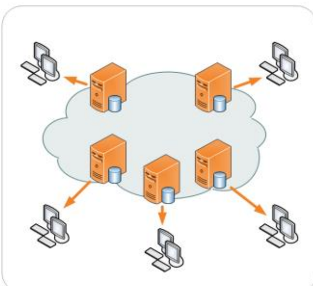
Fair Caching Networks

Yuezhou Liu[a], Yuanyuan Li[a], Qian Ma[b],
Stratis Ioannidis[a], Edmund Yeh[a]

[a]Northeastern University

[b]Sun Yat-sen University

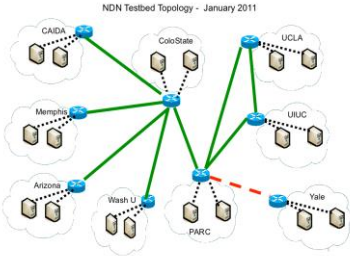
Motivation



CDN

Cloud Computing

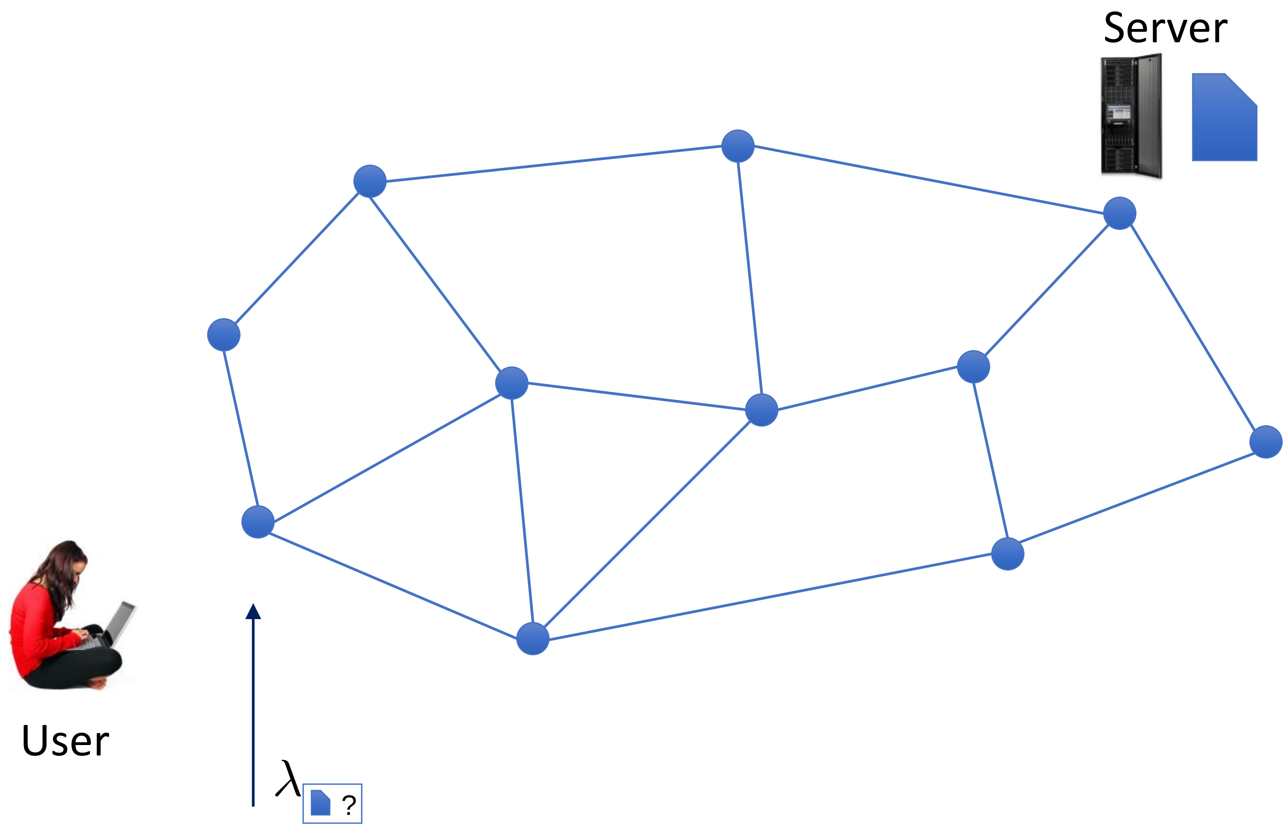
Internet of Things



Content-Centric Networks

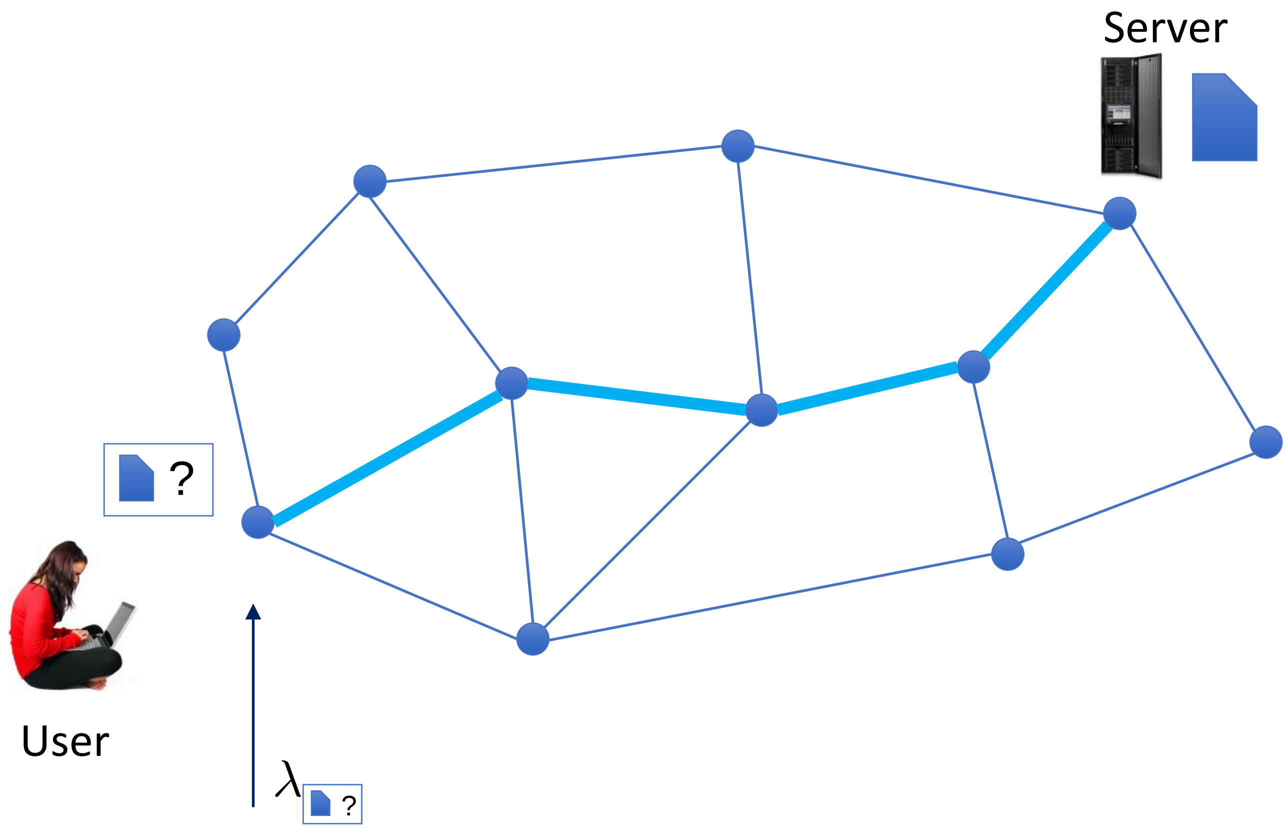
Caching and resource allocation problems are ubiquitous

