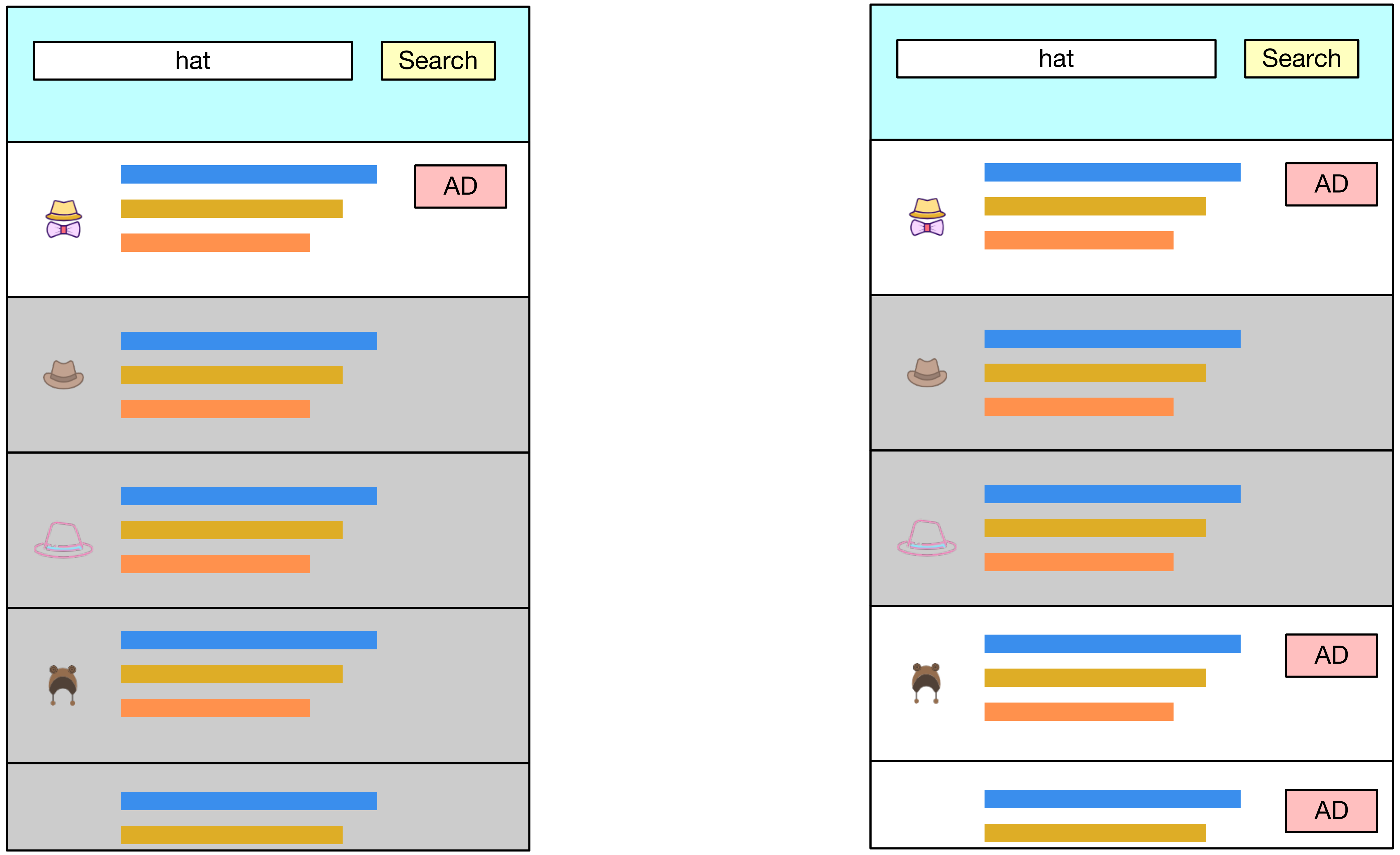
ANT GROUP

Multi-Slots Online Matching With High Entropy

Xingyu Lu   Qintong Wu   Wenliang Zhong

{sing.lxy, qintong.wqt, yice.zwl}@antgroup.com

Multi-slots Online Matching Applications



- Multiple Ads are presented under resource constraints
- User pays the most attention onto particular ads slots
- Diversity shall be maintained across different slots

$$\max_{\mathbf{X}\in\mathscr{X}} \sum_{t=1}^{T} \mathbf{r}_t^\top \mathbf{X}_t \mathbf{c} + \alpha \mathscr{H}(\mathbf{X}_t \mathbf{c})$$

$$\sum_{t=1}^{T} \mathbf{M}_t^\top \mathbf{X}_t \mathbf{c} \le T\mathbf{B}$$

Here, ***c*** characterizes slots' impression capacity. Correspondingly, ***c*** influences both the objective as well as the consumption of resources.

