Learnable and Instance-Robust Predictions for Online Matching, Flows, and Load Balancing

Thomas Lavastida, Benjamin Moseley,
R. Ravi, Chenyang Xu

Carnegie Mellon University
Tepper School of Business

ML is Everywhere

- Massive successes in image classification, NLP, etc.
- Widening array of applications
- Can ML help to improve algorithms for combinatorial problems?
  - (Online algorithms, in particular)

Current Status

- Ski Rental
  - [Purohit et al. 2018], [Gollapudi et al. 2019]
- Caching
  - [Lykouris et al. 2018], [Rohatgi 2020]
- Scheduling
  - [Purohit et al. 2018], [Lattanzi et al. 2020]
- Secretaries
  - [Antoniadis et al 2020], [Dütting et al. 2021]
- … and more

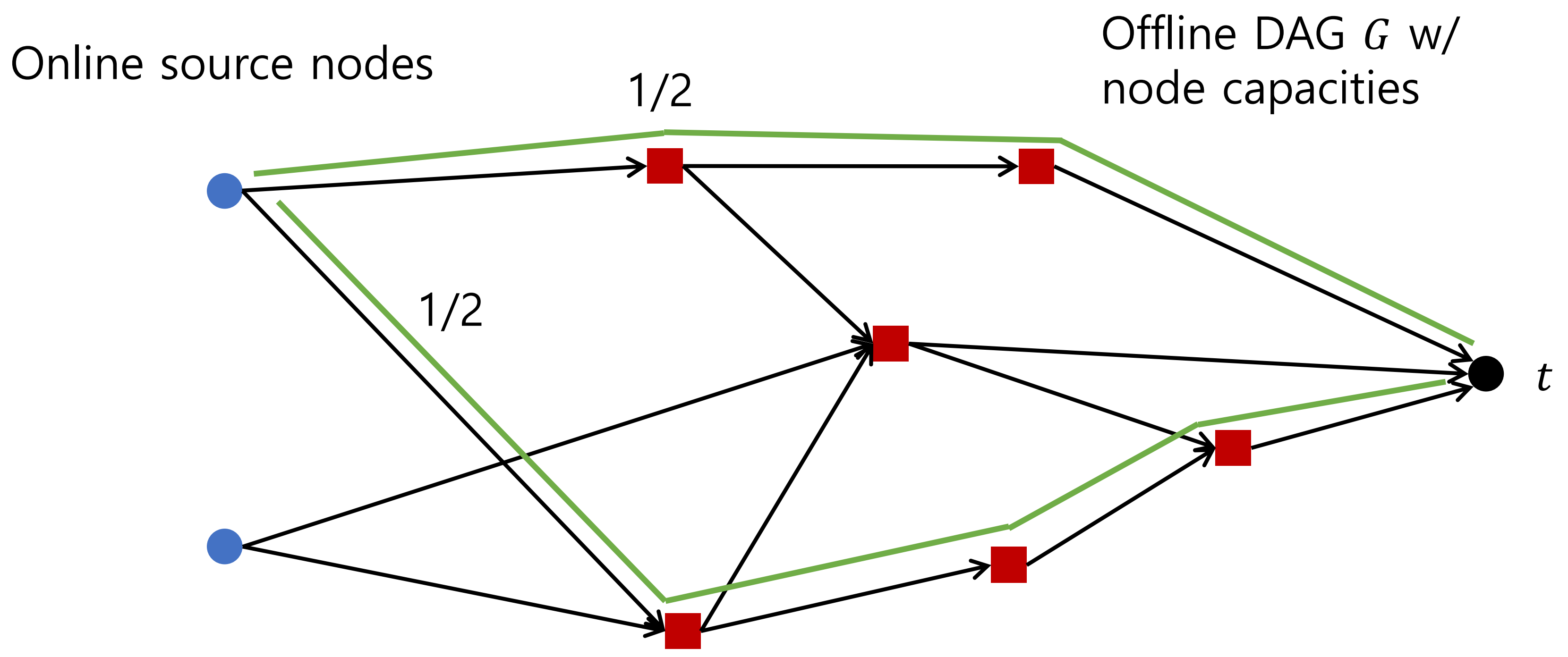
Learning Augmented Algorithms

- Algorithm given access to predictions about online input
- There may be errors – algorithm must account for this
  - "Consistency" – no error case
  - "Robustness" – infinite error case
- Benefits
  - Accurate predictions can go beyond worst case
  - Focuses on algorithm design – abstracts the learning step

