Learning-NUM: Network Utility Maximization With Unknown Utility Functions and Queueing Delay

XINZHE FU, EYTAN MODIANO

IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 30, NO. 6, DEC. 2022

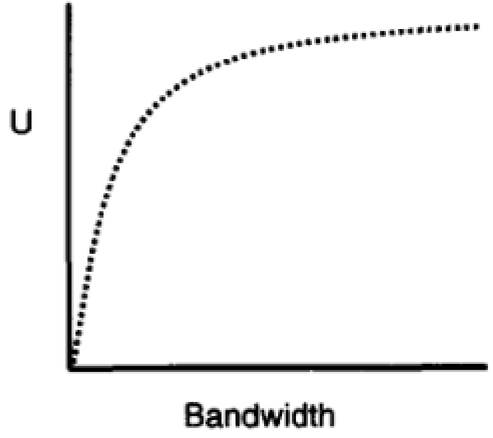
Burak Bozdağ – 504211552

Introduction

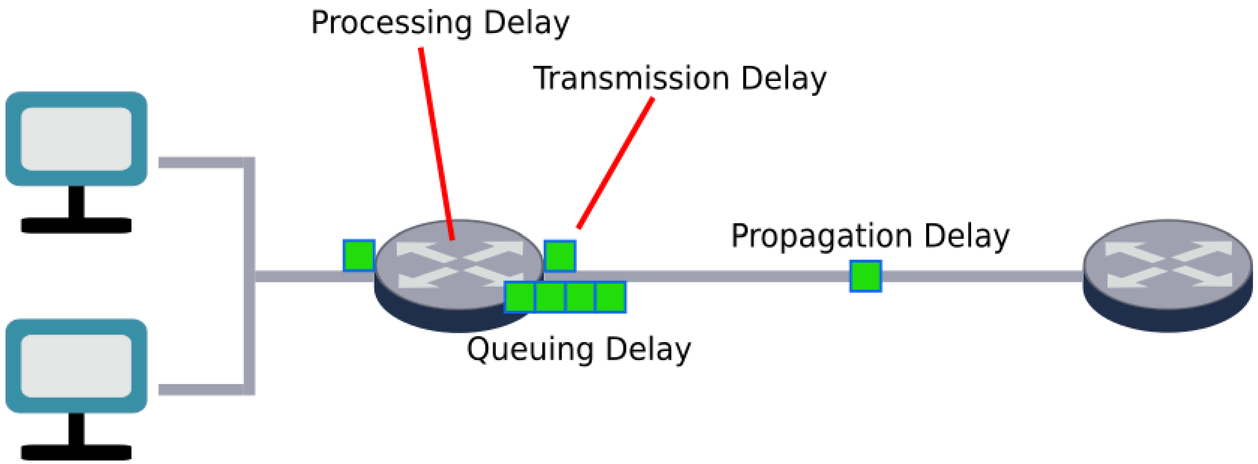
- Network utility maximization
- Fundamental problem in communication networks
- Unknown utility functions and queueing delay
- Learning-NUM: Maximizes utility over time

Background

- Utility functions:
  - Defined over [0, +∞)
  - Continuously differentiable
  - Non-decreasing
  - Concave

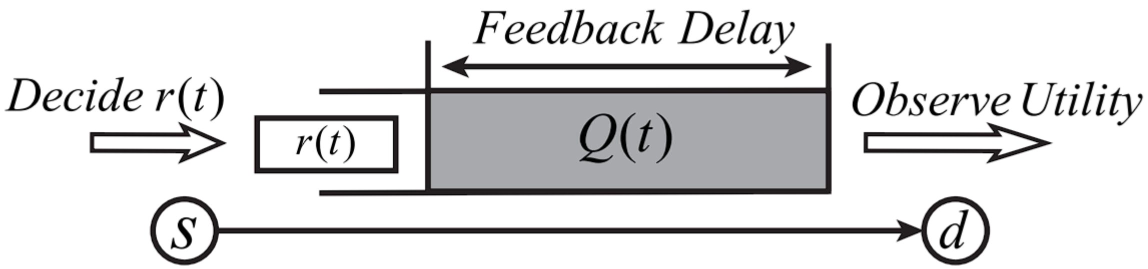


- Queueing delay



Challenges

- Utility functions and queueing delay are unknown
- Optimizing an unknown function
- Delays depends on decisions