A Cache Network



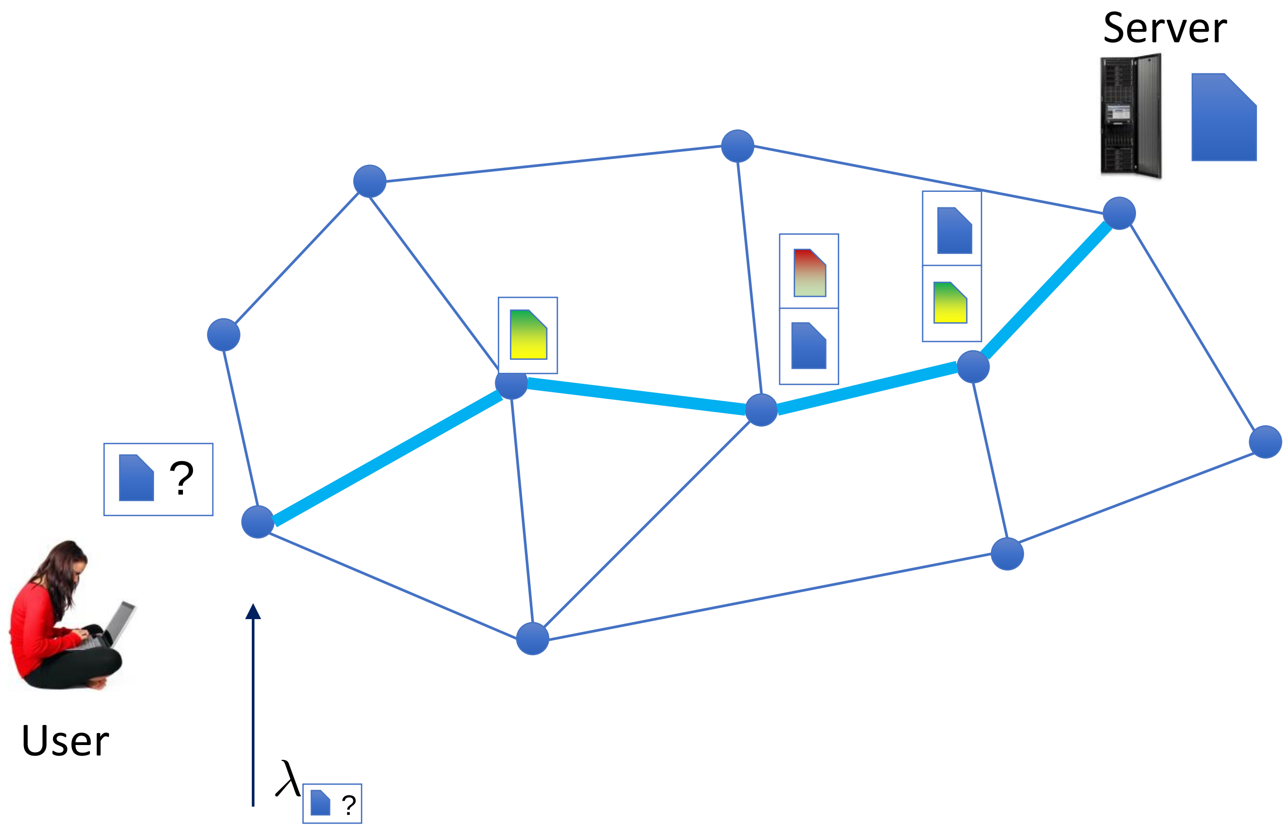
User nodes generate **requests** for content items with certain request rates

A Cache Network



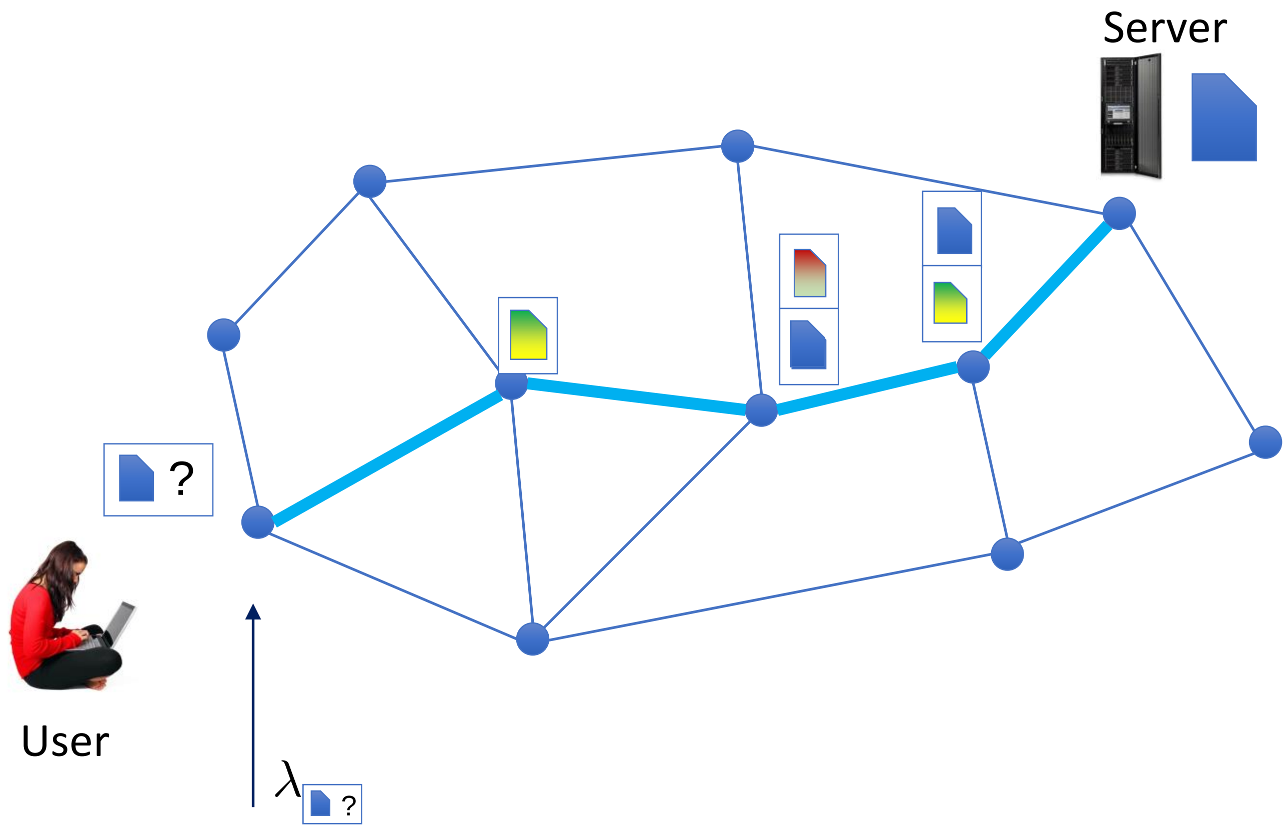
Requests are **routed** towards a **designated server**

