Online Load Balancing via Learned Weights

Silvio Lattanzi, **Thomas Lavastida**,
Benjamin Moseley, Sergei Vassilvitskii

MAPSP 2019

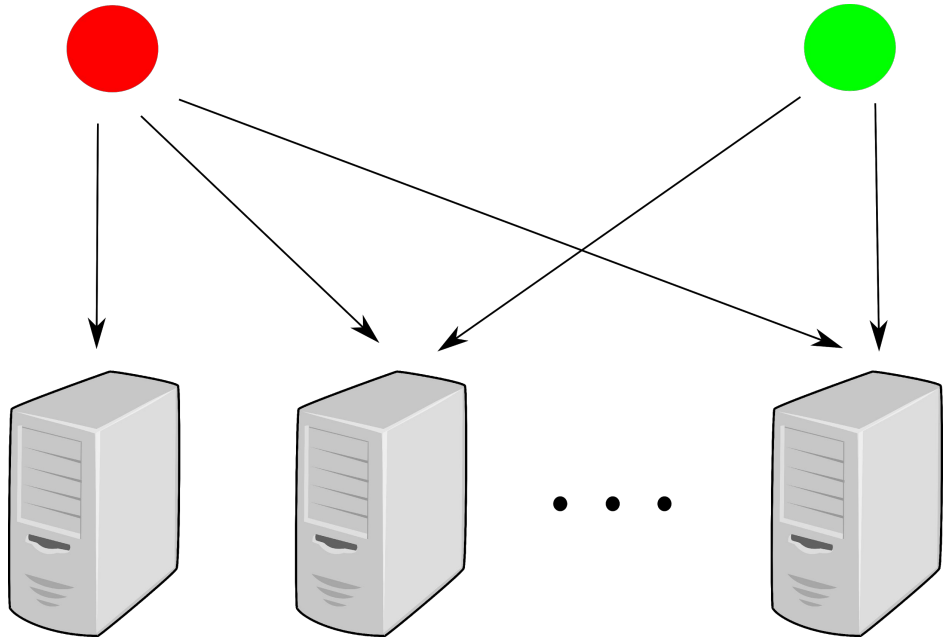
Carnegie Mellon University
Tepper School of Business

Data Center Scheduling

- Jobs arrive online
- Heterogeneous servers or jobs with constraints
- Assign jobs to servers under restricted assignment
- Minimize the maximum load

Online Load Balancing w/ Restricted Assignment

- *m* machines
- *n* jobs that arrive online
  - N(j) = subset of feasible machines for job j
  - p(j) = size of job j
- Load = total size of jobs assigned to a machine
- Goal: minimize max load



Online Load Balancing w/ Restricted Assignment

- Worst Case Competitive Analysis
  - For any input **I:**

$$\frac{ALG(I)}{OPT(I)} \leq c$$

- No algorithm can be better than Ω(log(*m*)) competitive
- Greedy algorithm is O(log(*m*)) competitive
  - [Azar, Naor, and Rom 1995]

Data Center Scheduling Revisited

- Want to go beyond worst case analysis
- Access to data about *past* jobs
- Machine learning successful in many other settings
- Can we learn something useful from the data?

- How do we model online with learning?
- What should we use learning to predict?
- How to account for errors?