$\mathscr{H}(x)$ refers to the entropy regularizer, which is designed to promote diversity.

Problem Formulation

Original Problem

$$\max_{\mathbf{X}\in\mathcal{X}} \sum_{t=1}^{T} \mathbf{r}_t^\top \mathbf{X}_t \mathbf{c} + \alpha \mathcal{H}(\mathbf{X}_t \mathbf{c})$$

$$\sum_{t=1}^{T} \mathbf{M}_t^\top \mathbf{X}_t \mathbf{c} \le T\mathbf{B}$$

Directly solving $\mathbf{X}_t$ requires $\mathcal{O}(N^3 A_t^3)$ complexity!

Unacceptable for real-world applications in general

Our Two-Steps Approach

Introduce the intermediate variable $\mathbf{y_t} := \mathbf{X_t c}$ representing the total expected impressions in all slots.

$$\max_{\mathbf{y}\in\mathcal{Y}} \sum_{t=1}^{T} \mathbf{r}_t^\top \mathbf{y}_t + \alpha \mathcal{H}(\mathbf{y}_t)$$

$$s.t. \sum_{t=1}^{T} \mathbf{M}_t^\top \mathbf{y}_t \le T\mathbf{B}$$

First, we optimize a simpler problem by reducing the number of variable.

$$\hat{\mathbf{X}}_t = \arg_{\mathbf{X}_t\in\mathcal{X}} \{\mathbf{X}_t \mathbf{c} = \hat{\mathbf{y}}_t\}$$

Next, we solve a linear system to recover the decision matrix.

Online subGradient descent for Multi-slots Allocation (OG-MA)

Algorithm 1 Online subGradient descent for Multi-slots Allocation (OG-MA)

Input: User set $\mathbb{T}$, the step-size η; and initialize dual variables $\boldsymbol{\lambda}_0 = \mathbf{0}$.

for $t = 1$ **to** T **do**

Receive a stochastic request with $(\mathbf{r}_t, \mathbf{M}_t)$.

Solve the expected impressions $\hat{\mathbf{y}}_t$ for all advertisements using **efficiency pooling projection**;

Update the allocated impressions under the remaining resources:

$$\widetilde{\mathbf{y}}_t = \begin{cases} \hat{\mathbf{y}}_t, & \text{if} \sum_{s=1}^{t-1} \mathbf{M}_s^\top \widetilde{\mathbf{X}}_s \mathbf{c} + \mathbf{M}_t^\top \hat{\mathbf{y}}_t \le T\mathbf{B}, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{11}$$

Make the realization $\widetilde{\mathbf{X}}_t$ of primal solution $\mathbf{X}_t$ with given $\widetilde{\mathbf{y}}_t$ by **roulette swapping allocation**.

Compute gradients $\mathbf{g}(\boldsymbol{\lambda}_t)$ of $\boldsymbol{\lambda}_t$ where:

$$\mathbf{g}(\boldsymbol{\lambda}_t) := \mathbf{B} - \mathbf{M}_t^\top \hat{\mathbf{y}}_t.$$

Update $\boldsymbol{\lambda}$ by **projected subgradient descent**:

$$\boldsymbol{\lambda}_{t+1} = \text{Proj}_{\boldsymbol{\lambda} \ge 0}\{\boldsymbol{\lambda}_t - \eta \mathbf{g}(\boldsymbol{\lambda}_t)\} \tag{12}$$

end for

- Efficiency Pooling Projection estimates $\hat{\mathbf{y}}_t$
- Roulette Swapping Allocation samples $\tilde{\mathbf{X}}_t$
- Projected subGradient Descent updates λ_t

Results:

OG-MA achieves $\mathcal{O}(N + NA_t + A_t log A_t)$ complexity

Recall that a vanilla method takes $\mathcal{O}(N^3 A_t^3)$

OG-MA attains $\mathcal{O}(C(\sqrt{K} + logT)\sqrt{T})$ regret

Choose the position-based click model[1] $c_n = \dfrac{1}{n^\gamma}$

- $\gamma = 1$, the regret is of order $\mathcal{O}(logN)$
- $\gamma = \dfrac{1}{2}$, the regret is of order $\mathcal{O}(\sqrt{N})$

[1]: Craswell, N., Zoeter, O., Taylor, M., and Ramsey, B. An experimental comparison of click position-bias models.