A Cache Network



Nodes have **caches** with finite capacities

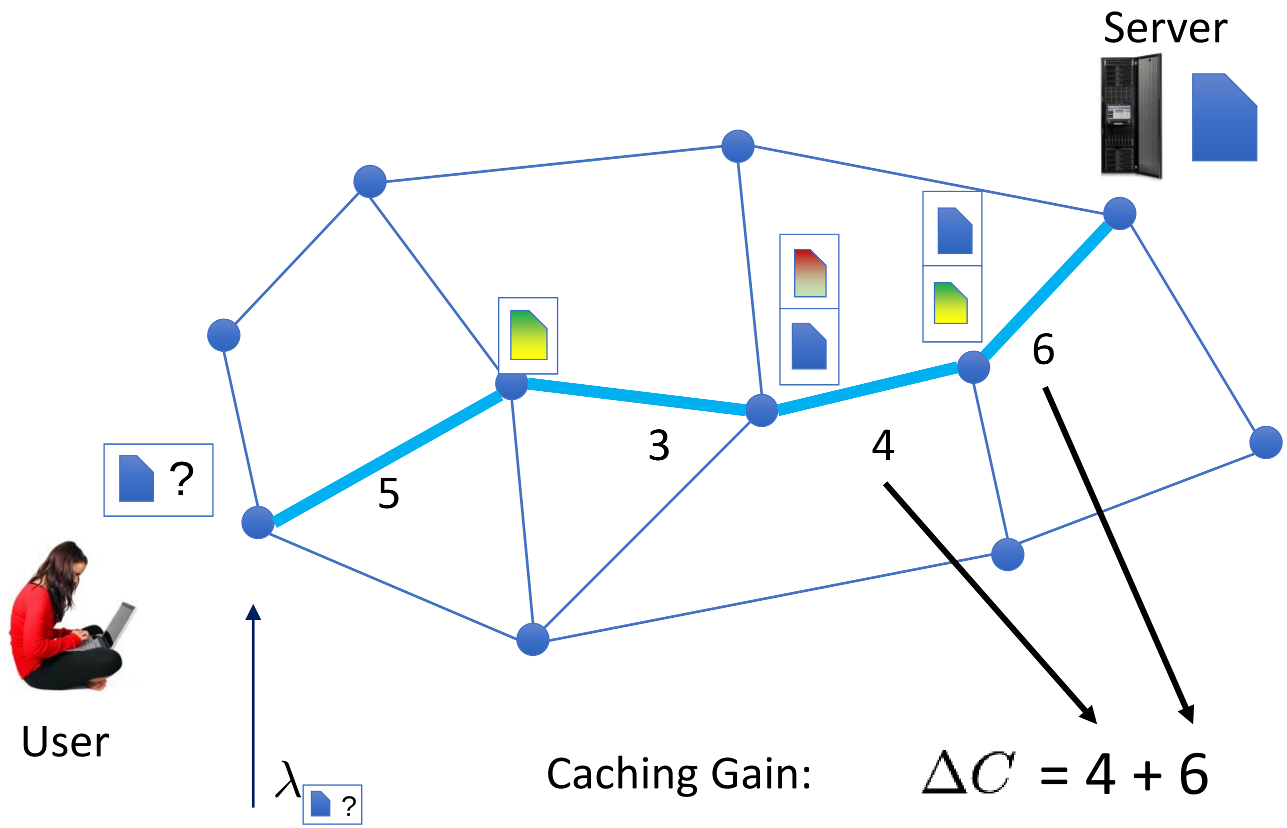
A Cache Network



Requests terminate early upon a **cache hit**

Content is sent back over **reverse** path

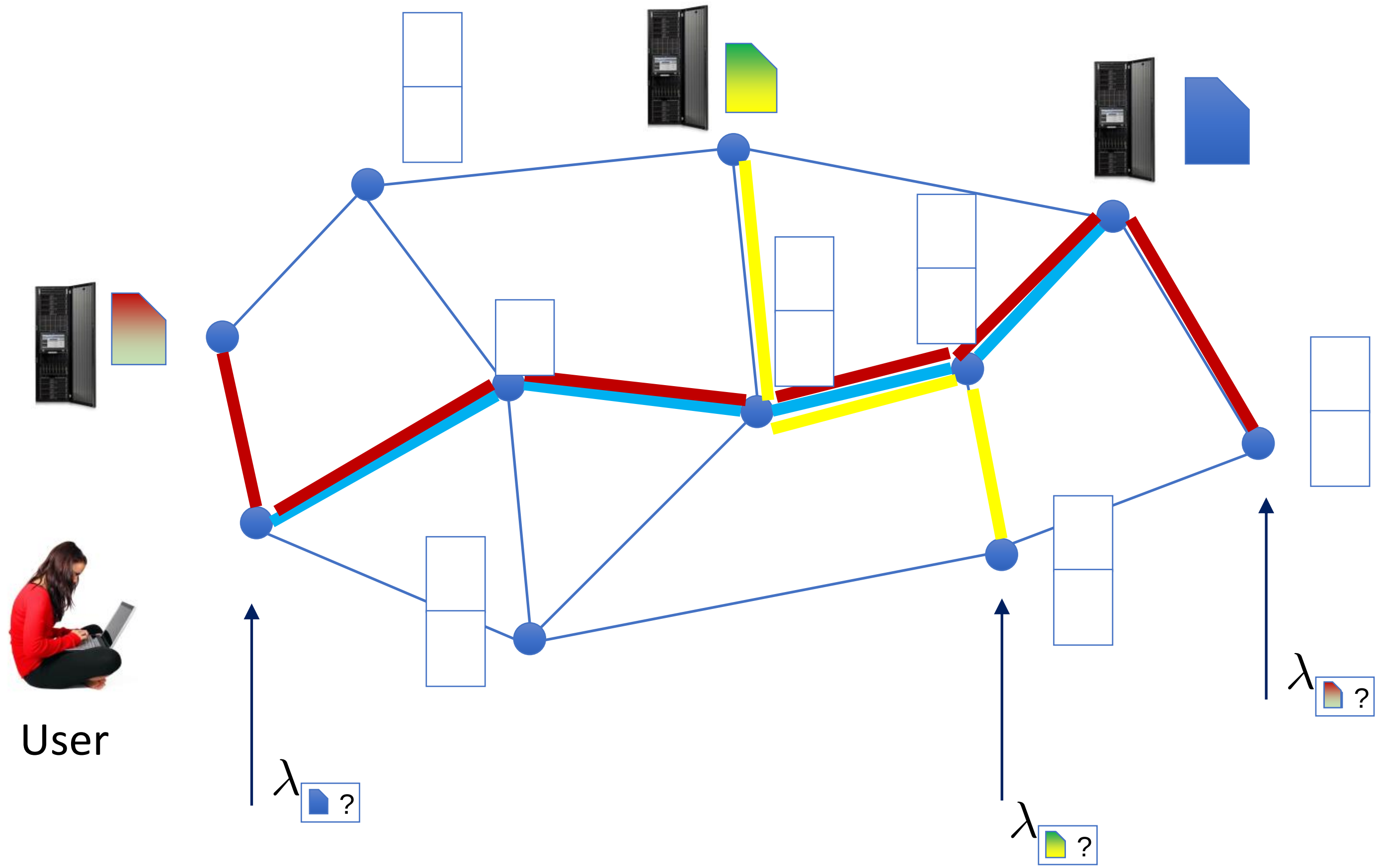
A Cache Network



Caching Gain: ΔC = 4 + 6

Caching Gain Rate: $\lambda \cdot \Delta C$

Optimal Content Allocation



Q: How should items be allocated to caches?

Existing Works: Caching Gain Rate

$$\sum \lambda \cdot \Delta C$$

Several papers study the maximization of the overall caching gain rate

- Caching in general networks [Ioannidis and Yeh 2016]
- Caching in resilient networks [Li et al. 2018]
- Joint routing and caching [Ioannidis and Yeh 2018]

Advantages:

- It captures popularities and **routing cost**
  - In a **general cache network**, algorithms that ignore routing cost can be arbitrarily **suboptimal** [Ioannidis and Yeh 2016]
- The objective function is **submodular** [Shanmugam et al. 2013]
  - Approximation algorithms exist

This work: Instead of simply maximizing the overall caching gain rate, we take **fairness** into consideration.

Existing Works: Fair Caching

- Either specific fairness notions (e.g. proportional fairness) or **α-fair utility functions**
- They consider different utilities/objectives.
- Hit ratio, e.g. [Dehghan et al. 2016], [Panigraphy et al. 2017], [Chu et al. 2017]
- Storage and fetching cost [Wang et al. 2016], video quality [Avrachenkov et al. 2019], throughput [Bonald et al. 2017], delay [Rezvani et al. 2019]
- They do not capture the multi-hop routing cost. Can be suboptimal in a general cache network
- EX: Requests served with hit ratio 1 at a distant server, in reality, have a lower utility than requests served locally with a lower hit ratio.

This work: To study the fair caching problem in a general cache network, we consider the utility of **caching gain rate**.

Contributions

- Formal statement of the fair caching network model
  - NP-Hard
- Maximizing submodular objective under matroid constraints
  - Greedy Algorithm, 1/2 approximation factor
  - Continuous Greedy, 1-1/e approximation factor
- Stationary Randomized strategy
  - L-method, $(1-1/e)^{1-\alpha}$ approximation factor
- Evaluations
  - Performance under synthetic and real-world network topologies
  - Analysis for the effect of fairness

Overview

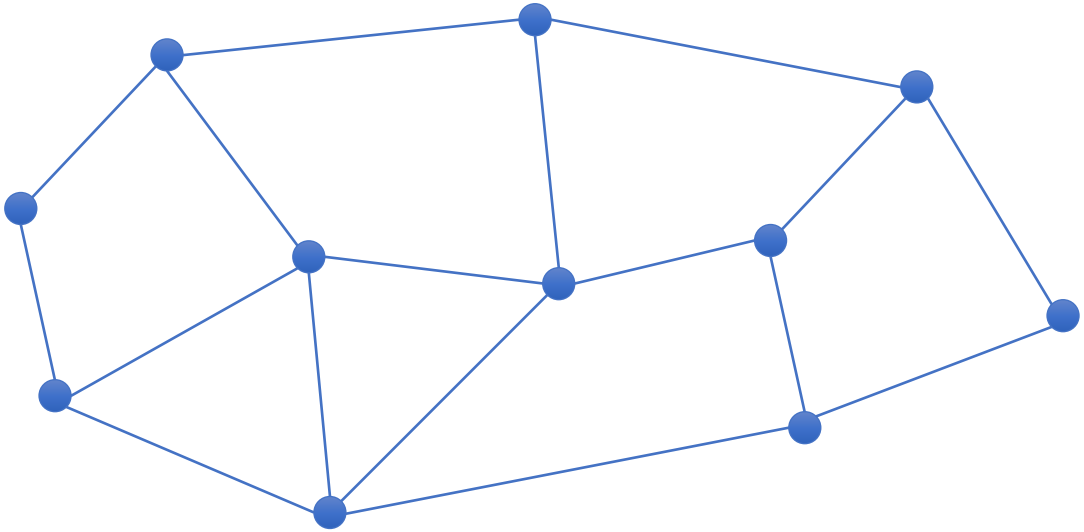
- Problem Formulation
- Deterministic Strategy
- Stationary Randomized Strategy
- Evaluation

Overview

- ❑Problem Formulation
- ❑Deterministic Strategy
- ❑Stationary Randomized Strategy
- ❑Evaluation

