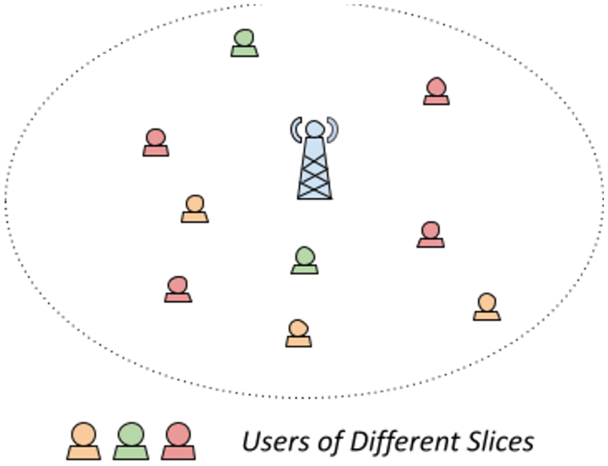
Online Learning for Hierarchical Scheduling to Support Network Slicing in Cellular Networks

Jianhan Song
ECE Department, UT Austin

Joint work with Gustavo de Veciana and Sanjay Shakkottai

Introduction: Network Slicing

- Hierarchical framework for resource allocation
  - Partition network resources into virtual slices
    - Map traffic flows to these slices
    - Slow timescale resource allocation
  - Use flow-level schedulers within slices
    - Fast timescale resource allocation



- Various motivations for slicing
  - Isolate groups from each other in the presence of traffic load fluctuations, e.g., Mobile Virtual Network Operator (MVNO)
  - Group flows with similar Quality of Service (QoS) requirements

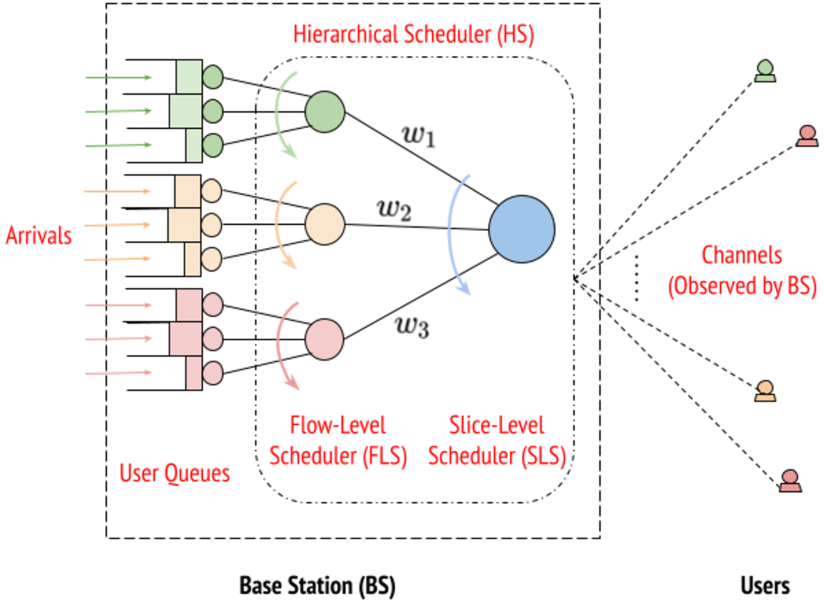
System Model: Hierarchical Scheduler

Traffic & Service Model

- Stochastic Arrivals & Channels

Hierarchical Scheduler HS($\mathbf{w}$)

- Users grouped into s slices
- **Slice-Level Scheduler:** allocate resources to slices based on weight vector $\mathbf{w}$ (e.g. GPS)
- **Flow-Level Scheduler:** pre-selected opportunistic scheduler (e.g. MaxWeight)
- $\text{HS}(\mathbf{w}) := (\text{SLS}(\mathbf{w}), (\text{FLS}_j)_{j\in[s]})$



How to Allocate Resources Among Slices?