Efficiency Pooling Projection Algorithm (EPP)

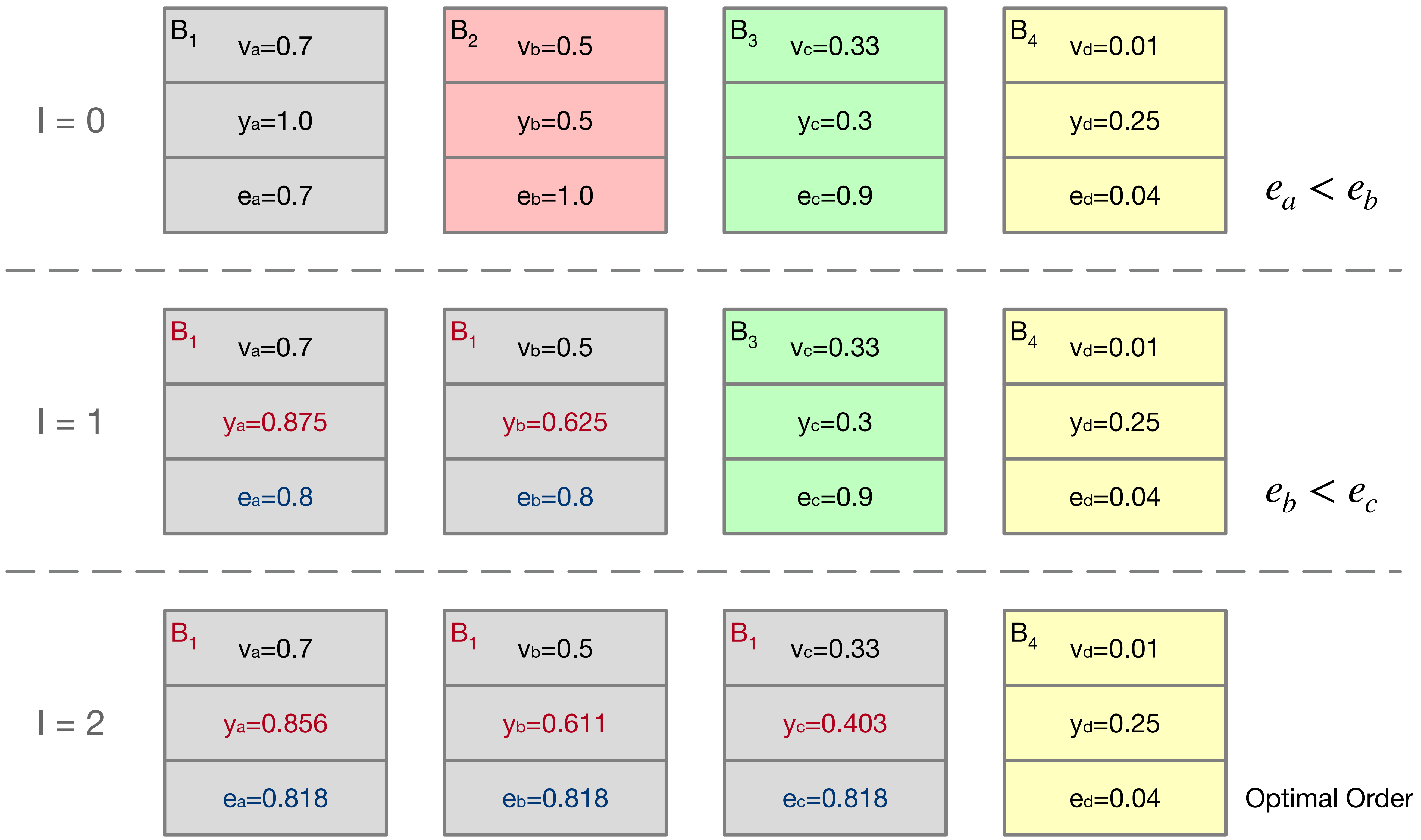
Algorithm 2 Efficiency Pooling Projection (EPP)

Input: User request $(\mathbf{r}_t, \mathbf{M}_t)$, dual variable $\boldsymbol{\lambda}$.
Sort $\mathbf{E}_t$ in decreasing order by $v_{t,a}$.
Initialize $\mathbf{E}_t$ into blocks $\{\mathbb{B}_r^{(0)}\}_{r=1}^{N+1}$ by (13), compute efficiency value $E(\mathbb{B}_r^0)$ by (14) and set $l = 0$.
repeat
  Step1. Merge $\mathbb{B}^{(l)}$-blocks if $E(\mathbb{B}_r^{(l)}) \leq E(\mathbb{B}_{r+1}^{(l)})$.
  Step2. Update the merged blocks $\mathbb{B}_r^{(l+1)} := \mathbb{B}_r^{(l)}$ and efficiency value $E(\mathbb{B}_r^{(l+1)})$ for all r, i.e..
  Step3. If exists $E(\mathbb{B}_r^{(l+1)}) \leq E(\mathbb{B}_{r+1}^{(l+1)})$, then increase $l = l + 1$ and go back to Step1.
until $E(\mathbb{B}_r^{(l)}) > E(\mathbb{B}_{r+1}^{(l)})$ for all block r.
Output: $\hat{y}_{t,a} = v_{t,a}/E(\mathbb{B}_r), \forall a \in \mathbb{B}_r$ and block index r.