Learning and Online Algorithms

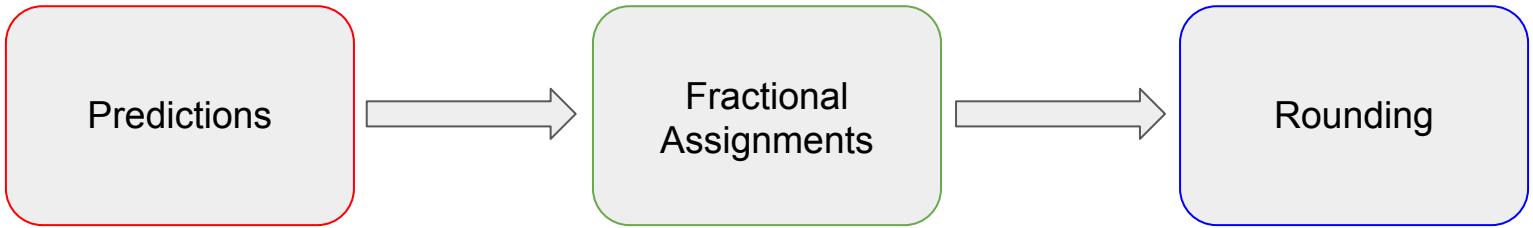
- Caching Problem [Lykouris and Vassilvitskii 2018]
- Ski Rental [Purohit et al 2018]
- Non-clairvoyant single machine scheduling [Purohit et al 2018]
- Many other works using predictions
  - Heavy hitters sketches [Hsu et al 2019]
  - Improved Bloom filters [Mitzenmacher 2018]

Model for Learning and Online Algorithms

- Abstract away exact ML model used
  - Many models may be applicable
  - Focus on quantity we want to predict
- Online Algorithm has access to predictions with some error
  - Size of predictions should be “small”
  - Ski Rental: predict number of days
  - Caching: predict next occurence of item
- Competitive ratio depends on the error
- Goal: beat log(m) when error is small

What to Predict?

- Ideas
  - Load of each machine in OPT?
    - Can pad instance so loads all the same
  - Optimal dual variables?
    - Very sensitive to small errors in the formulations we considered
- Our approach



Our Results

Theorem 1 (Machine Weights):

Let T be optimal max load. For any $\varepsilon > 0$, there exist machine weights and a rule to convert the weights to *fractional* assignments such that the resulting fractional max load is at most $(1+\varepsilon)T$.

Given predictions of the machine weights with *relative* error $\eta > 1$, there exists an online algorithm yielding *fractional* assignments for which the fractional max load is bounded by $O(\log(\eta)T)$.

Our Results

Theorem 2 (Rounding):

There exists an online algorithm that takes as input fractional assignments and outputs integer assignments for which the maximum load is bounded by $O((\text{loglog}(m))^3 T')$, where T' is maximum fractional load of the input. The algorithm is randomized and succeeds with probability > $1 - 1 / m^c$.

Corollary: There exists an $O((\text{loglog}(m))^3 \log(\eta))$ competitive algorithm for restricted assignment in the Online Algorithms with Learning setting

- Unit sized jobs
- We know T := optimal max load

Weights to Fractional Assignments

- Associate a weight to each machine
- Fractional assignment according to:

$$x_{ij}(w) = \frac{w_i}{\sum_{i' \in N(j)} w_{i'}}$$

- Weights chosen to satisfy following for each machine i

$$\sum_j x_{ij}(w) \leq (1+\epsilon)T$$

- Existence built off of [Agrawal, Zadimoghaddam, Mirrokni 2018]

Accounting for Error

- Fractional assignment rule

$$x_{ij}(w) = \frac{w_i}{\sum_{i' \in N(j)} w_{i'}}$$

- Say given predictions $\hat{w}$
- Look at relative error:

$$\eta = \max_i \frac{\hat{w}_i}{w_i}$$

- Assignment rule yields $O(\eta(1+\varepsilon)T)$ fractional load
- Key idea: modify weights over time to get $O(\log(\eta)\ T)$ fractional load

Rounding Algorithm

- Can’t use LP rounding due to [Lenstra, Shmoys, Tardos 1990]
  - Offline procedure and requires BFS
- log(*m*) lower bound for *deterministic* online rounding algos
- Standard randomized rounding?
  - Independently sample a machine according to each jobs fractional assignment
  - Loses log(*m*) factor
- Key idea (for one case): Analyze bipartite graph induced by jobs/machines
  - Graph breaks up into components with log(*m*) machines after some random assignments
  - Apply Greedy to remaining jobs
- Result nearly optimal - can’t do better than Ω(loglog(*m*)) w/ randomization

Conclusions

- Predictions can be used to break through worst case lower bounds
- What can we say about other online problems in this setting?
- Practical implementations

Thank You!