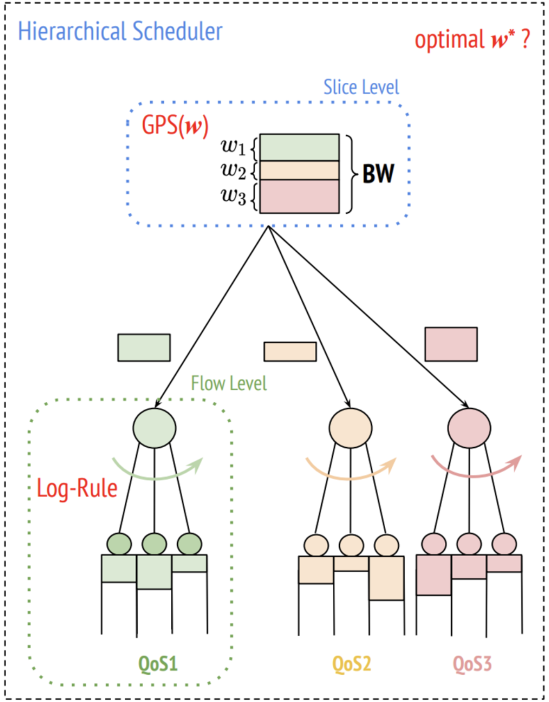
Various Rewards in Applications:

- ❖ mean-delay
- ❖ deadline constraints
- ❖ video quality, etc.

Question: What is the best slice-level allocation with respect to the current reward model?

Answer: It depends on …

- ❖ traffic load/service rate
- ❖ slicing structure
- ❖ flow-level scheduler deployed, etc.



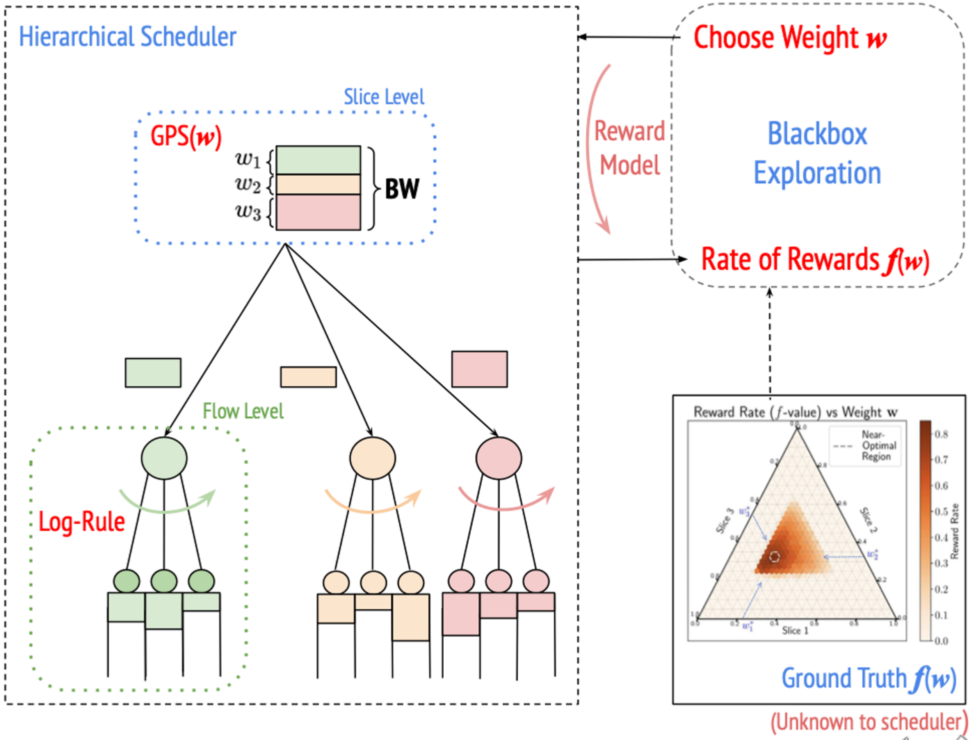
A Bandit Perspective

Our Approach:

- ❖ Model the problem as a *blackbox optimization*:

$$\text{Weight } \boldsymbol{w} \longrightarrow \text{Rate of Rewards Accrued over Time } f(\boldsymbol{w})$$

- ❖ Use a Bandit (online learning)
  $\Rightarrow$ Pull arms $\boldsymbol{w}$ and collect noisy feedbacks on $f(\boldsymbol{w})$



A Bandit Perspective: Challenges

I. In queueing systems, rewards collected are typically queue-dependent

  ⇒ e.g., delay-related rewards: long queues $\longleftrightarrow$ low utility

  ⇒ How to ensure feedback samples are conditionally independent?

➢ Bandit algorithm operating at new timescale -- Queueing Cycles

  ⇒ Ratio of rewards over cycles to be optimized

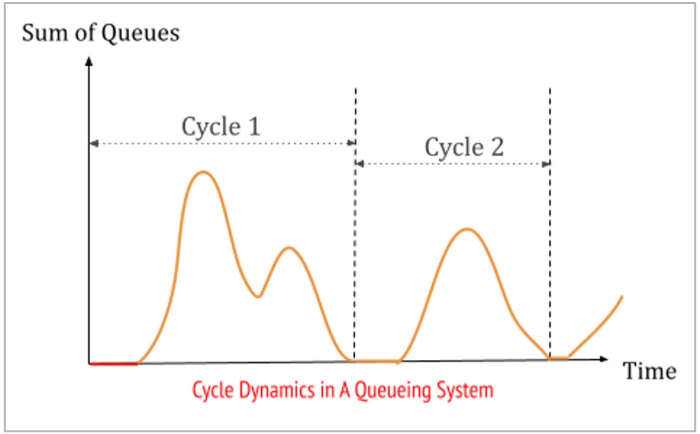
II. Infinitely many arms (weight choices) over a continuous set

➢ Optimistic Tree Search on the timescale of random cycles

Timescale: Cycles

Describing the system dynamics via cycles:

- ❖ Each cycle associated with a random length and reward accrued over its time
- ❖ Conditional Independence
  - ➢ Length/reward variables are independent across cycles** for any fix w
    $\Rightarrow$ essential for comparison of different arms

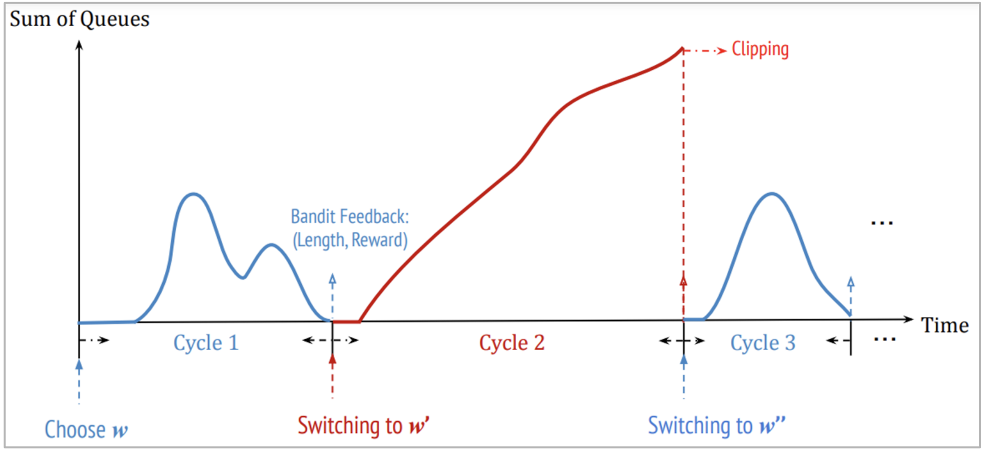


The empirical estimate of the rate of rewards $f(w)$:

$\rightarrow$ "cycle reward average / cycle length average" (of w-induced cycles)

** *under appropriate assumptions on traffic/reward models*

Cycles & Clipping



Caveat: What if a cycle never ends?
→ weights destabilizing the queueing system might be played during exploration

Clipping Mechanism: discarding packets currently in the system and force the start of a new cycle when a cycle is "too long"

- ❖ We define clipping thresholds -- a cycle is clipped if exceeding its threshold
- ❖ Threshold slowly growing -- logarithmically increasing with cycle index
  → To ensure "stable weights" eventually not getting clipped
- ❖ "Unstable weights" are penalized when clipped and rarely played

Optimistic Tree Search

Find optimal w^* via tree search:

- ❖ Partition the weight space into a binary tree $\mathcal{T}$
- ❖ General idea:
  - ➢ Build an "estimation tree" for the function $f(.)$ corresponding to $\mathcal{T}$ via bandit feedback: the deeper the tree, the better is the estimate
  - ➢ Choose weights from partitions with good estimates
  - ➢ Further grow the tree towards the optimal w^*

"Optimistic" -- Under-explored partitions are compensated (in terms of estimate score) to encourage exploration

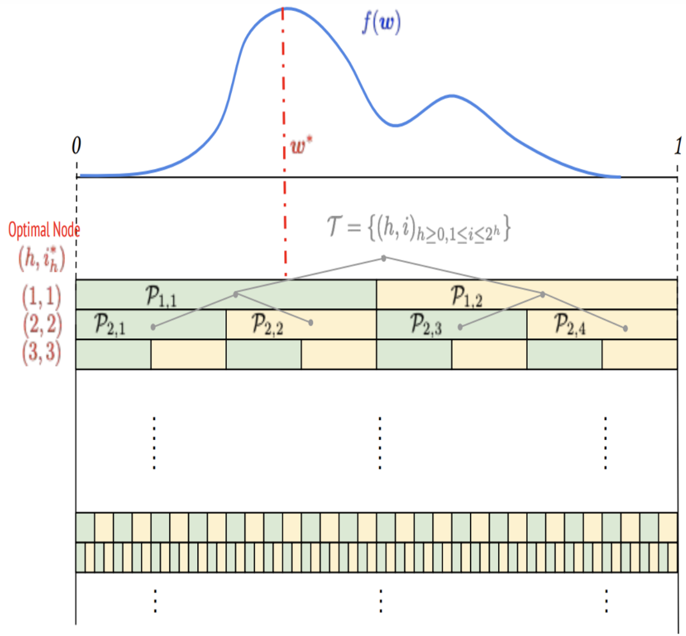


Illustration of Tree Partitioning (1-D Example)

Algorithm Overview

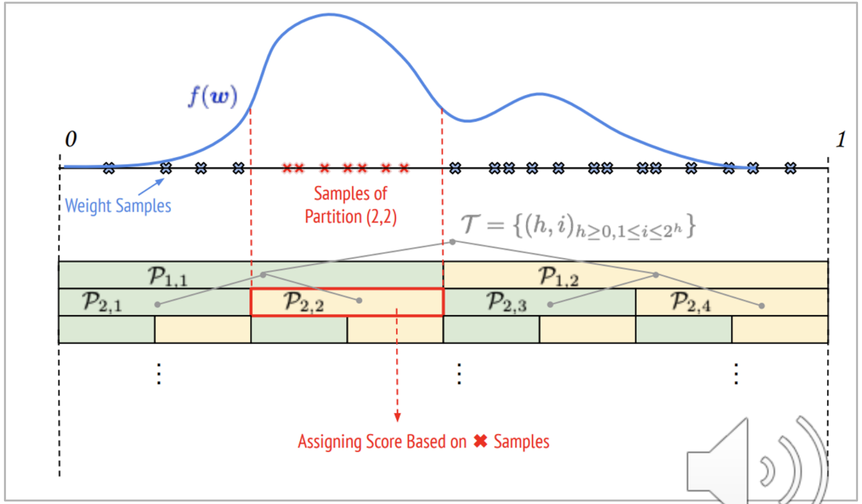
Our Algorithm -- Cycle-Based HOO with Clipping (CHOOC)

