Caching in Base Station with Recommendation via Q-Learning

Code : GitHub - facundolezama19/indoor-localization-gnn: Notebooks con los códigos utilizados para el proyecto final del curso.

INDEX

INTRODUCTION

Introduction

- **Background**
  - 모바일 데이터 traffic 폭발적인 증가 = 사용자 대기 시간 증가
  - “proactively caching”(능동적 캐싱) is popular contents at base station (BS) : 낮은 가격, 더 나은 결과
  - cached files 이 많이 요청되고, 분포도가 잘 알려져 있는 경우 사용하기 좋다.
  - But Mobile System과 같이 small and dynamic user 가 존재하는 환경에서는 완벽하지 않다.

- **Existing Approach problem**
  - Blind : the users in a cell do not know which files have been cached at the BS.
  - 사용자 행동의 randomness 때문에 user가 요청한 파일이 base station(BS)에 cached되어 있음에도 요청을 보낼 수 없는 경우가 존재.
  - As a result, the efficiency in using the cache resource is low

- **Solution**
  - integrate “recommendation” with local caching in wireless edge.

Introduction

Informed information = abstract = recommendation

❑ Solution

- integrate “recommendation” with local caching in wireless edge.
- Base Station(BS) inform the users about “what it has cached” and “why the files are cached” in a cost-effective way.
- 예시 : 요청 수에 따른 파일 순위를 사용자에게 알려줌

❑ Benefit

- 사용자는 일종의 “recommendation”(=abstract)으로 인식. 이후 request 여부를 판단.
- 이전 approach에 비해 사용자는 자신에게 필요한 파일이 근처에 cached 여부를 알 수 있음. 혹은 몰랐던 파일의 존재 여부도 알 수 있음.
- 이는 사용자가 정보를 알게 됨으로써 할 수 있는 다른 행동을 유도할 수 있음.

❑ Challenge

- 추천 후 사용자의 request 확률은 알 수 없음 = 파일을 BS에 캐싱할 가치가 있는지 미리 알 수는 없음
- 따라서 Q-learning을 사용하여 request 확률과 mobile user의 random arrival and departure 확률을 예측.
- 최종적인 목표 : cached file을 replace하는 “policy 수립”

SYSTEM MODEL

System Model

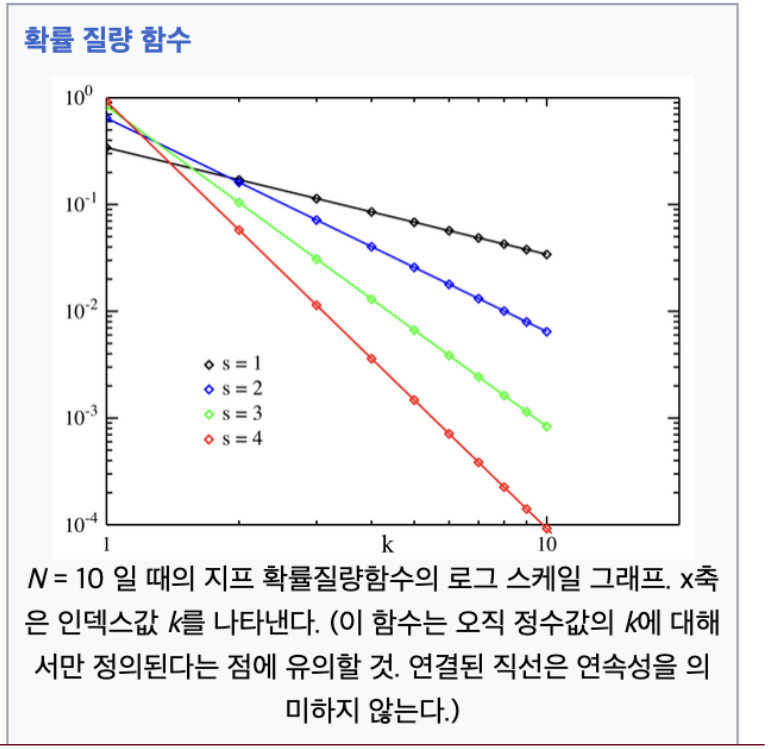
❑ Zipf’s distribution

- 지프의 법칙(Zipf's law)은 수학적 통계를 바탕으로 밝혀진 경험적 법칙으로, 물리 및 사회 과학 분야에서 연구된 많은 종류의 정보들이 지프 분포에 가까운 경향을 보인다는 것을 뜻한다.

