Comparative Analysis and Enhancement of Online and Offline Optimization Algorithms for Ad Allocation

Arav Chadha[1,4], Maggie Zhang[1,4], Lindsay Kossoff[2,4], Isabella De La Garza[3,4], Alina Ene[4]

Canyon Crest Academy, 5951 Village Center Loop Rd, San Diego, CA 92130[1]; Holton-Arms School, 7303 River Rd, Bethesda, MD 20817[2]; United High School, 2811 United Ave, Laredo, TX 78045[3]; Boston University, Commonwealth Ave, Boston, MA 02215[4]

INTRODUCTION

- In an increasingly commercial world, advertising is essential for promoting products and services to targeted audiences.
- The generalized assignment problem (GAP) is a well researched problem in this domain.
- The goal is to find the most profitable allocation of ad impressions to budget-constrained advertisers, given that advertisers value, or weigh, impressions differently based on user data.

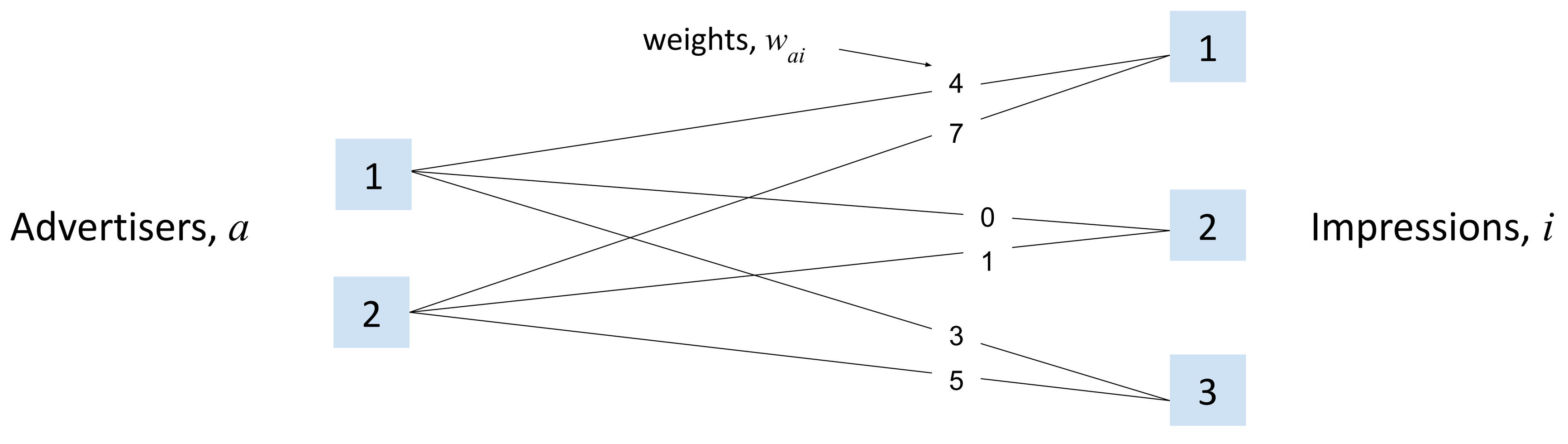


Fig 1. bipartite graph with advertisers, impressions, and weights

- We implement and evaluate an online and offline ad allocation algorithm:
  - Online - impressions arrive in real-time and weights calculated from user data
  - Offline - impressions, user information, and therefore weights known beforehand
- Our objective is to enhance both algorithms in terms of performance and efficiency and compare their effectiveness under various conditions.
- We analyze both algorithms using synthetic instances, corrupted instances, and real-world data from the Stanford Large Network Dataset Collection [4].
- We evaluate allocation thresholds, fine-tune parameters, and implement predictions.

METHODS

Synthetic Instances:

- We generate impressions with random types and advertisers with random budgets based on the instance size. Advertisers' valuations for different impression types are sampled from an exponential distribution.

Algorithm 1 (Online) by Spaeh and Ene (2023) [1]:

- Input: bipartite graph of a, i, w, parameter $\alpha \in [1, \infty)$, advertiser budgets $B_a \in \mathbb{N}$.