→ Optimistic Tree Search: UCT [KS2006], Zooming [KSU2008], HOO [BMSS2011], etc.

→ CHOOC: modified HOO algorithm adaptive to random queueing cycles & clipping

Algorithm Outline:

- ❖ Create hierarchical partitions → binary tree
- ❖ Dynamically assign scores to partitions
  - ➔ Score = Reward Average / Cycle Average + Exploration Bonus
- ❖ Exploit samples from partitions with "best" score

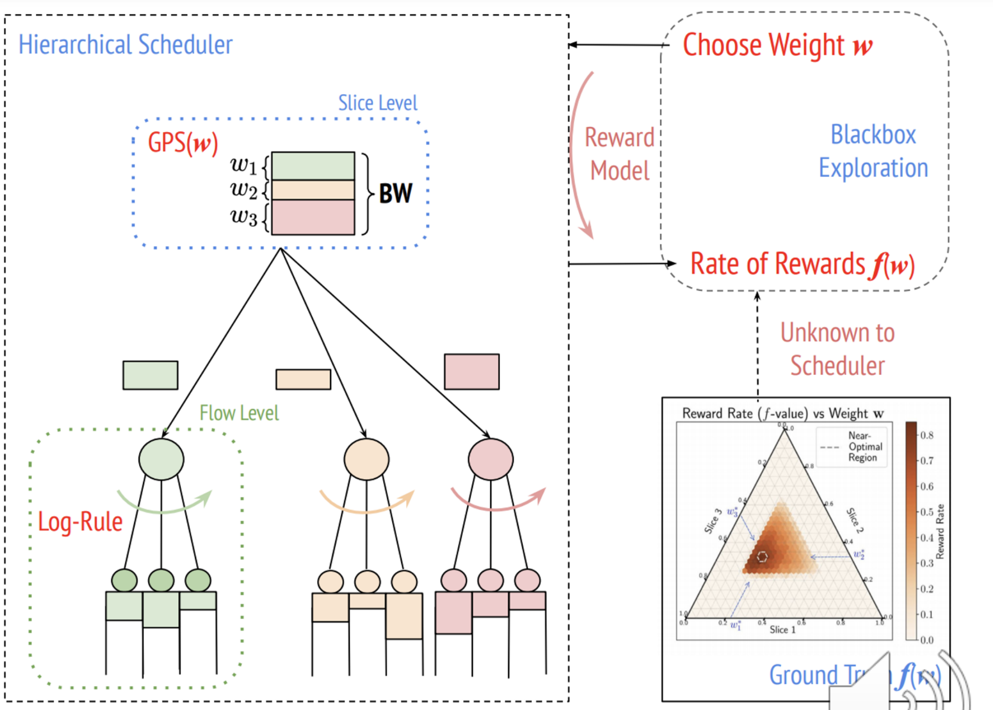


CHOOC Framework Summary

- Outer Step: CHOOC
- Inner Step: Samples generated by Hierarchical Scheduler with random length cycles + clipping

Theoretical Result:

- **Regret**: loss of rewards with respect to the optimal choice ***w****
- Sublinear Regret -- Same order of HOO despite random cycles and clipping mechanism