지프의 법칙에 따르면 N개의 요소들 가운데 순위가 k 번째인 요소의 사용빈도 $f(k;s,N)$는 다음과 같다.

$$f(k;s,N) = \frac{1/k^s}{\sum_{n=1}^{N}(1/n^s)}$$

지프의 법칙



N = 10 일 때의 지프 확률질량함수의 로그 스케일 그래프. x축은 인덱스값 k를 나타낸다. (이 함수는 오직 정수값의 k에 대해서만 정의된다는 점에 유의할 것. 연결된 직선은 연속성을 의미하지 않는다.)

출처 : 위키피디아
https://ko.wikipedia.org/wiki/%EC%A7%80%ED%94%84%EC%9D%98_%EB%B2%95%EC%B9%

System Model

- **File popularity (=the probability that the i-th file is requested)**
  - **can be modeled as a Zipf's distribution with skewness parameter γ_0**
  - **Probability($p_{i,0}$) = $i^{-\gamma_0} / \sum_{j=1}^{N_f} j^{-\gamma_0}$ (N_f : $file$ N 개)**
    - **Recommend list(=catalog) contains N_f files**

- **BS starts to operate**
  - **초기 구성은 과거 request 순으로 recommend list 구성 or 랜덤 선택**
  - **Each time interval 마다 update 진행.**
    - **replaces some files in the cache by online learning,**
  - **Update 된 recommend list(=abstract) 를 사용자에게 broadcast.**

System Model

- In real-world systems, the request probability with recommendation relates to many factors that help people make choice.
- Nonetheless, previous research indicates that recommending the most popular items will make them more popular. (원래도 사용 빈도 높은데 추천해주게 되면 더 높아지는 현상)
  - 따라서 recommend 이후 request probability를 예측할 때는 skewness parameter(γ_1)를 설정
  - $\gamma_1 > \gamma_0$
  - Then, probability $\begin{cases} p_{i,1} : if\ the\ ith\ file\ is\ recommended \\ p_{i,0} : otherwise \end{cases}$

System Model

❑ Model Structure

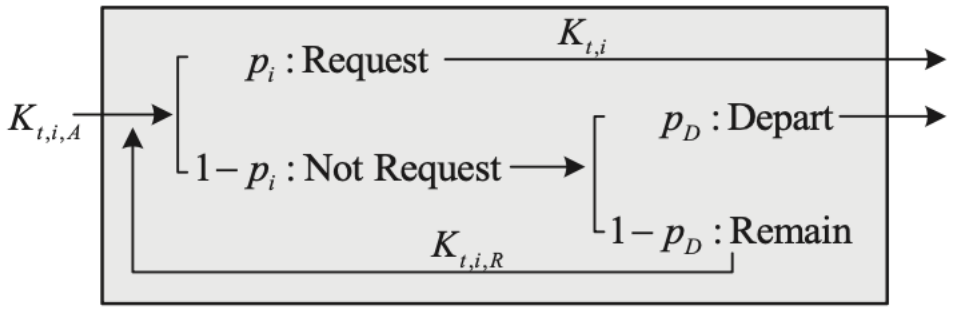


Fig. 1. System model and numbers of users.

- $K_{t,i,A}$: the number of users that enter the cell in the t-th interval and have not requested the ith file before
- $K_{t,i,R}$: the number of users that neither request the i-th file nor depart the cell in the $(t-1)$th interval.

- p_D : *probability of user depart the cell*
- $1 - p_D$: *probability of user do not depart*$(= remain)$

- p_i :the probability that the i-th file is requested
- $1 - p_i$:the probability that the i-th file is not requested

PROBLEM FORMULATION

PROBLEM FORMULATION

❑ Replacement policy(어떤 file을 cache 할 것 인가?)

- $r_{t,i} = a_{t,i}(K_{t,i}b - c)$, b : benefit, c : cost
  - $r_{t,i}$: reward.
  - $K_{t,i}$: the number of users requesting the i-th file in the t-th time inverval
  - $a_{t,i}$: reflects the caching action(if i-th file is cached in the t-th time interval, $a_{t,i}$=1. Otherwise $a_{t,i}$=0)

❑ Replace

- Benefit : Reduce access latency and the traffic load of the backhaul
- Cost : incurs extra backhaul traffic and consumes wireless resources.