Model: Network

$G(V, E)$

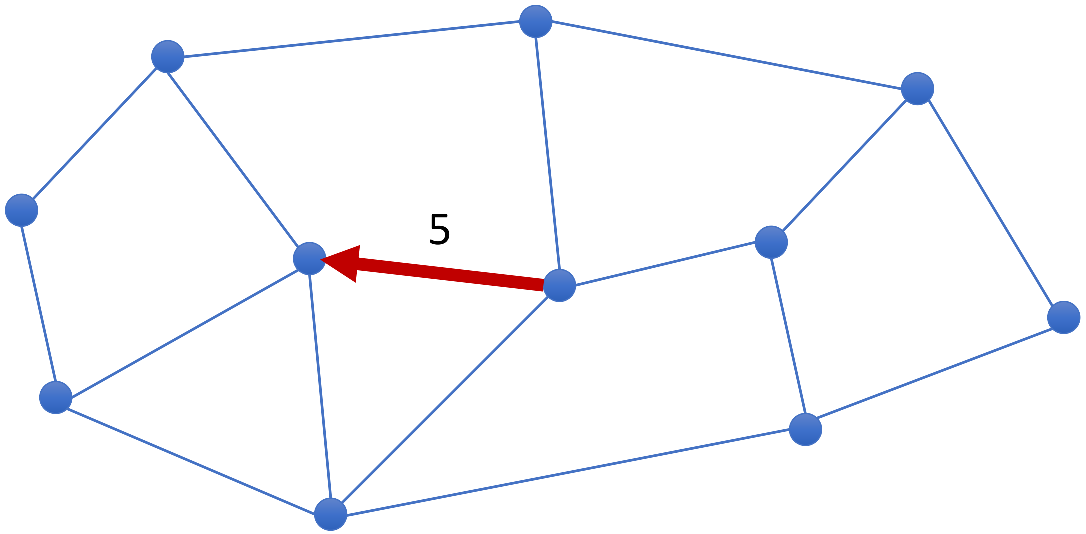


Network represented as a **directed**, **bi-directional** graph $G(V, E)$

Model: Edge Costs

$G(V, E)$

Edge costs: $w_{uv}, (u, v) \in E$



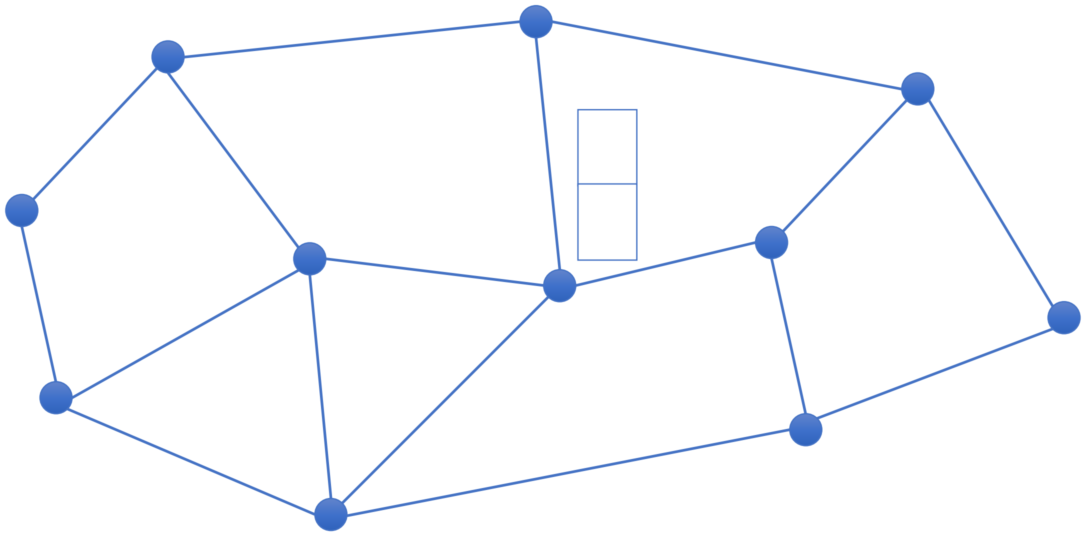
Each edge $(u, v) \in E$ has a **cost/weight** w_{uv}

Model: Node Caches

$G(V, E)$

Edge costs: $w_{uv}, (u,v) \in E$

Node capacities: $c_v, v \in V$



Node $v \in V$ has a cache with capacity $c_v \in \mathbb{N}$

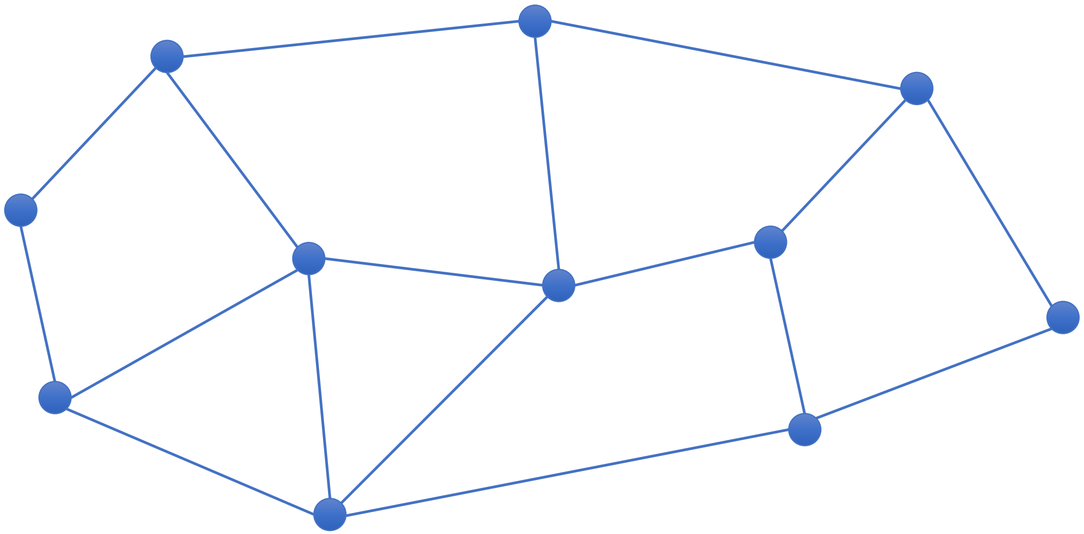
Model: Cache Contents

$G(V, E)$

$\mathcal{C} = \{$  $\}$

Edge costs: $w_{uv}, (u, v) \in E$

Node capacities: $c_v, v \in V$



$\mathcal{C}$: the **catalog** of equally sized contents

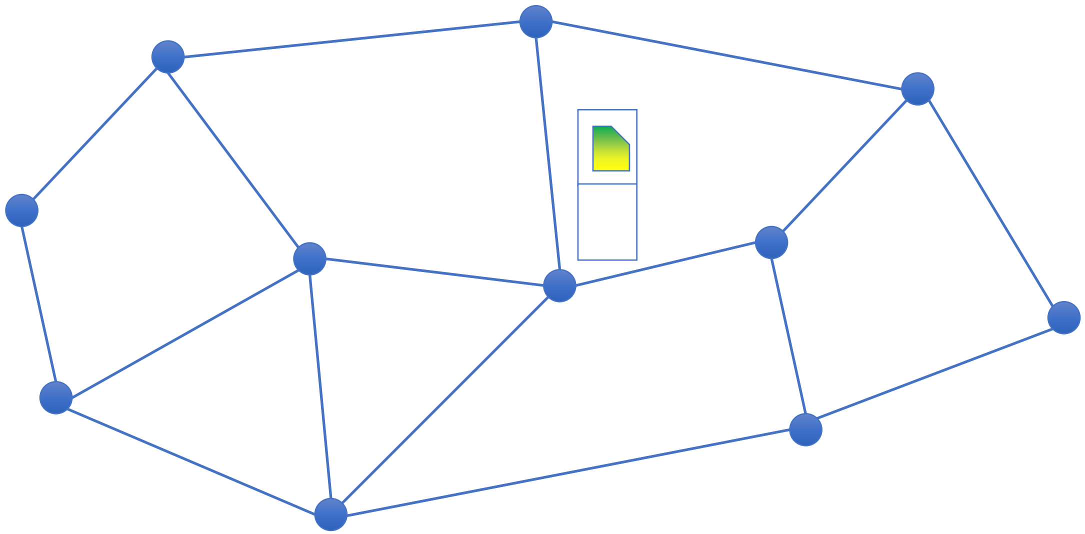
Model: Cache Contents

$G(V, E)$

$\mathcal{C} = \{$  $\}$

Edge costs: $w_{uv}, (u, v) \in E$

Node capacities: $c_v, v \in V$



For $v \in V$ and $i \in \mathcal{C}$, let

$$x_{vi} = \begin{cases} 1, & \text{if } v \text{ stores } i \\ 0, & \text{o.w.} \end{cases}$$

Then, for all $v \in V$, $\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$