Contributions

- Develop a new model for online algorithms with predictions
- Instance robustness
  - New way to understand "error"
- Learnability
  - What types of predictions are reasonable to construct?
- Focus on two online problems
  - Online Flow Allocation in DAGs
  - Online Load Balancing with Restricted Assignments

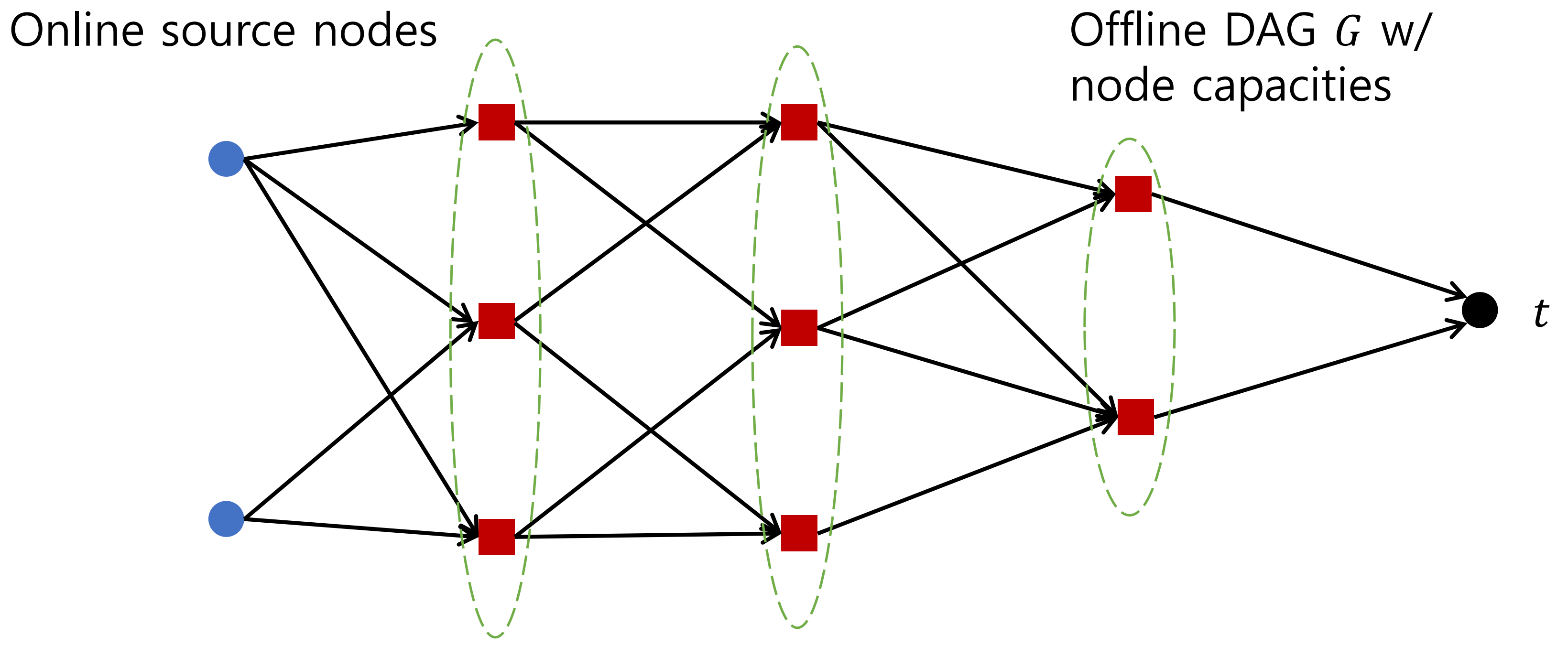
Online Flow Allocation in DAGs



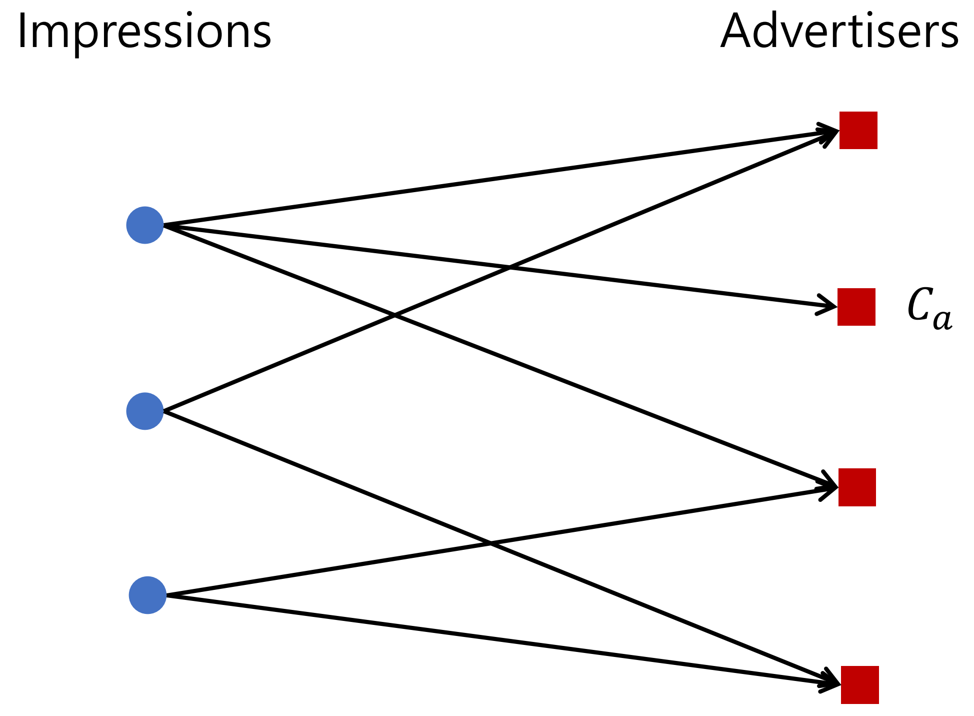
- Assign fractional unit flow to t from each online node s subject to node capacities
- Goal: maximize total value reaching t