백홀(Backhaul)
다수의 통신망을 통해 데이터를 전송하는 계층적 구조로 된 통신망에서 주변부 망(edge|com network)을 기간망(backbone network)이나 인터넷에 연결시키는 링크.
https://terms.naver.com/entry.naver?docId=5678575&cid=42346&categoryId=42346

PROBLEM FORMULATION

- $r_{t,i} = a_{t,i}(K_{t,i}b - c)$
  - $K_{t,i}$ (number of users who request file) t – th time interval 시작점에 주어지면 reward 극대화하는 $a_{t,i}$(action) 를 구할 수 있다.
  - But random user arrivals and random user requests 로 인해 사실상 $K_{t,i}$ 정보는 사전에 얻을 수 없음.
  - 따라서 $K_{t,i}$ is a random variable.

- $K_{t,i,A}(= k_A)$, $K_{t,i,R}$ $(= k_R)$
  - $K_{t,i,A}$: t-th interval에 cell에 들어왔고, i-th file을 요청한적 없는 사용자 수
  - $K_{t,i,R}$: (t-1)-th interval에 cell을 떠나지 않고 i-th file을 요청하지도 않은 사용자 수
- p_i :the probability that the i-th file is requested
- $K_{t,i}$ follows B(n,p) : B($k_A + k_R$, p_i)
  - B : Binomial distribution
  - requests the i-th file independently with probability p_i

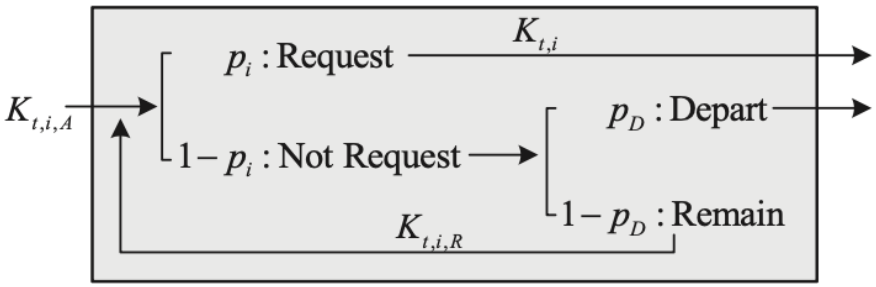


Fig. 1. System model and numbers of users.

PROBLEM FORMULATION

❑ Poisson distribution

- 푸아송 분포(Poisson distribution)는 확률론에서 “단위 시간” 안에 어떤 사건이 몇 번 발생할 것인지를 표현하는 이산 확률 분포이다.

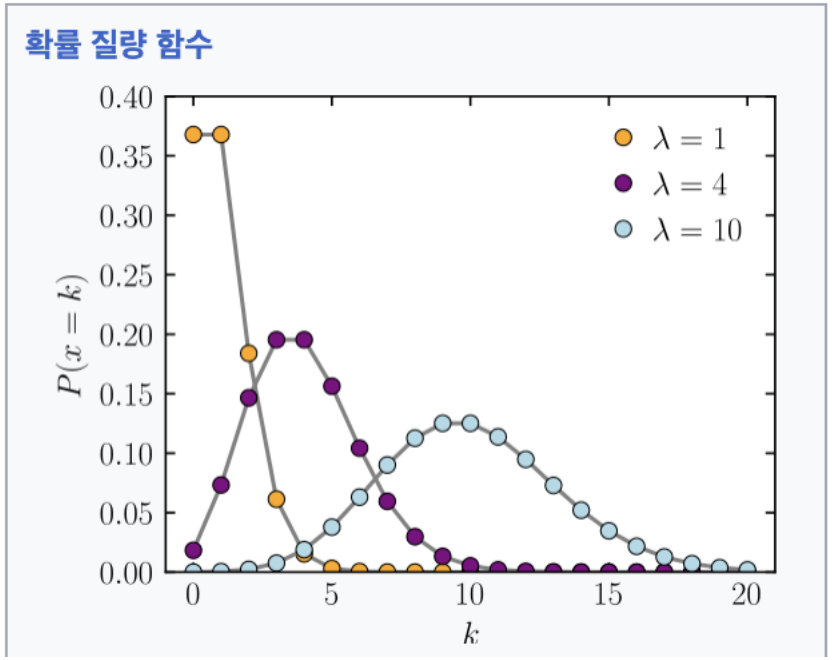
정의 [편집]

정해진 시간 안에 어떤 사건이 일어날 횟수에 대한 기댓값을 λ라고 했을 때, 그 사건이 k회 일어날 확률은 다음과 같다.

$$f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!},$$

여기서 e는 자연상수이다.