- Define $v_{t,a}$ as the contribution value to the objective, and let $e_{t,a} := v_{t,a}/y_{t,a}$ be the efficiency value for primal solution $y_{t,a}$
- The optimal solution $\mathbf{y}_t^*$ and its efficiency $\mathbf{e}_t^*$ are in the same order

Key idea:

- Follow the idea of Pool Adjacent Violators Algorithm[2] (PAVA)
- Iteratively enforce the efficiency $e_{t,a}$ in non-increasing order by merging adjacent ads and sharing the same efficiency
- Update the expected impression $y_{t,a}$ after merging operations



[2]:De Leeuw, J., Hornik, K., and Mair, P. Isotone optimization in r: pool-adjacent-violators algorithm (pava) and active set methods.

Roulette Swapping Allocation

Algorithm 3 Roulette Swapping Allocation (RSA)

Input: Expected impressions $\widetilde{\mathbf{y}}_t$ computed by (11).
Initialize position order $r(a) = a$, the expectation of allocated impressions $y_{t,a} = c_a \mathbb{I}(1 \le a \le N), \forall a \in \mathbf{E}_t$ and index set $\mathbb{S} = \{\}$.
for $j = 1$ **to** A_t **do**
  if $y_{t,j} > \widetilde{y}_{t,j}$ **then**
    Put j into the index set: $\mathbb{S} = \mathbb{S} \cup \{j\}$
  else
    Swap $r(j)$ and $r(s), s \in \mathbb{S}$ with probability:

$$p_s = \frac{\widetilde{y}_{t,j} - y_{t,j}}{y_{t,s} - y_{t,j}} \frac{\widetilde{y}_{t,s} - y_{t,s}}{\sum_{s' \in \mathbb{S}} (\widetilde{y}_{t,s'} - y_{t,s'})}. \quad (15)$$

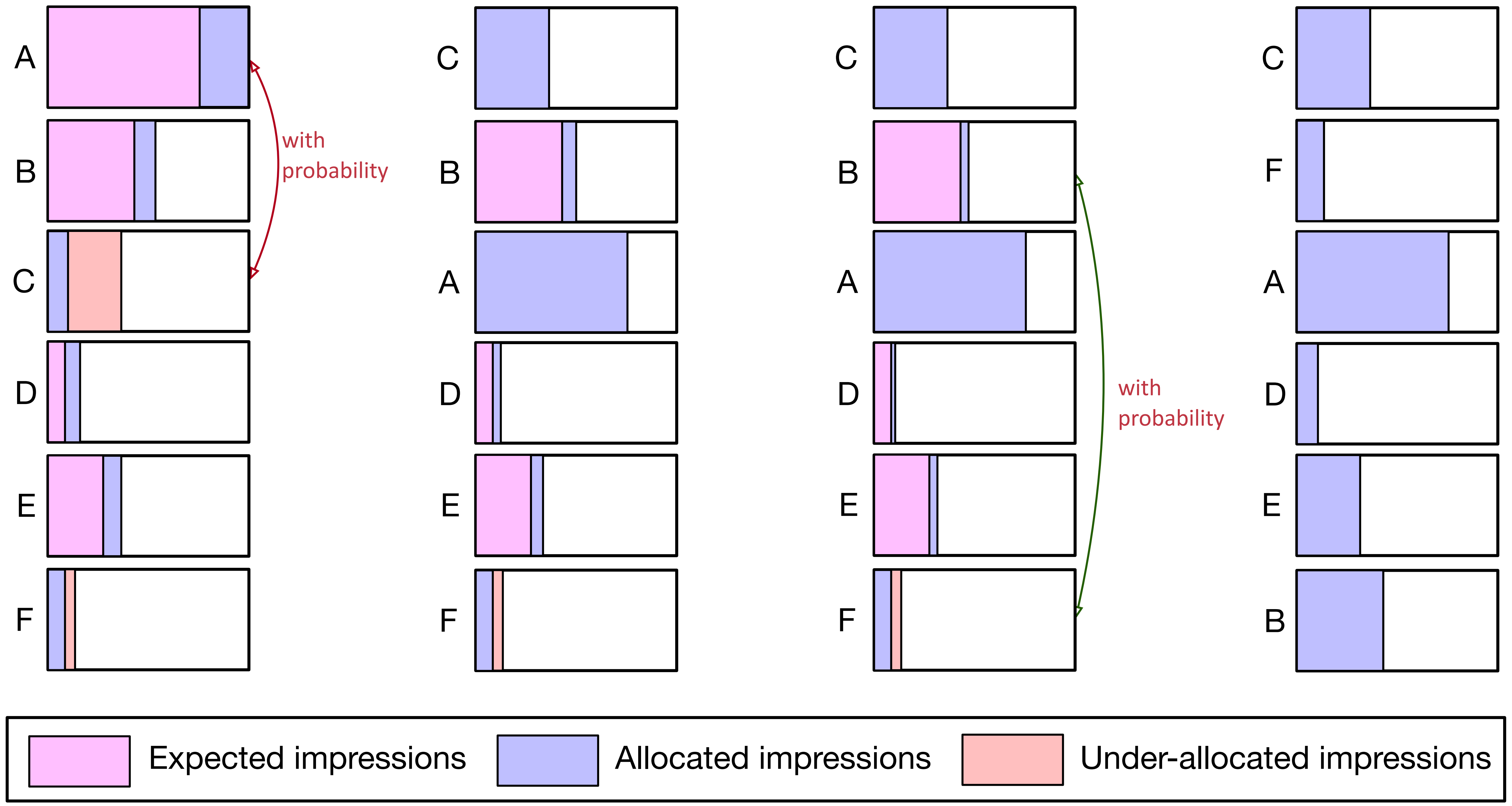
    Update the allocated impressions of j by $y_{t,j} = \widetilde{y}_{t,j}$
    Update the allocated impressions of $s \in \mathbb{S}$ by:

$$y_{t,s} = (1 - p_s) y_{t,s} + p_s y_{t,j},$$

    and then remove s from $\mathbb{S}$ if $y_{t,s} = \widetilde{y}_{t,s}$.
  end if
end for
Output: Allocate $a \in \mathbf{E}_t$ to $r(a)$-th slot if $r(a) \le N$.

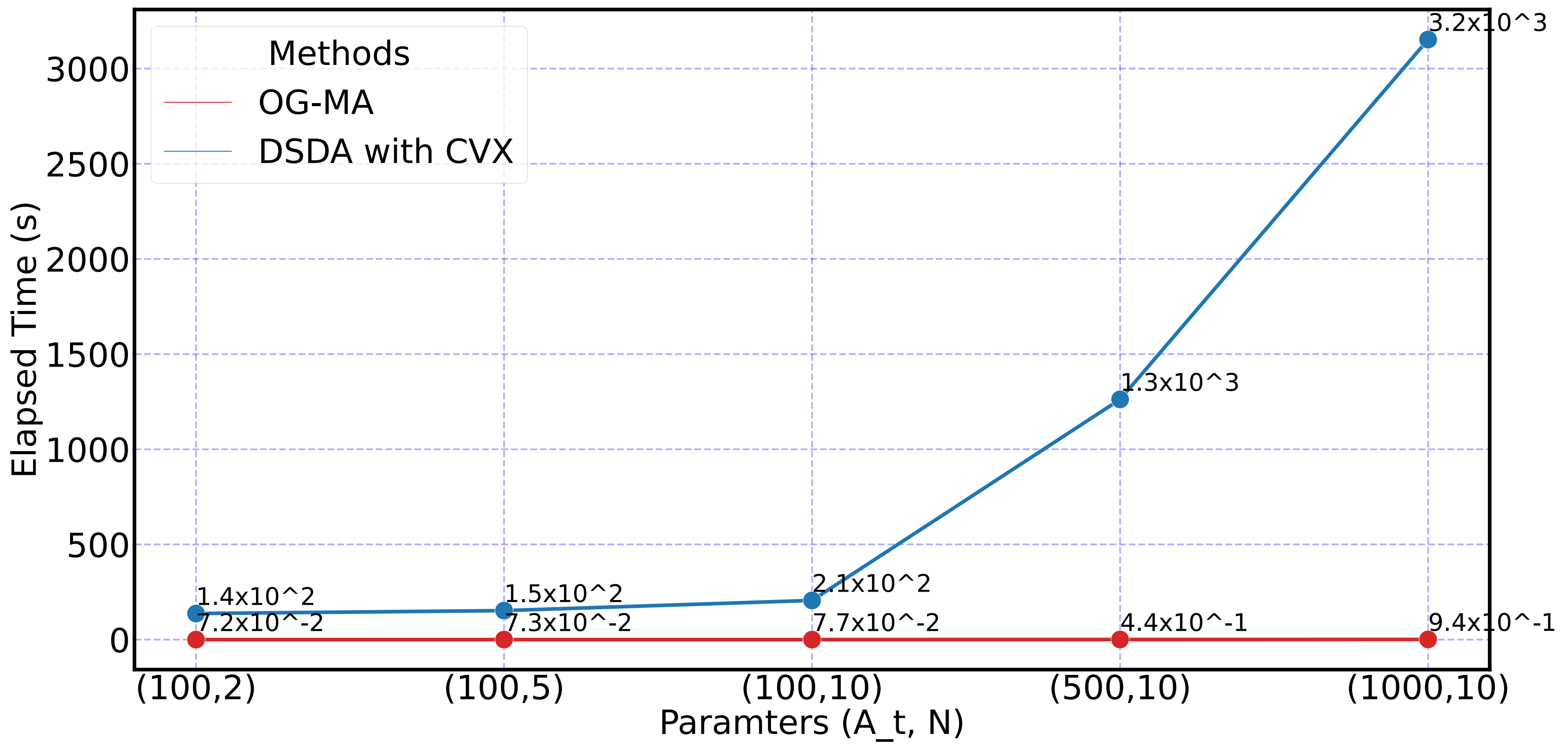
Key idea:

- Allocate the expected impressions by swapping the positions of advertisements
- Swapping operations utilize excess impressions to make up for under-allocated advertisements



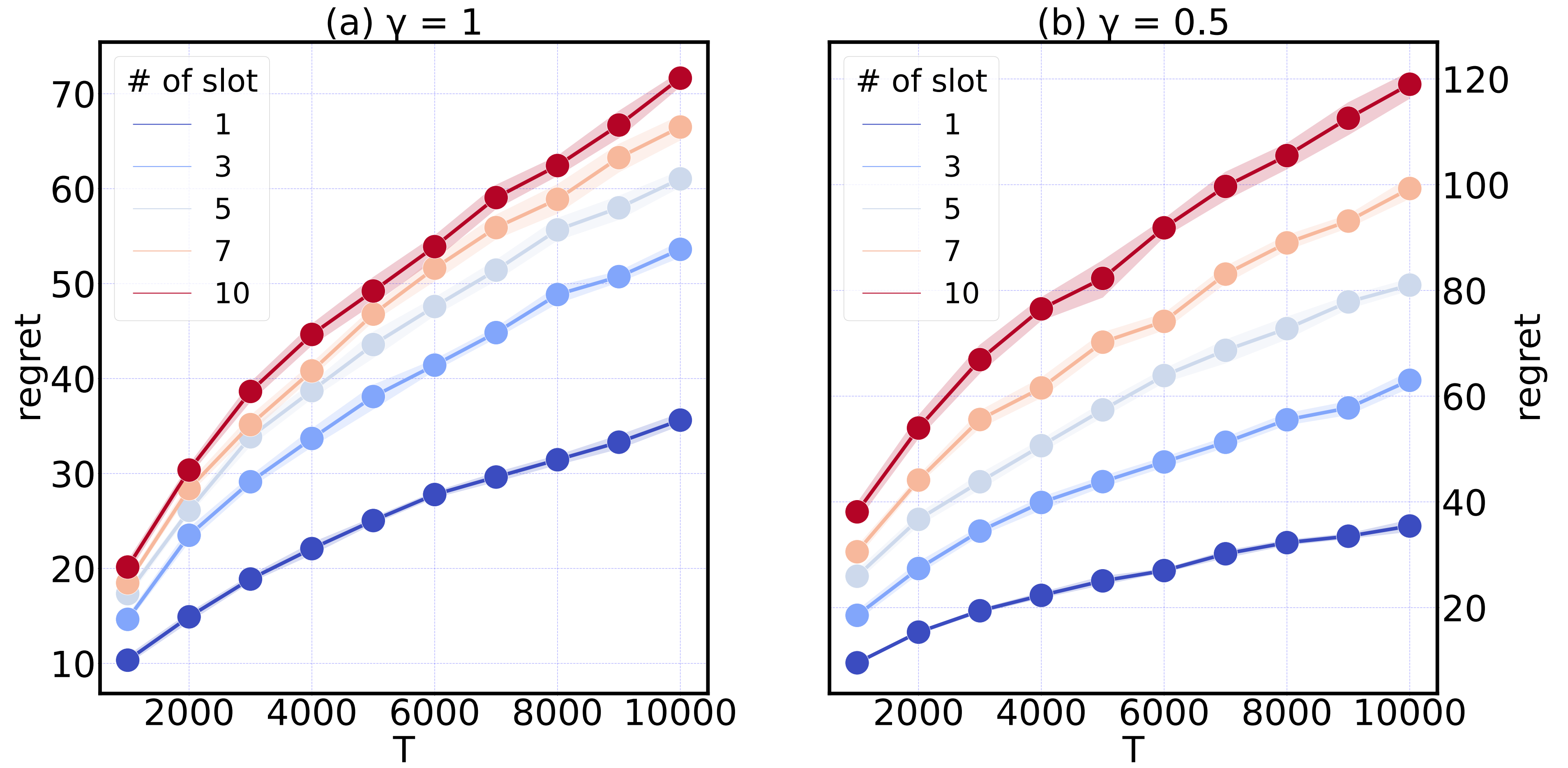
Experiments

Effectiveness on Reducing Computation



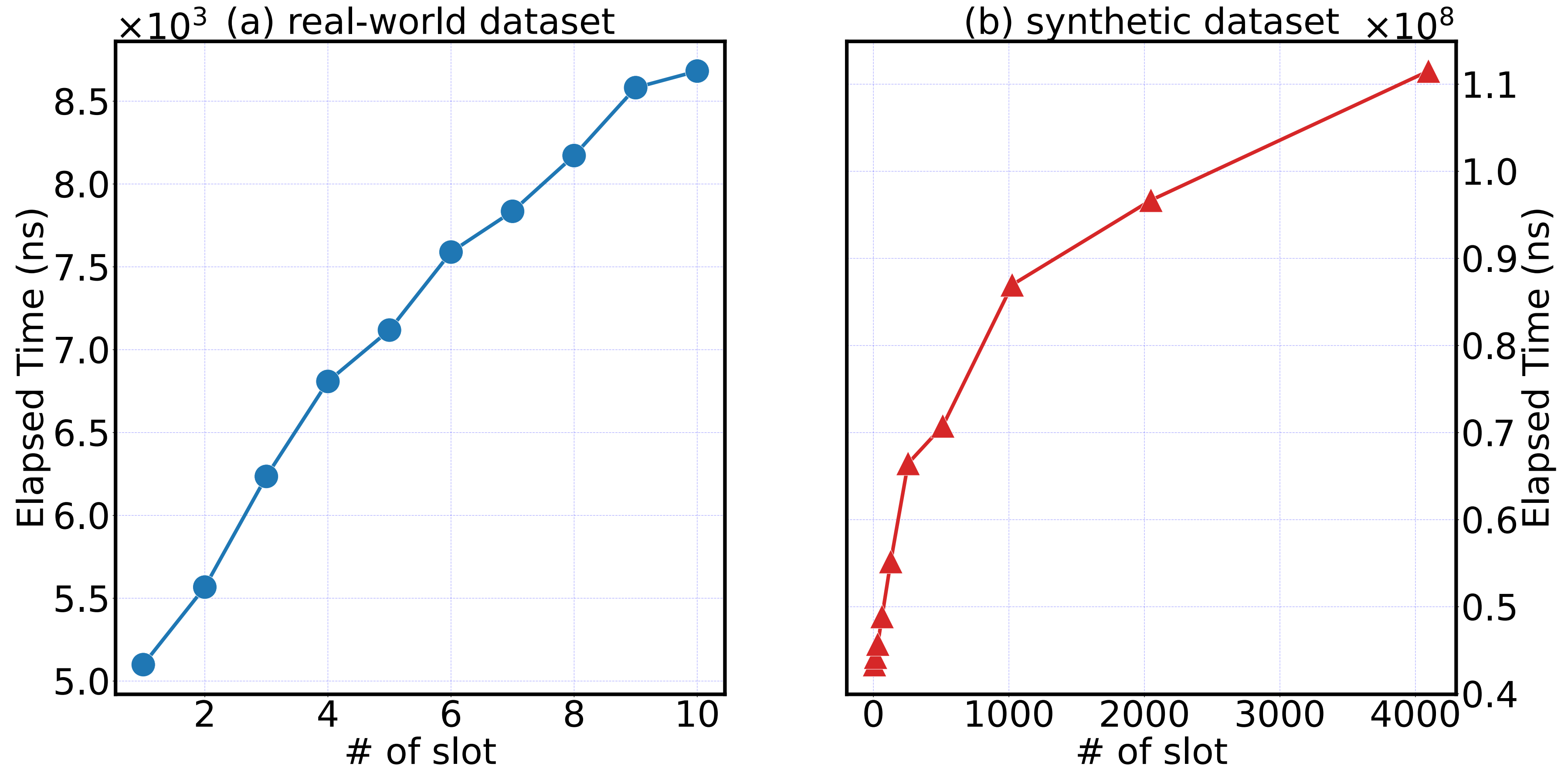
The OG-MA is 3 ~ 4 order faster than dual subgradient descent.