Online Flow Allocation in DAGs

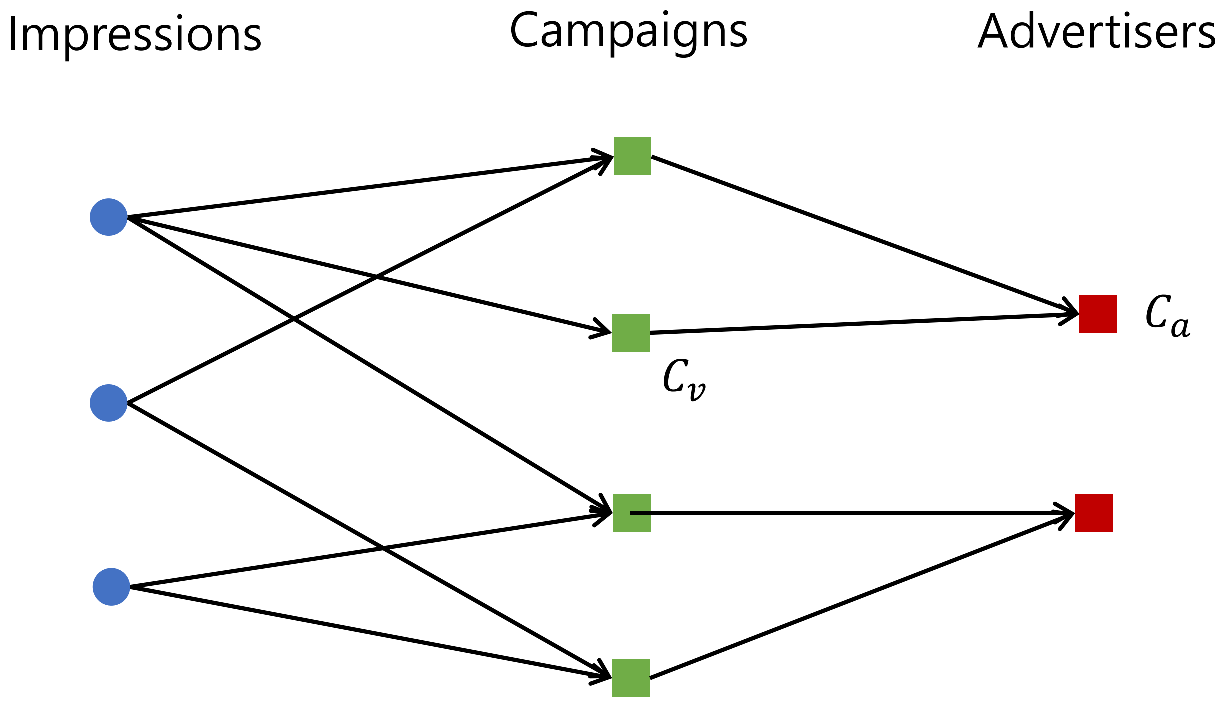
Layered case



Capacitated Online Matching



Online Matching w/ Campaigns



Parameter Robustness

- Algorithm given parameters $\hat{y}$
  - Prediction of true parameters y^*
  - E.g. guess of length of ski trip vs. true length
- Parameterize competitiveness by error γ
  - E.g. $\gamma = \|\hat{y} - y^*\|_1$
- Error typically depends on problem/prediction considered
- How to compare algorithms for the same problem with different types of predictions?

Instance Robustness

- Instance $\mathcal{I}$ is a vector of impression types
  - Type defined by outgoing neighborhood of an impression
  - $\mathcal{I}_i$ = number of impressions of type i
- $y(\mathcal{I})$ = "correct" prediction for instance $\mathcal{I}$
- Algorithm given $y(\mathcal{I})$ as advice
- Sees instance $\mathcal{I}'$ as online input
- Parameterize competitiveness by $\gamma = \|\mathcal{I} - \mathcal{I}'\|_1$

Learnability

- Formulate in terms of Data-driven algorithm design
  - [Gupta and Roughgarden, 2017], [Balcan et al. 2019]
  - Similar to PAC learning
- $ALG(I, y)$ = algorithm's value when given prediction y
- Unknown distribution $\mathcal{D}$ over instances
- Best prediction $y^* = \arg\max_y \mathbb{E}_{I\sim\mathcal{D}}[ALG(I, y)]$
- Use samples from $\mathcal{D}$ to compute $\hat{y}$ s.t. following occurs w.h.p.:

$$\mathbb{E}_{I\sim\mathcal{D}}[ALG(I, \hat{y})] \geq (1-\epsilon)\ \mathbb{E}_{I\sim\mathcal{D}}[ALG(I, y^*)]$$

  - Ideally polynomial number of samplers
  - Can also compare to $\mathbb{E}_{I\sim\mathcal{D}}[OPT(I)]$

Node Weights

- Predict a weight α for each offline node
- Lower weight means more restrictive node
- Use weights to define a fractional flow
  - Node u has a weight α_u
  - For each edge (u, v) let $x_{uv}(\alpha) = \frac{\alpha_v}{\sum_{v' \in N(u)} \alpha_{v'}}$
  - Node u sends an x_{uv}-fraction of its incoming flow along edge (u, v)