푸아송 분포



출처 :https://ko.wikipedia.org/wiki/%ED%91%B8%EC%95%84%EC%86%A1_%EB%B6%84%ED%8F%AC

PROBLEM FORMULATION

- $K_{t,i}$ is a random variable following Poisson distribution.

- $K_{t,i,R}$ depends on p_i and p_D(probability i-th file is requested and user depart the cell)
  - $K_{t,i,R}$ associated with the caching action in the last interval $a_{t-1,i}$
    - $K_{t,i,R}$ = 1 ~ t-1 interval 까지 i-th file을 요청한적 없고, 남아있는 사용자의 수. 따라서 $K_{t,i,R}$ follows a first-order Markov chain
  - First-order Markov chain : $P(K_{t,i,R} \mid K_{1,i,R}, a_{1,i}, K_{2,i,R}, a_{2,i}, \dots, K_{t-1,i,R}, a_{t-1,i})$
    - $= P(K_{t,i,R} \mid K_{t-1,i,R}, a_{t-1,i})$
    - 누적으로 볼 수 있지만 어쨌든 $K_{t-1,i,R}$에 다 포함되어 있는 것
  - 결론적으로 $K_{t,i}$ 는 이전과 현재 interval의 caching actions($a_{t-1,i}$)에 의해 결정된다.

- Brief conclusion:
  - Each interval 의 시작점에서 file의 cache 여부에 대해 current reward와 future reward 모두 고려해야함을 알 수 있음.

PROBLEM FORMULATION

- To this end, we define the long-term system reward(current reward와 future reward 모두 고려)
  - $w_t = \sum_{\tau=t}^{\infty} \beta^{\tau-t} \sum_{i=1}^{N_f} r_{\tau,i}$
  - β : discount factor (current action이 more distant interval에 대한 reward는 적음을 나타냄)

- 목적 : maximize long-term system reward , design of caching policy
  - design of caching policy can be treated as a Markov decision process (MDP)
  - multi-armed bandit (MAB) algorithm is used.

- 문제 : p_i , p_D unknown, even given a caching policy π ($r_{t,i} = a_{t,i}(K_{t,i}b - c)$)
  - MAB 는 immediate reward가 있는 상황에서만 적용 가능.
  - cannot deal with the long-term reward.

- Solution:
  - Q-learning based cached policy

PROBLEM FORMULATION

❑ Q-learning based cached policy

- dynamically select files from the catalog in each interval.
- Online algorithm에서는 state와 action이 current interval(t)에 존재함이 당연하므로 subscript는 생략함.

❑ Caching Policy with a Single File

- 1. optimal policy
  - $\pi^*(k_R) = \underset{a}{argmax}\, Q(k_R, a)$, $Q(k_R, a)$: *Q-value*
    - Q-value : estimated maximal long-term system reward(a under state k_R)
- 2. maximal system reward (over all possible actions)
  - $V^{\pi^*}(k_R) = Q(k_R, \pi^*(k_R))$
- Q-value(=Q-value function의 output) 를 학습시키는 것.

PROBLEM FORMULATION

❑ Caching Policy with a Single File

1. **Given Q-value and state**
2. **The action should be selected according to** $\pi^*(k_R) = \underset{a}{argmax}\, Q(k_R, a)$
   - **Expected long-term system reward maximized**
3. **Then, action(a)** $\begin{cases} \underset{a}{argmax}\, Q(k_R, a) : with\ probability\ \varepsilon \\ either\ 0\ or\ 1\ randomly : with\ probability\ 1-\varepsilon \end{cases}$
4. **After action a**
   1. state$(k_R) \rightarrow update\ k_R'$
   2. $Q(k_R, a) \leftarrow (1-\alpha)Q(k_R, a) + \alpha\left[r + \beta \max_{a'} Q(k_R', a')\right]$ **update. (**α *is predetermined parameter*)
   3. $\left[r + \beta \max_{a'} Q(k_R', a')\right] = r + \beta V^{\pi^*}(k_R')$, $\quad V^{\pi^*}(k_R) = Q(k_R, \pi^*(k_R))$
   4. $Q(k_R, a) \leftarrow (1-\alpha)Q(k_R, a) + \alpha(r + \beta V^{\pi^*}(k_R'))$
   5. **Immediate reward Long-term impact of a + Future reward**