Related Work

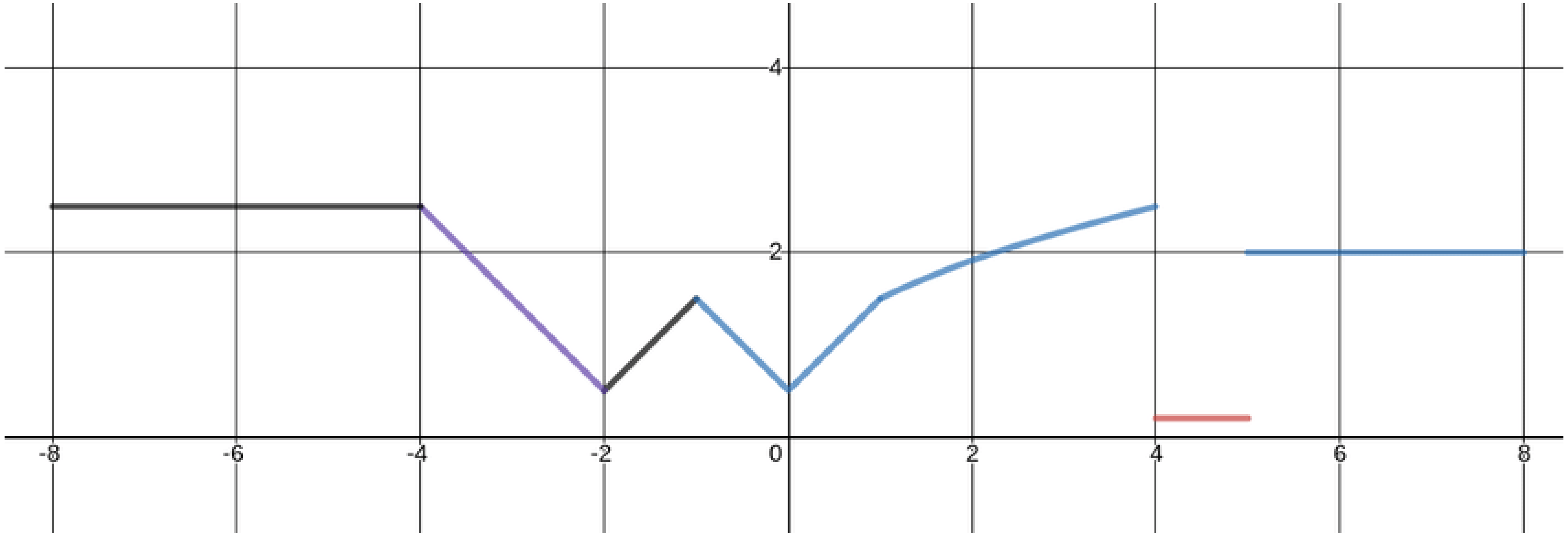
- Static – stochastic approaches
- Utility functions are **known**
- When utility depends on more constraints, it gets hard to estimate

Learning-NUM

- A new framework
- Utility values can be learned over time
- Value is observed after the job gets to the destination
- Determines job sizes and actions

Learning-NUM

- Online learning: Minimizing regret value
- Regret: Gap between expected and optimal
- Gradient sampling



GSMW Algorithm

- Gradient Sampling Max-Weight
  - Gradient sampling
  - Drift-plus-penalty
  - Backpressure routing
  - Max-Weight scheduling
- No-delay setting: $\tilde{O}(\sqrt{T})$-regret

Algorithm 1 The Gradient Sampling Max-Weight Algorithm.

Input: Network $\mathcal{G}(\mathcal{V}, \mathcal{E})$, parameters V, δ, α

Initialize: $\boldsymbol{x}(0) \in \mathcal{X}, \hat{r}_k(0) = \delta$.

for $t = 1, 2, \ldots, T$ **do**

$\boldsymbol{x}(t) := \arg\max_{\boldsymbol{x} \in \mathcal{X}} \sum_{i,j \in V} \sum_{k=1}^{K} A_{ij}^k(\omega(t), \boldsymbol{x})[Q_i^k(t) - Q_j^k(t)]$

for $k = 1, \ldots, K$ **do**

s_k injects job of size $\hat{r}_k(t) + \delta$ and observes $f_k(\hat{r}_k(t) + \delta)$.

s_k injects job of size $\hat{r}_k(t) - \delta$ and observes $f_k(\hat{r}_k(t) - \delta)$.