Existence of Good Weights

- Good weights give a $(1-\epsilon)$-approximate flow
- Existence for matching case shown by [Agrawal, Zadimoghaddam, Mirrokni 2018]
- Extension to general DAGs requires significant work
  - Careful algorithm and analysis

Results – Instance Robustness

- Instance $\mathcal{I}$ is a vector of impression types
- Algorithm given good weights $\alpha(\mathcal{I})$ as advice
- Sees instance $\mathcal{I}'$ as online input
- Parameterize competitiveness by $\gamma = \|\mathcal{I} - \mathcal{I}'\|_1$

Theorem: There is an online algorithm which achieves value

$$\max\left\{(1-\epsilon)OPT - 2\gamma, \frac{1}{d+1} OPT\right\}$$

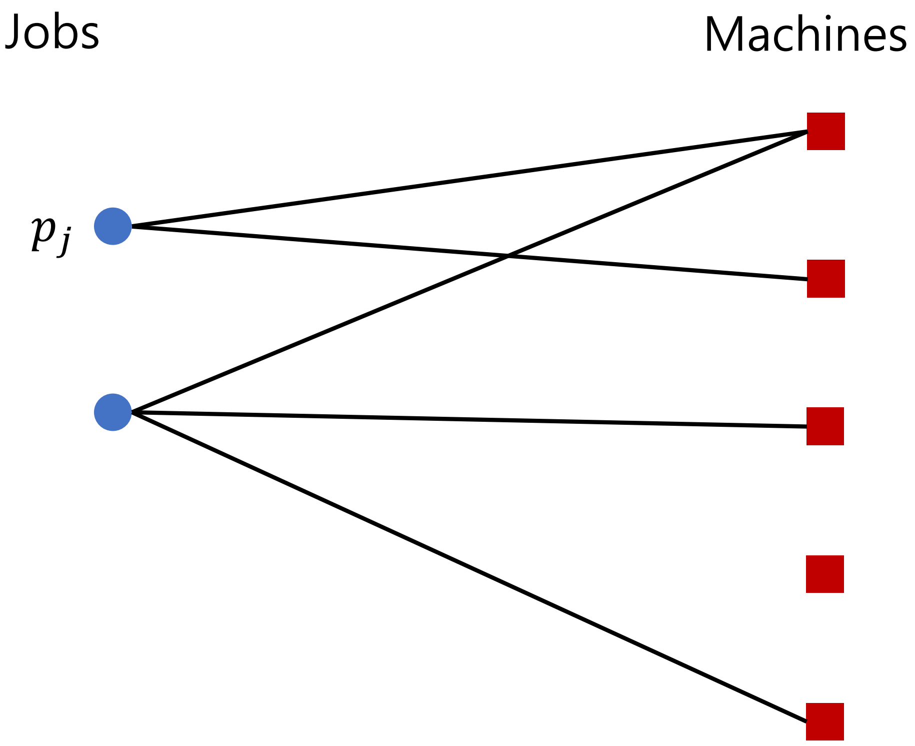
where d is the diameter of G without t

Results - Learnability

- Show learnability under two assumptions
  - $\mathcal{D}$ is a product distribution (impressions are independent)
  - The optimal flow in the “expected instance” routes at least a constant amount of flow through each node

Theorem: Under the two assumptions, there is an algorithm with polynomial sample complexity for learning the weights

Results – Load Balancing



- Consider node weights
- Existence of weights and parameter robustness considered in [Lattanzi et al. 2020]
- Show instance robustness and learnability under similar assumptions

Conclusions and Future Work

- Consider a new model for online algorithms with predictions
  - Learnability + Instance robustness
- How to construct predictions from past data?
- How do we measure robustness?
- Demonstrate improvements empirically

Thank you! Questions?