Model: Cache Contents

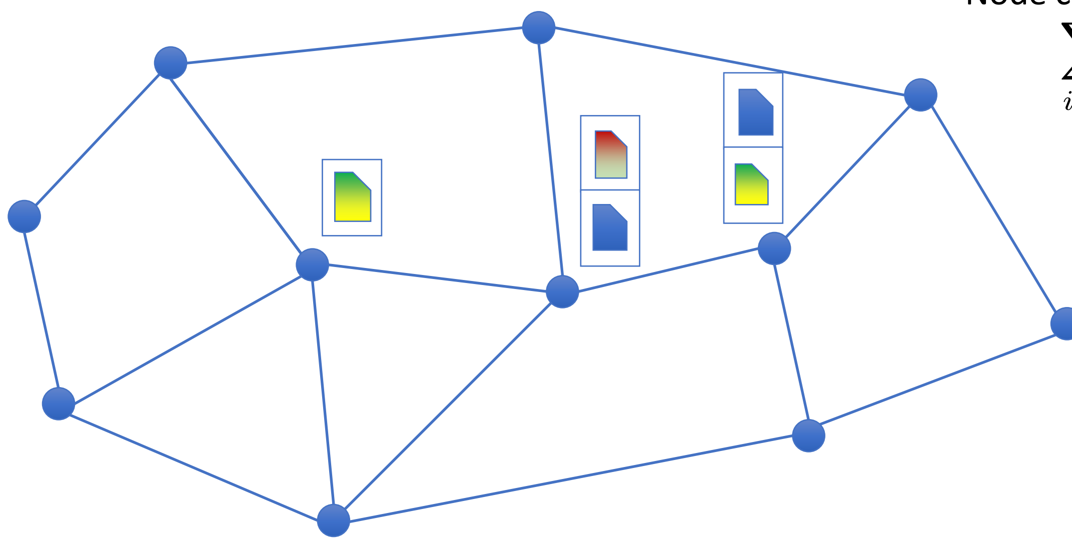
$G(V, E)$

$\mathcal{C} = \{$  $\}$

Edge costs: $w_{uv}, (u, v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$



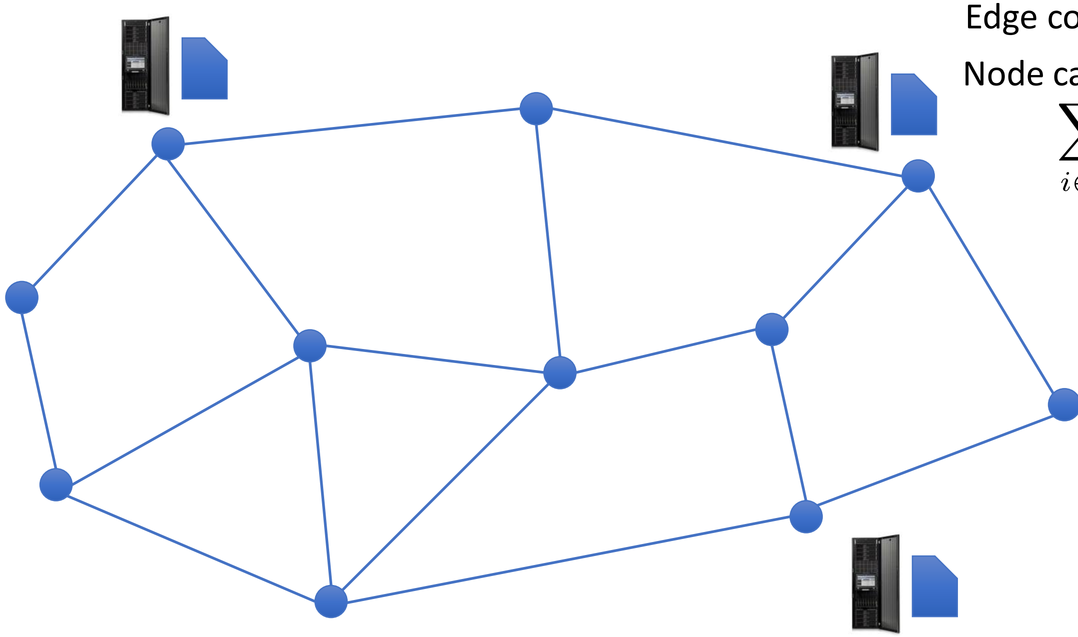
The global **allocation strategy** is the binary $|V| \times |\mathcal{C}|$ matrix

$$X = [x_{vi}]_{v \in V, i \in \mathcal{C}}$$

Model: Designated Servers

$G(V,E)$

$\mathcal{C} = \{$  $\}$



Edge costs: $w_{uv}, (u,v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

For each $i \in \mathcal{C}$, there exists a set of nodes $S_i \subset V$ (the **designated servers** of i) that **permanently store** i .

Model: Demand

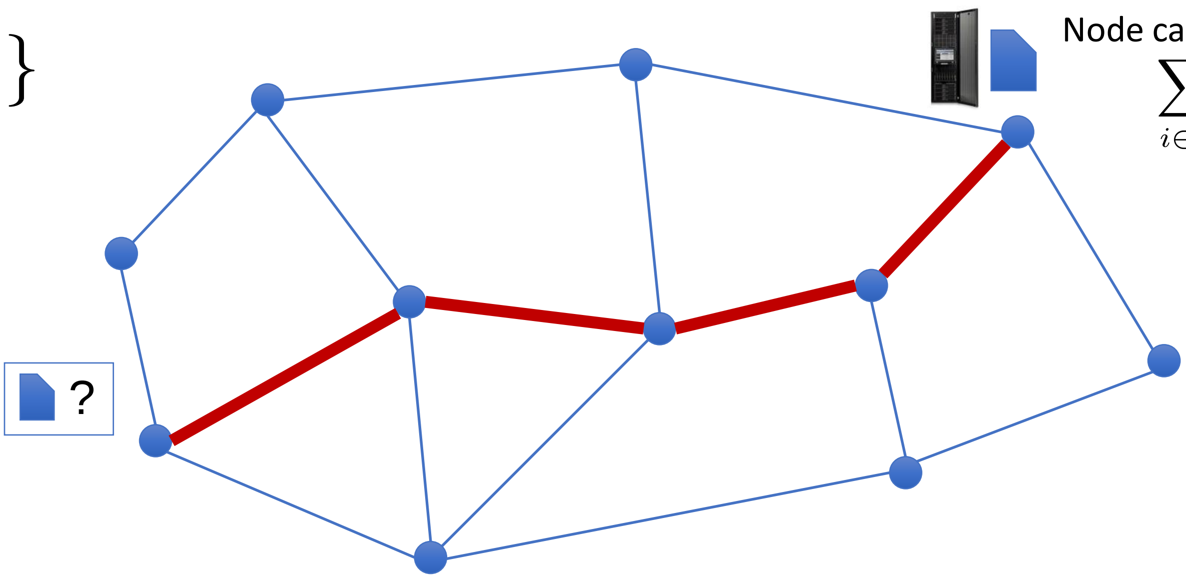
$G(V, E)$

$\mathcal{C} = \{$  $\}$

Edge costs: $w_{uv}, (u, v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$



Requests are always satisfied!

A **request** is a pair (i, p) such that:

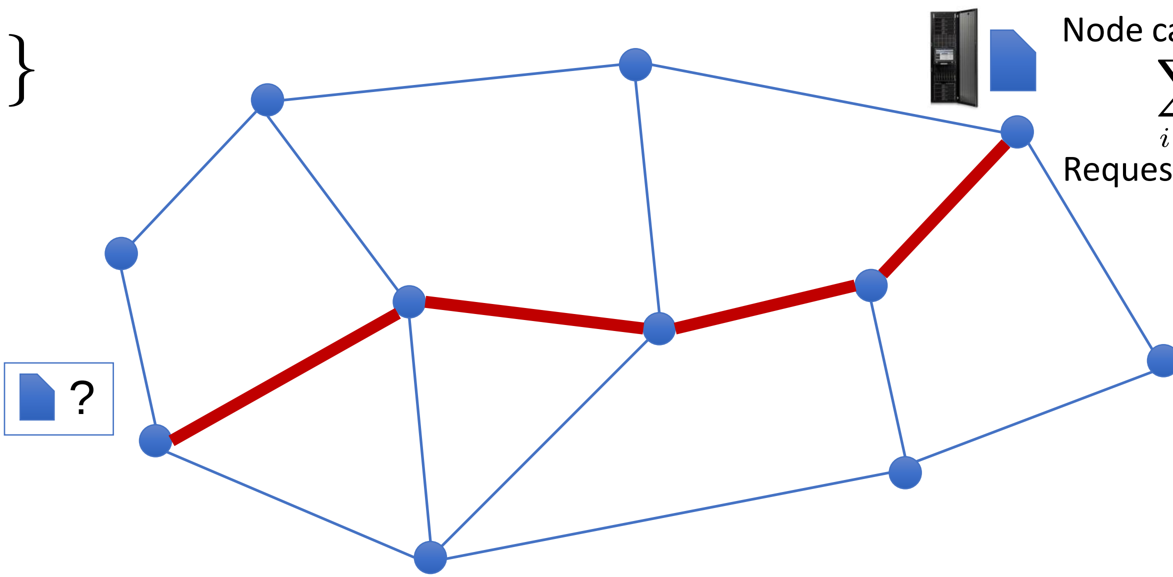
- i is an item in $\mathcal{C}$
- $p = \{p_1, \ldots, p_K\}$ is a simple path in G such that $p_K \in S_i$.