$\hat{\nabla} f_k(\hat{r}_k(t)) := \frac{f_k(\hat{r}_k(t)+\delta) - f_k(\hat{r}_k(t)-\delta)}{2\delta}$

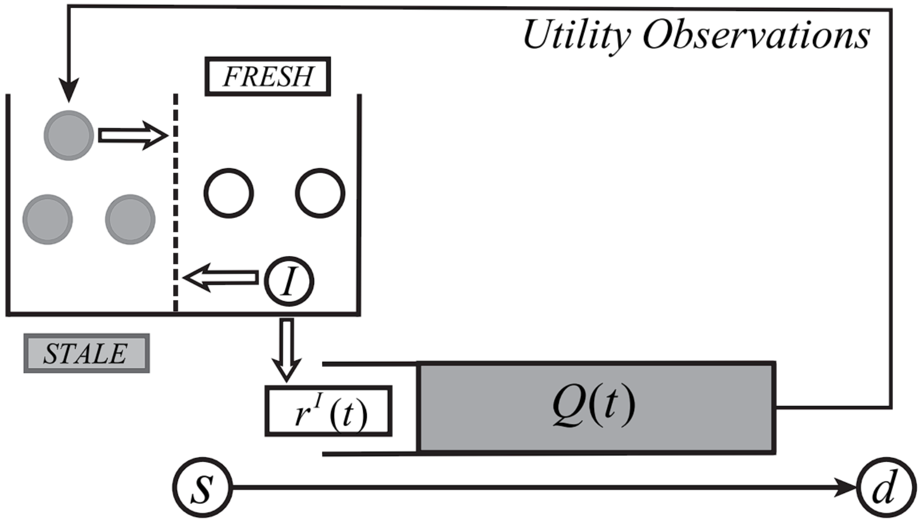
Update queue lengths according to $r_k(t), \boldsymbol{x}(t)$.

for $k = 1, \ldots, K$ **do**

$\hat{r}_k(t+1) := \mathcal{P}_{[\delta, B-\delta]} \left[\hat{r}_k(t) + \frac{1}{\alpha}(V \cdot \hat{\nabla} f_k(\hat{r}_k(t)) - Q_{s_k}^k(t)) \right]$

P-GSMW Algorithm

- Parallel-instance Gradient Sampling Max-Weight
- Two status: FRESH and STALE



Algorithm 2 The Parallel-Instance GSMW Policy.

Input: Network $\mathcal{G}(\mathcal{V},\mathcal{E})$, parameters V, δ, α, instance reservoir $\mathcal{I}$