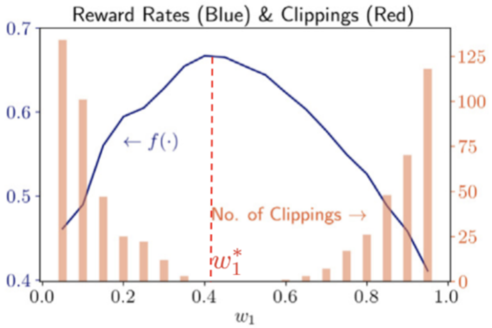
Performance Evaluation: Convergence Behavior

- Simulation setting: IMT Advanced evaluation guidelines for urban macro-cell deployments**

- 1 base station, 12 users grouped into 2 slices.
- Reward type:
  - Slice 1: Mean-delay
  - Slice 2: Meeting strict deadlines
- Slice-level Scheduler: GPS (Generalized Processor Sharing)
- Flow-level Schedulers: Log-Rule (opportunistic scheduler) for both slices

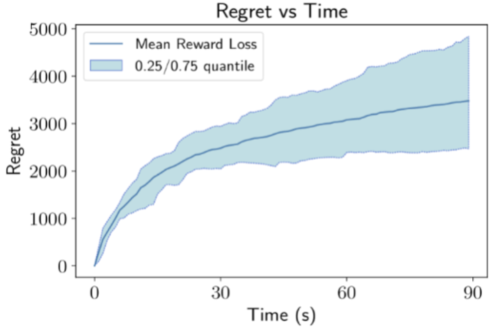
** M Series. "Guidelines for evaluation of radio interface technologies for imt-advanced." Report, *International Telecommunication Union*, 638, 2009.

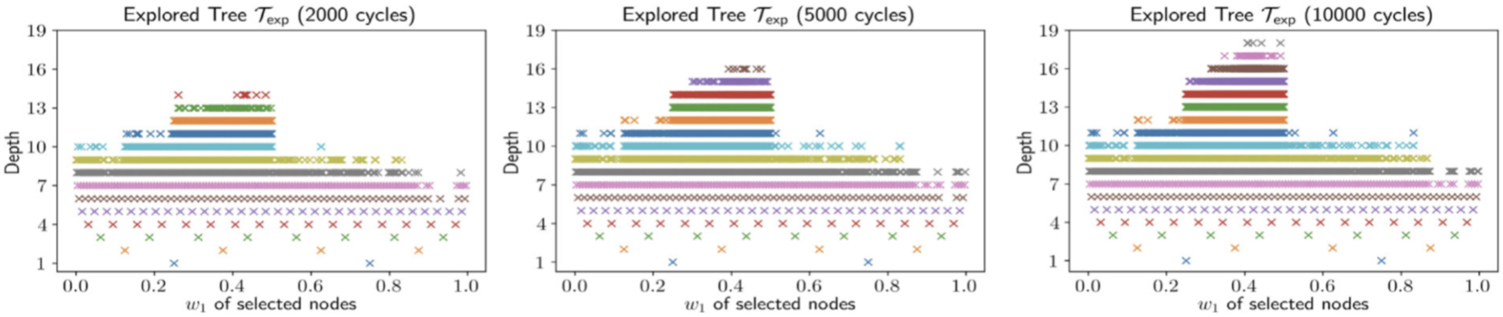
Performance Evaluation: Convergence Behavior



Ground Truth $f(\boldsymbol{w})$

Regret vs Time





Explored Tree (over Time)

Conclusion

- Parameterize the hierarchical scheduling model for network slicing by a weight vector & formulate it as an online blackbox optimization problem
- Bandit algorithm for online parameter selection - CHOOC
  - Optimistic tree search algorithm built from HOO with algorithmic/theoretic modifications to account for queueing cycles with clippings
  - Scheduler adaptively choosing weight vectors based on previous bandit feedback on a timescale of cycles
  - Verified by several simulation experiments

J. Song, G. de Veciana and S. Shakkottai, Online learning for hierarchical scheduling to support network slicing in cellular networks, *Performance Evaluation* (2021), doi: https://doi.org/10.1016/j.peva.2021.102237.