Model: Demand

$G(V, E)$

$\mathcal{C} = \{$  $\}$

$\mathcal{R}$: demand



Edge costs: $w_{uv}, (u, v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

Request rates: $\lambda_{(i,p)}, (i, p) \in \mathcal{R}$

Demand $\mathcal{R}$: set of all requests (i, p)

The request rate of each request is $\lambda_{(i,p)}$ (number of requests per unit of time)

Model: Routing Costs & Caching Gain

$G(V, E)$

$\mathcal{C} = \{ \quad \}$

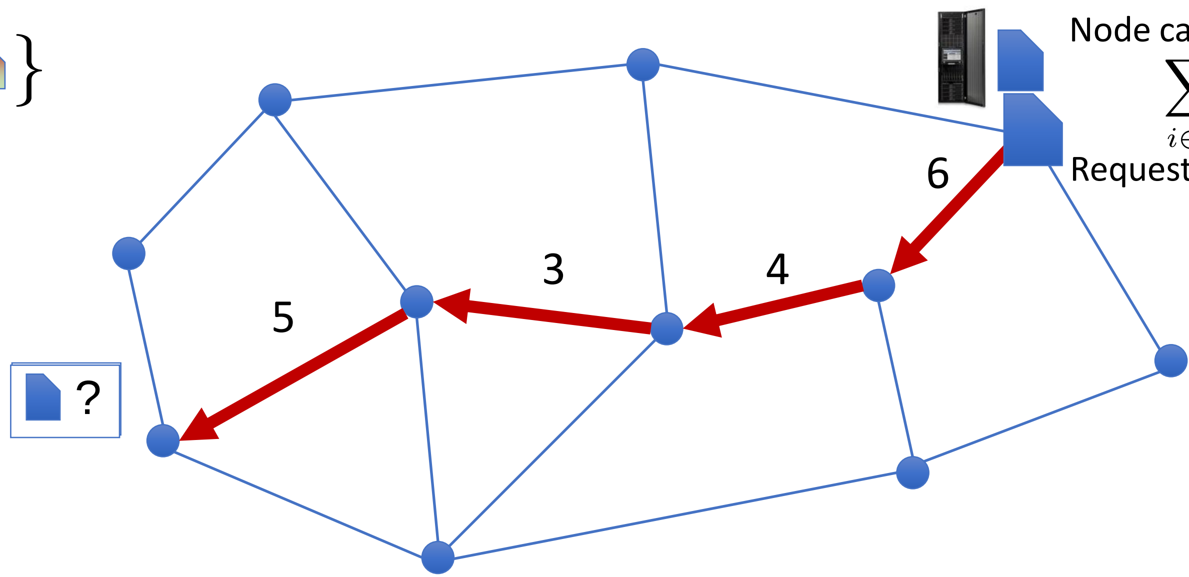
$\mathcal{R}$: demand

Edge costs: $w_{uv}, (u,v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

Request rates: $\lambda_{(i,p)}, (i,p) \in \mathcal{R}$



Request
(i, p)

Worst case routing cost: $\sum_{k=1}^{|p|-1} w_{p_{k+1}p_k}$

Model: Routing Costs & Caching Gain

$G(V, E)$

$\mathcal{C} = \{$  $\}$

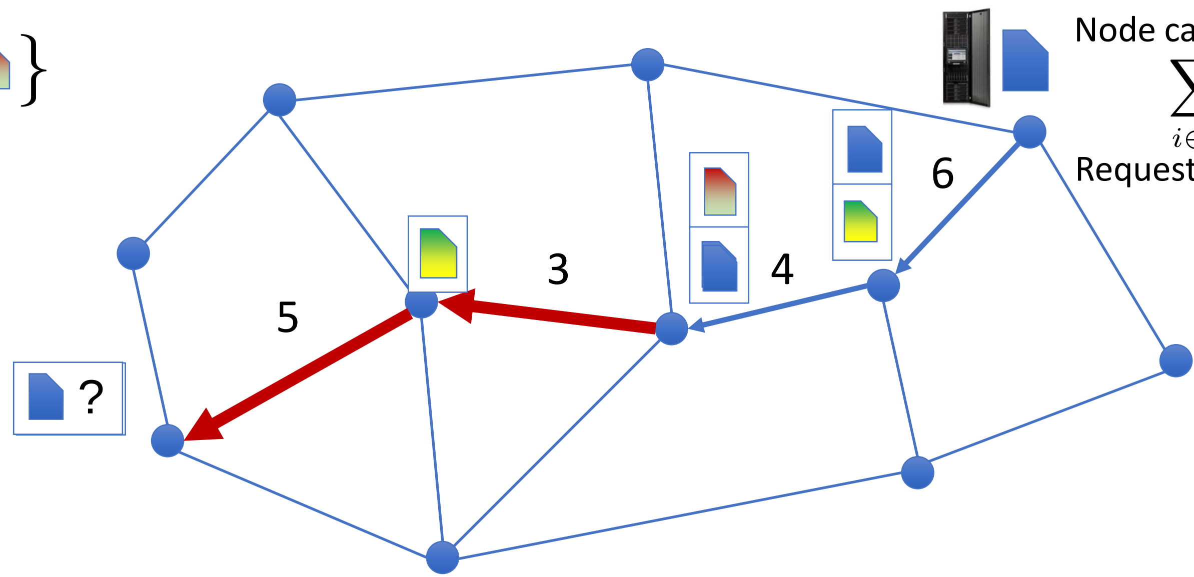
$\mathcal{R}$: demand

Edge costs: $w_{uv}, (u, v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

Request rates: $\lambda_{(i,p)}, (i,p) \in \mathcal{R}$



Request
(i, p)

Worst case routing cost: $\sum_{k=1}^{|p|-1} w_{p_{k+1} p_k}$

Cost due to intermediate caching: $\sum_{k=1}^{|p|-1} w_{p_{k+1} p_k} \prod_{k'=1}^{k} (1 - x_{p_{k'} i})$

Model: Routing Costs & Caching Gain

$G(V, E)$

$\mathcal{C} = \{$  $\}$

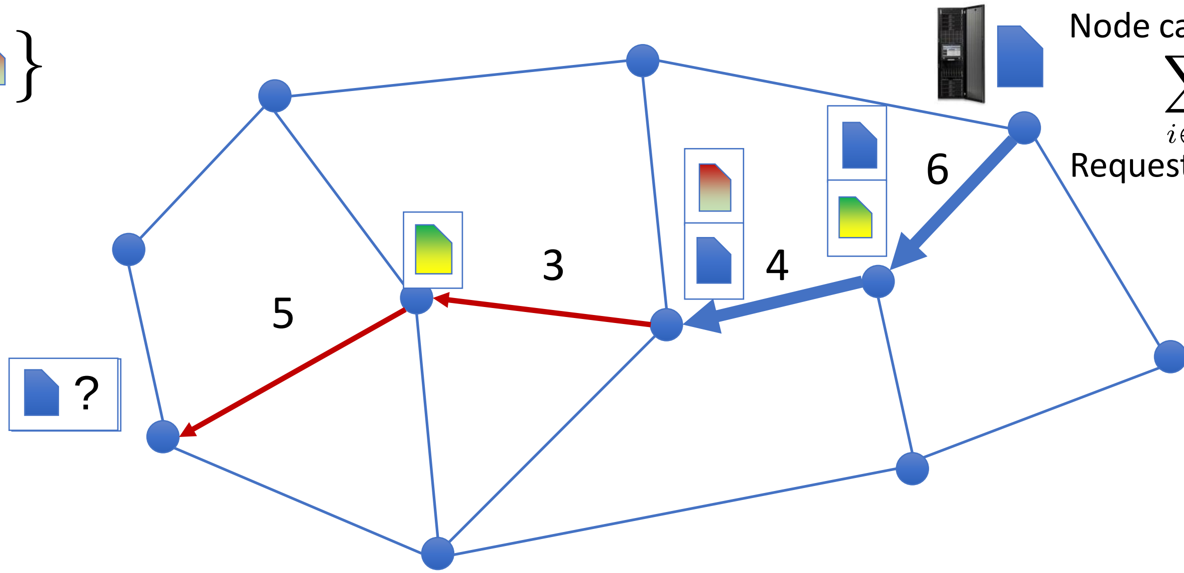
$\mathcal{R}$: demand

Edge costs: $w_{uv}, (u, v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

Request rates: $\lambda_{(i,p)}, (i, p) \in \mathcal{R}$



Request
(i, p)