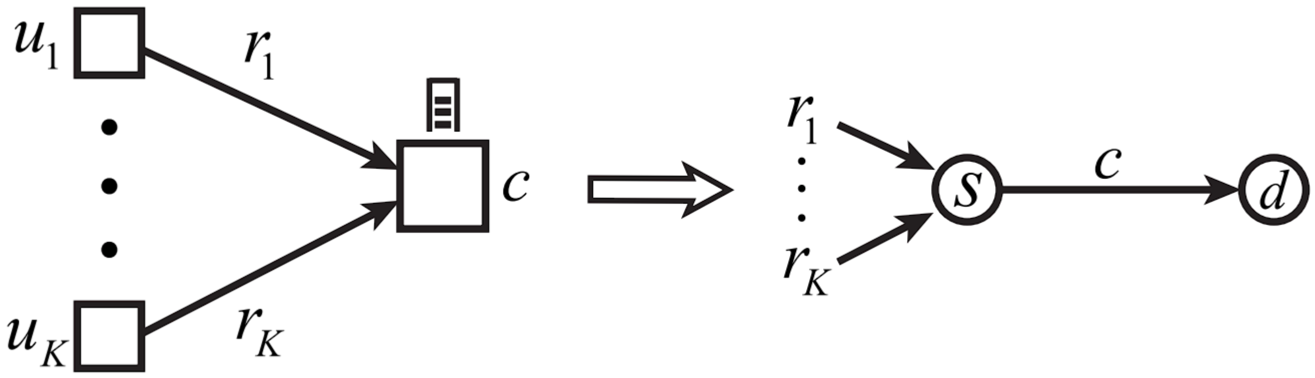
for $t = 1, 2, \ldots, T$ **do**
  $\boldsymbol{x}(t) := \arg\max_{\boldsymbol{x}\in\mathcal{X}} \sum_{i,j\in V} \sum_{k=1}^{K} A_{ij}^k(\omega(t), \boldsymbol{x})[Q_i^k(t) - Q_j^k(t)]$
  if There exists a FRESH instance $I_t \in \mathcal{I}$ **then**
    for $k = 1, \ldots, K$ **do**
      $\hat{r}_k^{I_t}(t) := \mathcal{P}_{[\delta, B-\delta]} \left[\hat{r}_k^{I_t}(t-1) + \frac{1}{\alpha}(V \cdot \hat{\nabla} f_k(\hat{r}_k^{I_t}(t-1)) - Q_{s_k}^k(t))\right]$
      s_k injects job of size $\hat{r}_k^{I_t}(t) + \delta$ and another job of size $\hat{r}_k^{I_t}(t) - \delta$.
      Change the status of I_t to STALE.
  else
    Create a new instance I_t
    For each k, initialize $\hat{r}_k^{I_t}(t) := \delta$, and s_k injects job of size $\hat{r}_k^{I_t}(t) + \delta$ and another job of size $\hat{r}_k^{I_t}(t) - \delta$
  Update queue lengths according to $r_k(t), \boldsymbol{x}(t)$.
  $\{\hat{r}_k^J(t)\} := \{\hat{r}_k^J(t)\}$ for $J \in \mathcal{I}, J \neq I_t$.
  Collect utility observations from delivered jobs and form gradient estimates $\hat{\nabla} f_k(\hat{r}_k^I(t)) := \frac{f_k(\hat{r}_k^I(t)+\delta) - f_k(\hat{r}_k^I(t)-\delta)}{2\delta}$
  for STALE instance $I \in \mathcal{I}$ **do**
    Change the status of I to FRESH if it has obtained all outstanding gradient estimates.

Applications

- Database query
- Job scheduling
- Video streaming

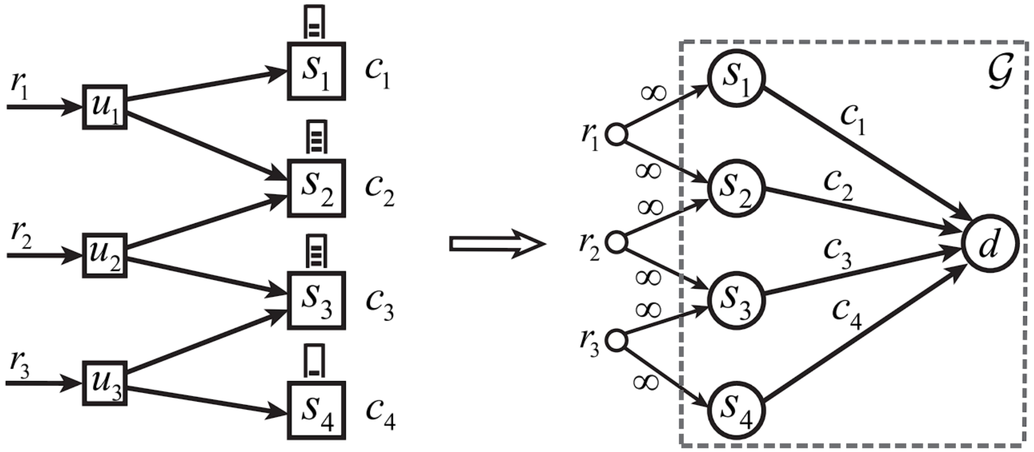
Database Query

- K users querying a central database
- Maximize total utililty of the processed queues
- Sublinear regret value: $\tilde{O}(T^{3/4})$-regret



Job Scheduling

- 3 Job schedulers and 4 servers
- Dispatcher determines resource
- Maximize the total utility gained from jobs
- Example of ML in Cloud
- Sublinear regret value: $\tilde{O}(T^{3/4})$-regret