Algorithm 2 (Offline) by Agrawal et al. (2018) [2]:

- Input: bipartite graph of a, i, r (weights), advertiser budgets $C_a \in \mathbb{N}$, parameter $\lambda \in (0, \infty)$, parameter $\varepsilon \in (0, 1)$, parameter R (number of rounds).

CVXOPT [3]:

- Linear program solver used to find optimal allocation for ad allocation graphs.

Objective Value:

- Sum of all allocated weights (total profit)—metric used to evaluate solution strength.

Testing:

- Corrupted Instances: ad allocation graphs with edges randomly removed or weights randomly scaled by large factors (type of synthetic instance).
- Real-world Data: data used and cleaned from Stanford Large Network Dataset Collection.
- Fine-tuning Parameters: we create an objective value heatmap to identify the strongest pairing of λ and ε for Algorithm 2 on synthetic instances of 50 advs and 1000 imps.
- Allocation Thresholds: we compare three methods of updating allocation thresholds for Algorithm 1: lowest weight, uniform weight average, exponential weight average.
- Comparison: we compare algorithms by obj value (performance) and time taken (efficiency)

Algorithm 1:

$B := \min_a B_{a'}$, $e_B := (1 + 1/B)^B$, and $\alpha_B := B\,(e_B^{\alpha/B} - 1)$

- For each adv initialize $\beta_a \leftarrow 0$
  - For each imp:
    - Find expected adv via $\mathrm{argmax}_a\{w_{ai} - \beta_a\}$ and find predicted adv using prediction method
    - If $\alpha_B(w_{a(\mathrm{PRD})i} - \beta_{a(\mathrm{PRD})}) \geq (w_{a(\mathrm{EXP})i} - \beta_{a(\mathrm{EXP})})$, select exp adv; else, select pred adv
    - Allocate imp to adv and if adv over budget, remove adv' least valuable impression
  - Update $\beta_a \leftarrow \frac{e_{B_a}^{\alpha/B_a} - 1}{e_{B_a}^{\alpha} - 1} \sum_{i=1}^{B_a} w_i e_{B_a}^{\alpha(B_a - i)/B_a}$ for adv

Algorithm 2:

- Assign priority score $\beta_a = (1 + \epsilon)^{-R}$ for all advs
- For R rounds:
  - For each imp: set allocation $\mathbf{x}_{i,a} = \begin{cases} \boldsymbol{\beta}_a \boldsymbol{D}_{i,a,\lambda} & \text{if } \sum_{a' \in \mathbf{N}_i} \boldsymbol{\beta}_{a'} \boldsymbol{D}_{i,a',\lambda} \leq 1 \\ \frac{\boldsymbol{\beta}_a \boldsymbol{D}_{i,a,\lambda}}{\sum_{a' \in \mathbf{N}_i} \boldsymbol{\beta}_{a'} \boldsymbol{D}_{i,a',\lambda}} & \text{otherwise} \end{cases}$
  - For each adv: update β_a $\quad \mathbf{Alloc}_a \leq \frac{\mathbf{C}_a}{(1+\epsilon)} \implies \boldsymbol{\beta}_a := (1+\epsilon)\boldsymbol{\beta}_a$; $\quad \mathbf{Alloc}_a \geq (1+\epsilon)\mathbf{C}_a \implies \boldsymbol{\beta}_a := \frac{\boldsymbol{\beta}_a}{(1+\epsilon)}$
- For all advs over budget: reduce least valuable imp until budget met