Worst case routing cost: $\sum_{k=1}^{|p|-1} w_{p_{k+1} p_k}$

Cost due to intermediate caching: $\sum_{k=1}^{|p|-1} w_{p_{k+1} p_k} \prod_{k'=1}^{k} (1 - x_{p_{k'} i})$

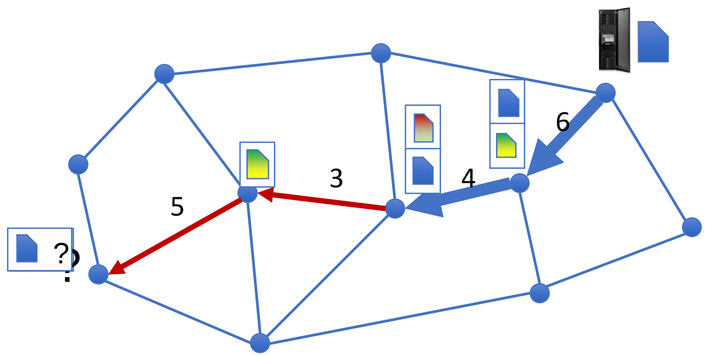
Caching Gain: $F_{(i,p)}(X) = \sum_{k=1}^{|p|-1} w_{p_{k+1} p_k} \left(1 - \prod_{k'=1}^{k} (1 - x_{p_{k'} i})\right)$

Utility Maximization

$G(V,E)$

$\mathcal{C} = \{\ \}$

$\mathcal{R}$: demand



Edge costs: $w_{uv}, (u,v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

Request rates: $\lambda_{(i,p)}, (i,p) \in \mathcal{R}$

Caching Gain:

$$\sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \left(1 - \prod_{k'=1}^{k} (1 - x_{p_{k'}i})\right)$$

$$U(z) = \begin{cases} \dfrac{z^{1-\alpha}}{1-\alpha} & \text{when } 0 \leq \alpha < 1, \\ \log(z+\epsilon) & \text{when } \alpha = 1, \text{ or} \\ \dfrac{(z+\epsilon)^{1-\alpha}}{1-\alpha} & \text{when } \alpha > 1, \end{cases}$$

α-fair utility functions [Mo et al. TON 2000]

α fairness

α=0, no fairness

α=1, proportional fairness

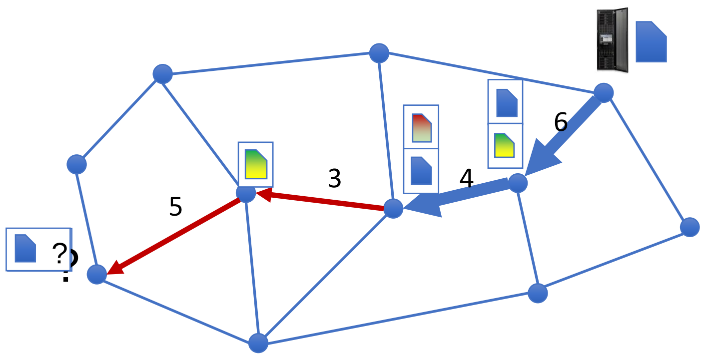
α→∞, max-min fairness

Utility Maximization

$G(V, E)$

$\mathcal{C} = \{$  $\}$

$\mathcal{R}$: demand



Edge costs: $w_{uv}, (u, v) \in E$

Node capacities: $c_v, v \in V$

$$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v, \text{ for all } v \in V$$

Request rates: $\lambda_{(i,p)}, (i, p) \in \mathcal{R}$

Caching Gain:

$$\sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \left(1 - \prod_{k'=1}^{k} (1 - x_{p_{k'}i})\right)$$

Utility of requests:

$$U(\lambda_{(i,p)} F_{(i,p)}(X)), \ (i, p) \in \mathcal{R}$$

Caching gain rate: $\lambda_{(i,p)} \cdot F_{(i,p)}(X), \quad (i, p) \in \mathcal{R}$

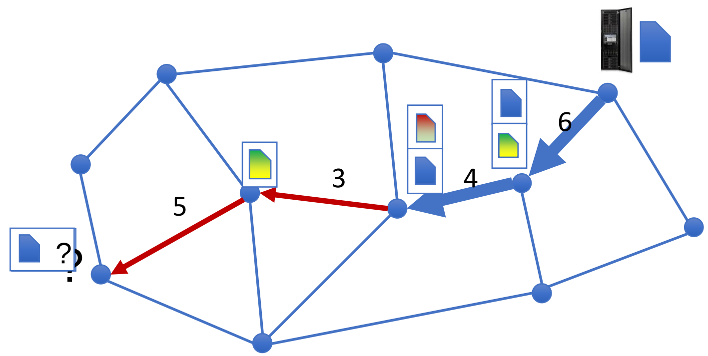
Utility of requests: $U(\lambda_{(i,p)} F_{(i,p)}(X)), \ (i, p) \in \mathcal{R}$

Utility Maximization

$G(V, E)$

$\mathcal{C} = \{$  $\}$

$\mathcal{R}$: demand



Edge costs: $w_{uv}, (u,v) \in E$

Node capacities: $c_v, v \in V$

$\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

Request rates: $\lambda_{(i,p)}, (i,p) \in \mathcal{R}$

Caching Gain:

$\sum_{k=1}^{|p|-1} w_{p_{k+1}p_k}\left(1 - \prod_{k'=1}^{k}(1 - x_{p_{k'}i})\right)$

Utility of requests:

$U(\lambda_{(i,p)} F_{(i,p)}(X)),\ (i,p) \in \mathcal{R}$

Maximize: $$G(X) = \sum_{(i,p) \in \mathcal{R}} U(\lambda_{(i,p)} F_{(i,p)}(X))$$

Subject to: $\sum_{i \in \mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

$x_{vi} \in \{0, 1\}$, for all $v \in V$ and $i \in \mathcal{C}$

$$X \in \mathcal{D}_1$$

NP hard, when α = 0, [Shanmugam et al. IT 2013]

Overview

- ❑Problem Formulation
- ❑Deterministic Strategy
- ❑Stationary Randomized Strategy
- ❑Evaluation

Submodularity

$$\begin{aligned} \text{Maximize:} \quad & G(X) = \sum_{(i,p)\in\mathcal{R}} U(\lambda_{(i,p)} F_{(i,p)}(X)) \\ \text{s.t.} \quad & \sum_{i\in\mathcal{C}} x_{vi} \le c_v \quad \text{for all } v \in V, \\ & x_{vi} \in \{0,1\} \quad \text{for all } v \in V,\ i \in \mathcal{C}. \end{aligned}$$

$$U(z) = \begin{cases} \dfrac{z^{1-\alpha}}{1-\alpha} & \text{when } 0 \le \alpha < 1, \\ \log(z+\epsilon) & \text{when } \alpha = 1, \text{ or} \\ \dfrac{(z+\epsilon)^{1-\alpha}}{1-\alpha} & \text{when } \alpha > 1, \end{cases}$$

Thm:
For all α-fair utility functions, the utility maximization problem is submodular maximization under matroid constraints

Two polynomial approximation algorithms:

Greedy algorithm produces a solution within 1/2 approximation factor [Calinescu et al. 2007].

Continuous greedy algorithm produces a solution within 1-1/e approximation factor [Calinescu et al. 2011].

Continuous-Greedy Algorithm

Continuous greedy algorithm produces a solution within 1-1/e approximation factor [Calinescu et al. 2011].