Regret Bound for different Click Model



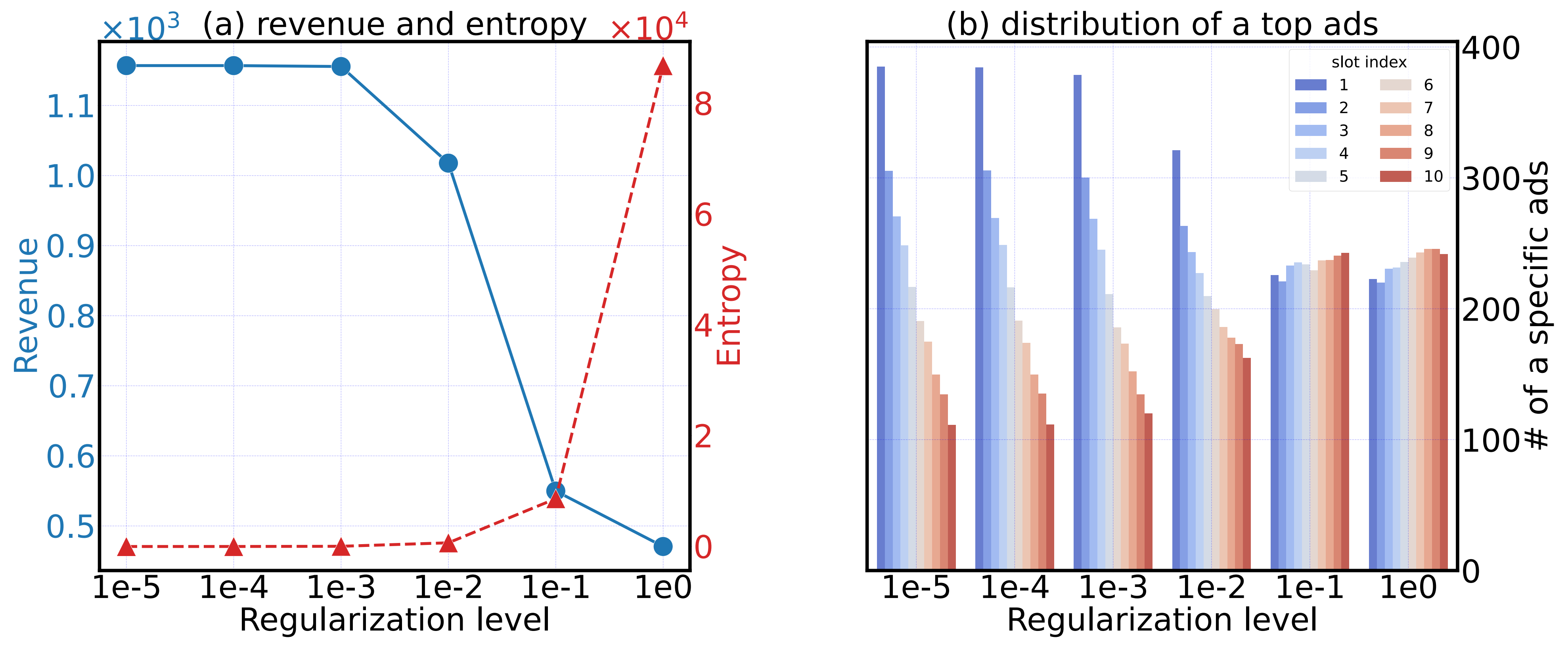
The experiment results coincide with the theoretical analysis.

Inference Efficiency



The complexity grows sub-linearly w.r.t the number of slots *N*.

Trade-off between Revenue and Diversity



Higher entropy leads to better diversity in matching.
Particularly, $\alpha = 0.01$ presents a good trade-off result.

Conclusion

Online subGradient descent for Multi-slots Allocation (OG-MA)

- **Scalable**: $\mathcal{O}(N + NA_t + A_t log A_t)$ complexity, good for large-scale applications
- **Effective**: sub-linear regret w.r.t. the number of slots
- **Diverse**: provides diversified ranking results without violating resource constraints
- **Easy implementation**: only consists of basic operations (i.e., swap and merge)