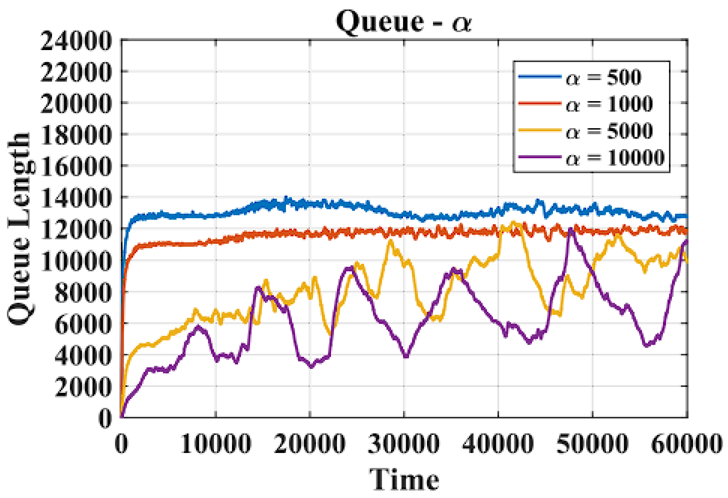


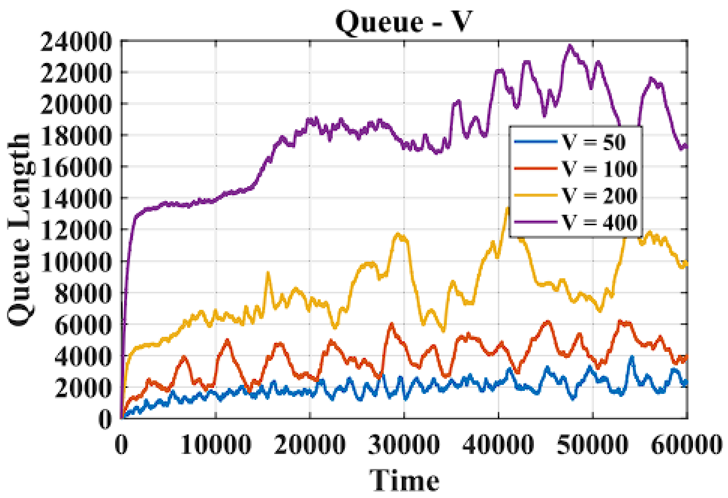
Video Streaming

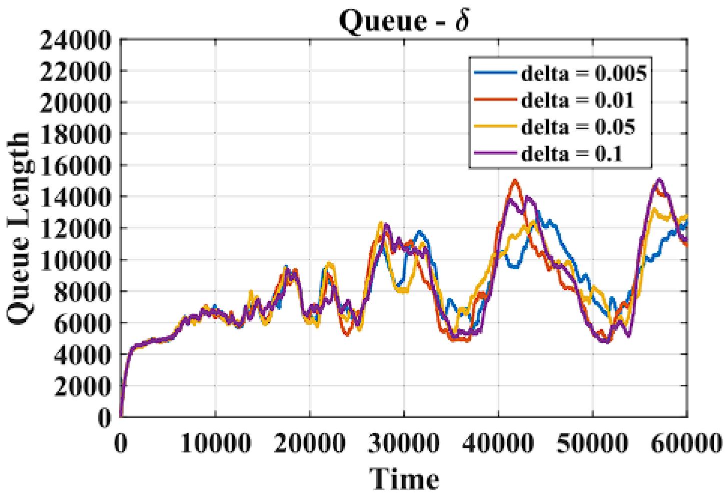
- Discuss the advantages of Learning-NUM, such as accuracy and efficiency
- K users streaming video from K servers
- Servers send chunks to users
- Determining size of the video chunks
- Maximizing delivered video chunks
- Sublinear regret value: $\tilde{O}(T^{3/4})$-regret

Evaluation: Parameters

- α: Controls the step size with a larger α indicating a smaller step size.
- V: Adjusts the relative weights on utility maximization and queue stability, with a larger V indicating that the policy tries to increase the job sizes more aggressively.
- δ: Controls the approximation error of our estimate gradients with respect to the true gradients.