$$\boldsymbol{D}_{i,a,\lambda} = e^{\frac{\mathbf{r}_{i,a}}{\lambda}} - 1$$

$$\mathbf{Alloc}_a := \sum_{i \in \mathbf{N}_a} \mathbf{x}_{i,a}$$

RESULTS

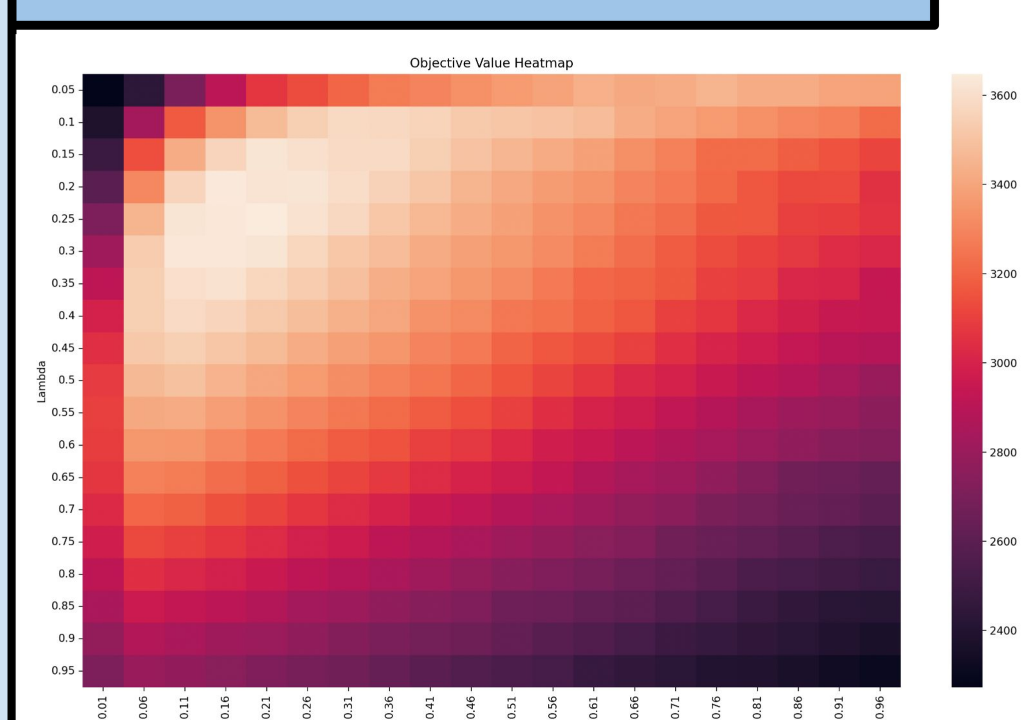


Fig 2. Objective value heatmap for tuning epsilon and lambda on 50 adv, 1000 imp instances. Rounds = 50 for all epsilon and lambda combinations