PROBLEM FORMULATION

- **Caching Policy with a Single File**
  - Includes an immediate reward and a future reward (same as MAP algorithm)
  - Q-value는 random sample이므로 p_i , p_D 등 선험적인 확률 분포에서 가져올 필요 없음.
    - p_i , p_D unknown 이라 못쓰는 문제 해결

- **결론 : Q-learning able to obtain the expectation “without the knowledge of probabilities”**

- **질문 : Q-value? Q-learning?**
  - Q-learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state.
  - It does not require a model of the environment (hence "model-free"), and it can handle problems with stochastic transitions and rewards without requiring adaptations.
  - https://en.wikipedia.org/wiki/Q-learning

PROBLEM FORMULATION

- **Algorithm 1. Q-Learning based Caching Policy with Single File**
  - **Initialize the Q-value for each state-action pair.**
  - **Step 1: At the beginning of the current interval, count the number of remaining users in the system k_R**
  - **Step 2: Choose action a according to, action(a)** $\begin{cases} \underset{a}{argmax}\, Q(k_R, a) : with\ probability\ \varepsilon \\ either\ 0\ or\ 1\ randomly: with\ probability\ 1-\varepsilon \end{cases}$
  - **Step 3: At the end of the current interval, observe the system reward r in the interval and the state in next interval k_R'**
  - **Step 4: Update the Q-value**
  - **Step 5: Set $k_R = k_R'$, enter the next interval, and return to Step 2.**

PROBLEM FORMULATION

- **Caching Policy with Multiple Files**
  - **Main challenge : curse of dimensionality**
    - **To reduce the complexity, set upper bound of the reward(reward의 상한선 설정)**
  - **Impacts of caching actions on long-term reward for different files are mutually "independent"**
    - **There is a constraint number of cached files cannot exceed the cache size.**
  - **we introduce the optimistic assumption that constraint is always satisfied in future**
  - **Upper bound of the reward denoted by $V^{\pi,ub}(k_R)$**
    - $V^{\pi,}(k_R) \leq V^{\pi,ub}(k_R)$

PROBLEM FORMULATION

- **Caching Policy with Multiple Files**
  - **All consider, caching policy is** $\max_{\pi} V^{\pi,ub}(k_R)$
    - **Constraint 1 :** $\sum_{i=1}^{N_f} a_{t,i} \leq N_c$ **(number of cached files cannot exceed the cache size 만족)**
    - **Constraint 2 :** $a_{t,i} \in \{0, 1\}$
  - **결과 : dimension reduce**
    - M^{N_f} **to** MN_f **(M : the number of possible states for one file.)**

- **State(k_R)이 주어졌을 때, 가능한 state-action 에 대해 Q-value를 확인함으로써 file의 cache 여부를 결정할 수 있다.**
  - **Expected long-term rewards of different files are irrelevant = can be obtained separately**
  - **Q-value is the estimate of the maximal long-term reward for a file, given a state-action pair.**

PROBLEM FORMULATION

- **Algorithm 2 Q-Learning based Caching Policy with Multiple Files**
  - **Initialize the Q-value for each state-action pair.**
  - **Step 1: At the beginning of current interval, count the number of remaining users for each file kR.**
  - **Step 2: Generate a random number $l \in (0, 1)$, if $l > \varepsilon$, choose the cached files according to (13), (14) and (15), otherwise, randomly choose Nc files to cache.**
  - **Step 3: At the end of current interval, compute the system reward for each file in this interval $\{ri\}$ and the states in the next interval k$'R$.**
  - **Step 4: Update the Q-value for each file according to (10).**
  - **Step 5: Set kR = k$'R$, enter the next interval, and return to Step 2.**