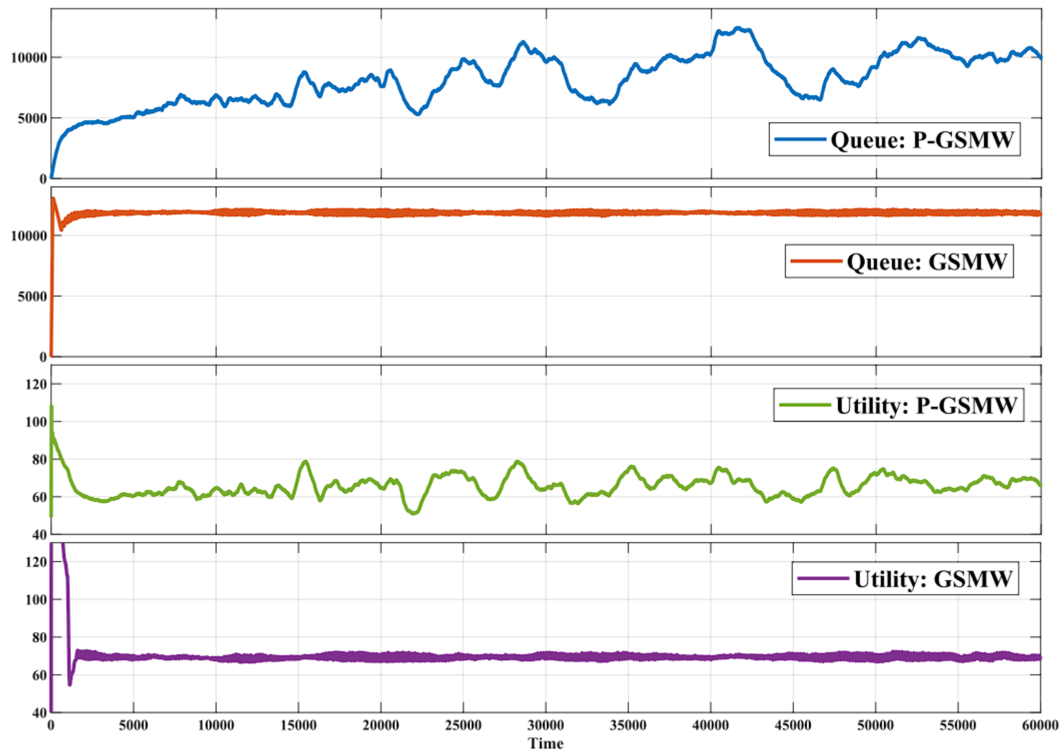






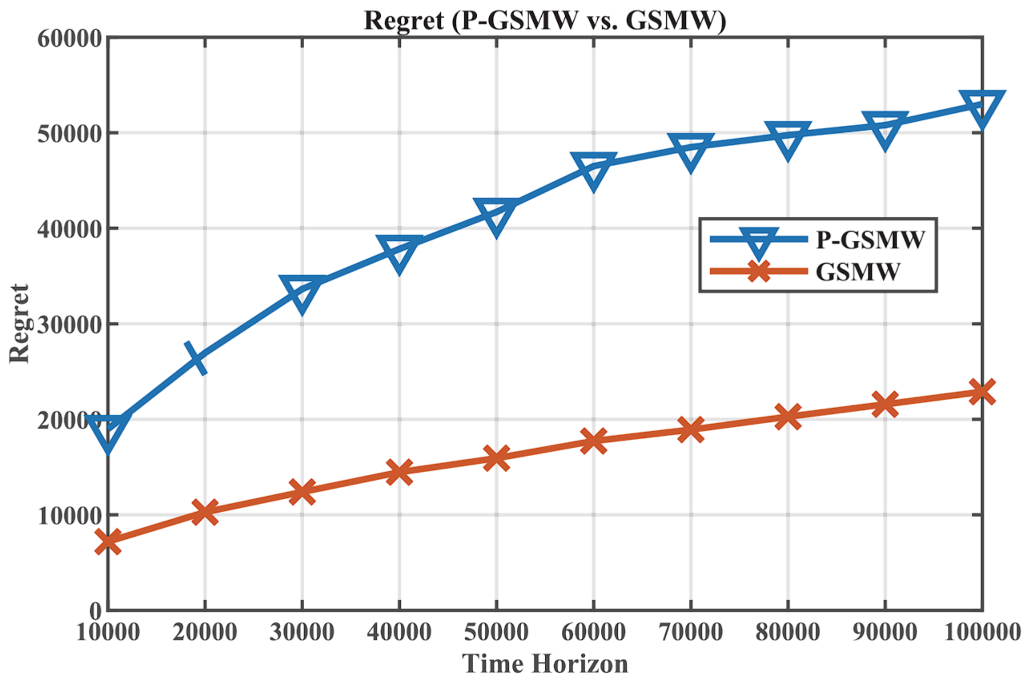
Evaluation

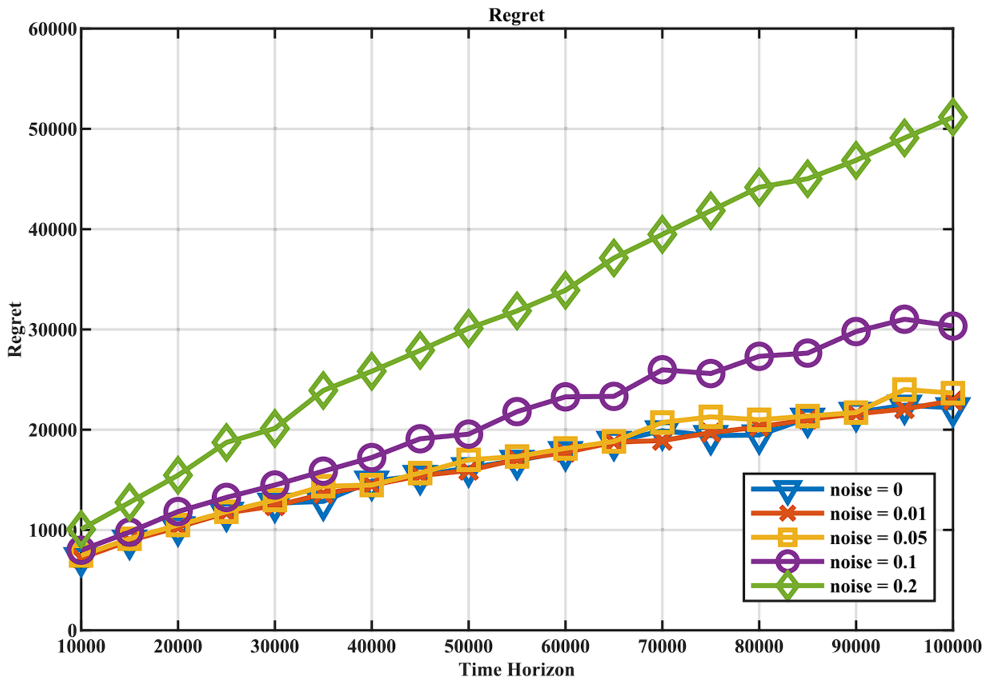
- Queue length and instantaneous utility behavior under P-GSMW and GSMW:



Evaluation

- Regrets of P-GSMW and GSMW:





Conclusion

- A new framework: Learning-NUM
- With unknown utility functions and queueing-style delay
- Achieved sub-linear regret

- Future work:
  - Noiseless, no-delay case does not achieve lower bound of regret
  - More robust policies for noisy scenarios

References

[1] X. Fu and E. Modiano, "Learning-NUM: Network Utility Maximization With Unknown Utility Functions and Queueing Delay," in IEEE/ACM Transactions on Networking, vol. 30, no. 6, pp. 2788-2803, Dec. 2022, doi: 10.1109/TNET.2022.3182890.

[2] Wikipedia contributors. (2022, June 2). Drift plus penalty. In Wikipedia, The Free Encyclopedia. Retrieved 21:49, May 2, 2023, from https://en.wikipedia.org/w/index.php?title=Drift_plus_penalty&oldid=1091167397

[3] Wikipedia contributors. (2021, September 21). Backpressure routing. In Wikipedia, The Free Encyclopedia. Retrieved 21:49, May 2, 2023, from https://en.wikipedia.org/w/index.php?title=Backpressure_routing&oldid=1045657229

[4] Wikipedia contributors. (2023, February 2). Regret (decision theory). In Wikipedia, The Free Encyclopedia. Retrieved 21:51, May 2, 2023, from https://en.wikipedia.org/w/index.php?title=Regret_(decision_theory)&oldid=1137023034