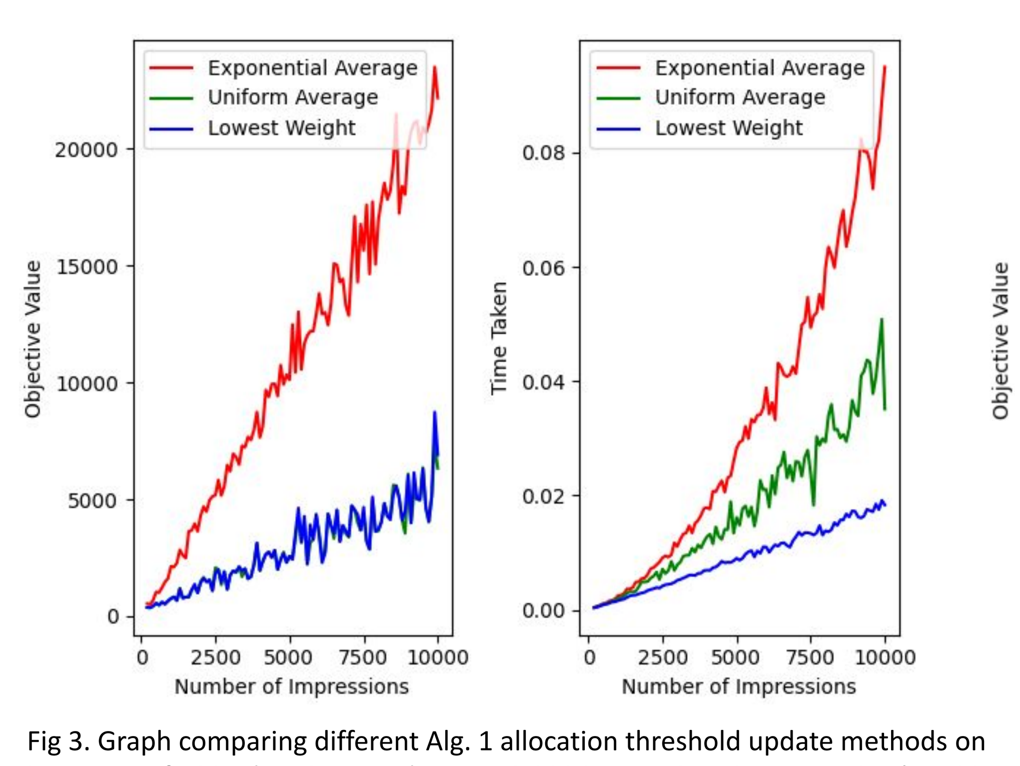


Fig 3. Graph comparing different Alg. 1 allocation threshold update methods on instances of 100 advertisers and up to 10,000 impressions incrementing by 100

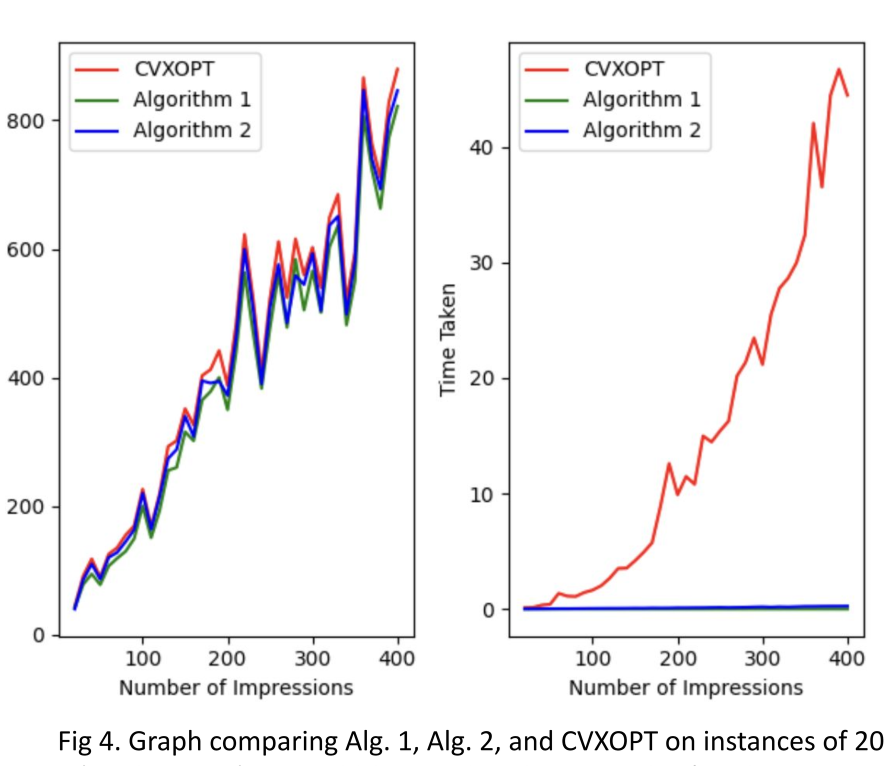
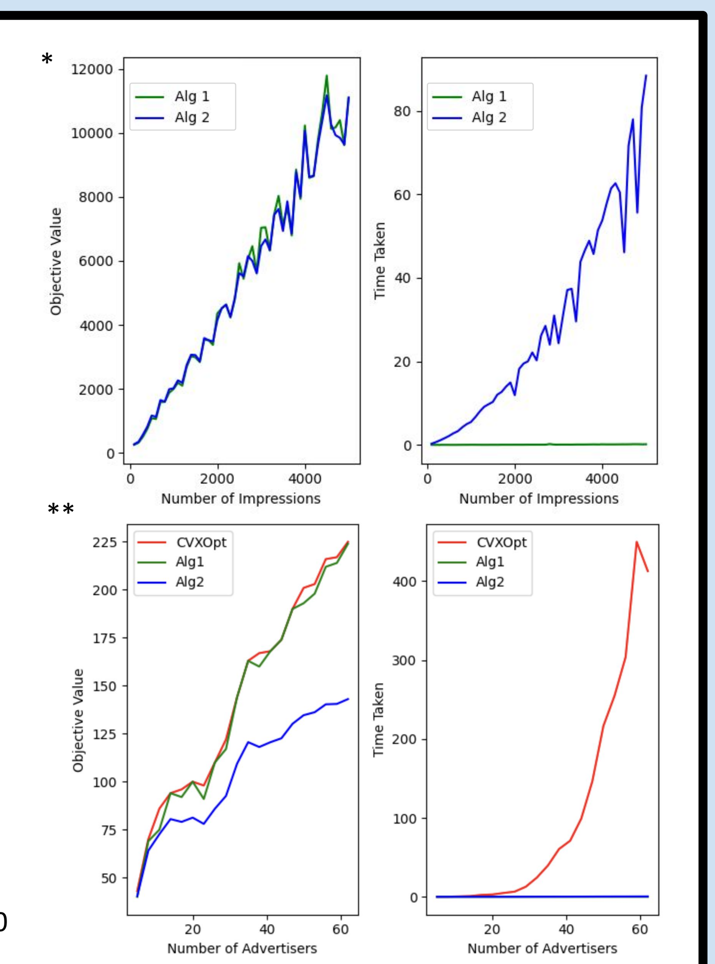