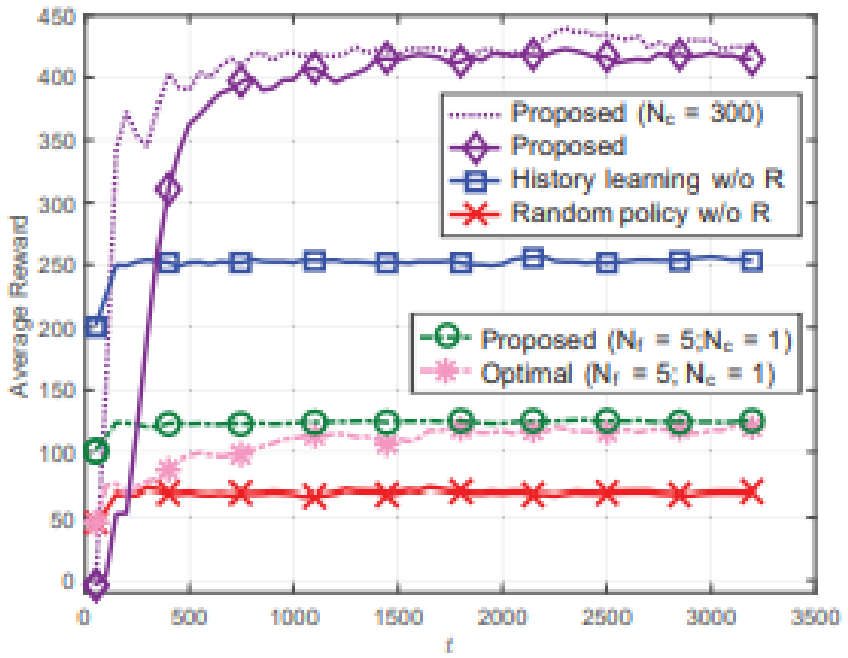
SIMULATION RESULTS

SIMULATION RESULTS

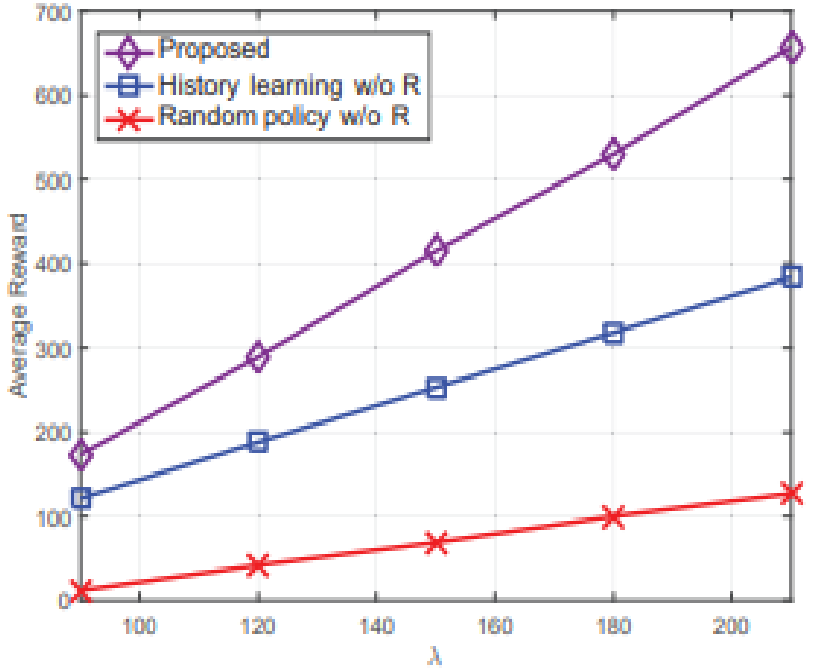
- In the simulation, the content catalog contains N_f = 300 files, where N_c = 30 files can be cached at the BS.
  - The users depart the cell with probability p_D = 0.1.
  - request probability p_i, $i = 1, \cdots, N_f$, which is controlled by the Zipf popularity distribution
  - $\begin{cases} \gamma_1 = 0.8\,, & if\ recommended \\ \gamma_0 = 0.4\,, & if\ not\ recommeded \end{cases}$

- 비교
  - Random caching policy without recommendation : 캐싱 작업을 수행하기 위해 어떠한 정보도 이용하지 않음
  - History learning without recommendation : Q-learning 학습 정책과 동일한 원칙을 공유.
    - 파일의 cached 여부에 관계없이 각 파일에 대한 요청 수를 관찰할 수 있음
    - Q-learning with recommedation은 cached file에 대한 요청 수만 관찰 가능 (차이점)
    - 따라서 학습 수렴 속도가 훨씬 빠름.

SIMULATION RESULTS



Y : Average system reward
X : number of intervals(t)



Y : Average system reward
X : average user arrival rate

Random policy < Q-learning policy < Q-learning policy with recommendation
임을 확인할 수 있다. (reward가 높음)

CONCLUSIONS

CONCLUSION

- Introduce “recommendation”
  - 더 좋은 결과(traffic 해결)

- Developed a Q-learning based algorithm
  - unknown statistics of random user arrival, user departure, and user request probability 에 대해 학습할 수 있는 일련의 solution 제공

Thank you