Fig 4. Graph comparing Alg. 1, Alg. 2, and CVXOPT on instances of 20 advertisers and up to 400 impressions incrementing by 10

*Fig 5. Graph comparing Alg. 1 and Alg. 2 on large, corrupted instances of 50 advertisers and up to 5,000 impressions incrementing by 100

**Fig 6. Graph comparing Alg. 1, Alg. 2, and CVXOPT on real-world instances of up to 60 advertisers incrementing by 3 and a predetermined number of impressions



CONCLUSION

Hyperparameter Tuning:

- Looking at Fig. 2, we identify that Alg. 2 performs best on instances of 50 advertisers and 1000 impressions with parameters **λ = 0.25**, **ϵ = 0.21**, **R = 50**.
- We use these values as approximations of ideal parameters for larger instances, since generating a heatmap for larger instances is highly inefficient.

Allocation Thresholds:

- Looking at Fig. 3, an exponential weight average is the best allocation threshold update method for Alg. 1.

Comparison:

- From Fig. 4 and 5, we see that Alg. 2 and Alg. 1 perform similarly on corrupted and synthetic data. However, in Fig 6, Alg. 1 outperforms Alg. 2 on real-world data. This could be due to Alg. 2 returning a fractional allocation.

Future Work:

- Implement machine-learned predictions instead of current mathematical predictions in Alg 1.

REFERENCES

[1] Spaeh, F. and Ene, A. Online ad allocation with predictions. May 26, 2023. https://doi.org/10.48550/arXiv.2302.01827 (accessed August 6, 2024).
[2] Agrawal, S.; Mirrokni, V.; Zadimoghaddam, M. Proportional Allocation: Simple, Distributed, and Diverse Matching with High Entropy. 2018. https://proceedings.mlr.press/v80/agrawal18b/agrawal18b.pdf (accessed August 6, 2024).
[3] Andersen, M.; Dahl, J.; Vandenberghe, L. CVXOPT, version 1.3, 2023.
[4] Leskovec, J. and Krevl, A. Wikipedia adminship election data, 2008. Stanford Large Network Dataset Collection. https://snap.stanford.edu/data/wiki-Elec.html (accessed August 6, 2024).

ACKNOWLEDGEMENTS

I would like to thank Professor Alina Ene for her invaluable guidance and continuous support throughout this project. I would also like to thank the Boston University RISE program and my family for making this experience possible. Lastly, a special thanks to my lab partners for their support and collaboration in this project.