Submodular maximization

$$
\begin{aligned}
\text{Maximize:} \quad & G(X) = \sum_{(i,p)\in\mathcal{R}} U(\lambda_{(i,p)} F_{(i,p)}(X)) \\
\text{s.t.} \quad & \sum_{i\in\mathcal{C}} x_{vi} \le c_v \quad \text{for all } v \in V, \\
& x_{vi} \in \{0,1\} \quad \text{for all } v \in V,\ i \in \mathcal{C}.
\end{aligned}
$$

Continuous-Greedy Algorithm

Continuous greedy algorithm produces a solution within 1-1/e approximation factor [Calinescu et al. 2011].

$$\text{Submodular maximization} \xrightarrow{\text{Multilinear Extension}} \text{Non-convex maximization}$$

$$
\begin{aligned}
\text{Maximize:} \quad & K(Y) = \mathbb{E}_{\mu}[G(X)] \\
\text{s.t.} \quad & \sum_{i \in \mathcal{C}} y_{vi} \leq c_v \quad \text{for all } v \in V, \\
& y_{vi} \in [0,1] \quad \text{for all } v \in V,\ i \in \mathcal{C}.
\end{aligned}
$$

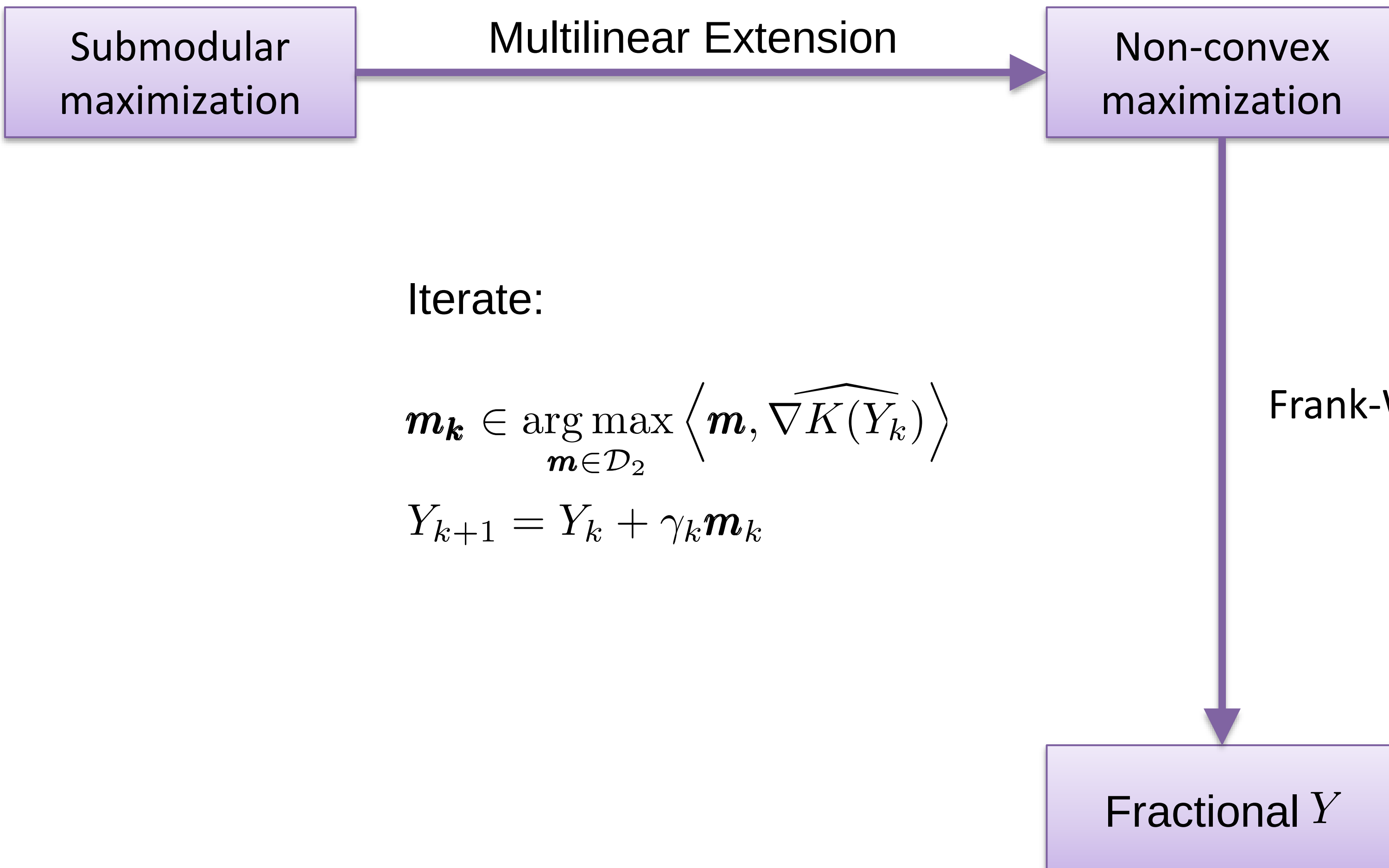
We consider x_{vi} as random variables with joint distribution:

$$\mu(X) = \prod_{v \in V} \mu_v(x_{v1}, \ldots, x_{v|\mathcal{C}|})$$

$$\mathbb{E}_{\mu_v}[x_{vi}] = \mathbf{P}_{\mu_v}[x_{vi}] = y_{vi}, \text{ for } v \in V,\ i \in \mathcal{C}$$

Continuous-Greedy Algorithm

Continuous greedy algorithm produces a solution within 1-1/e approximation factor [Calinescu et al. 2011].



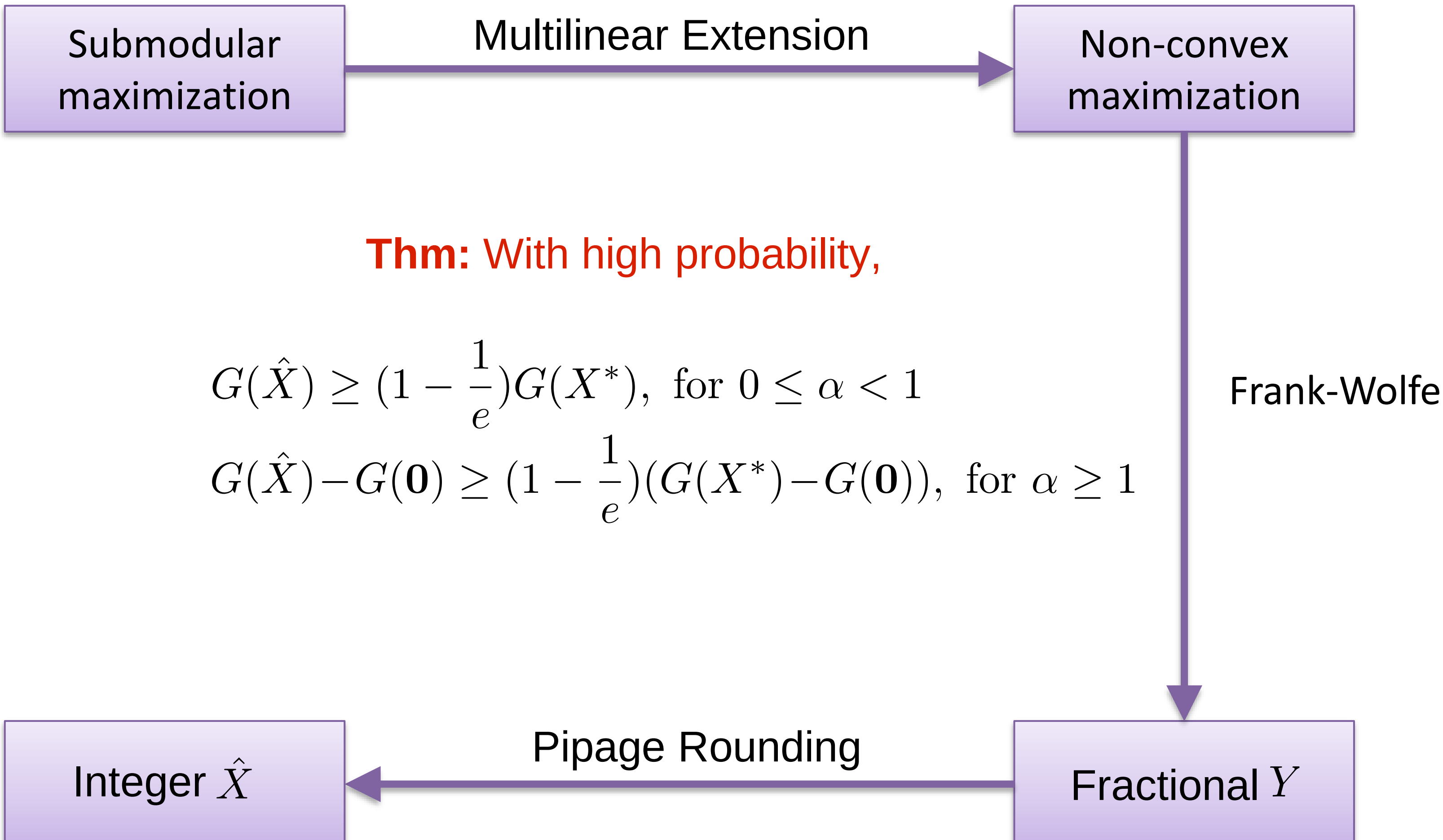
Iterate:

$$\boldsymbol{m_k} \in \arg\max_{\boldsymbol{m} \in \mathcal{D}_2} \left\langle \boldsymbol{m}, \nabla \widehat{K(Y_k)} \right\rangle$$

$$Y_{k+1} = Y_k + \gamma_k \boldsymbol{m}_k$$

Continuous-Greedy Algorithm

Continuous greedy algorithm produces a solution within 1-1/e approximation factor [Calinescu et al. 2011].



Thm: With high probability,

$$G(\hat{X}) \geq (1 - \frac{1}{e})G(X^*), \text{ for } 0 \leq \alpha < 1$$

$$G(\hat{X}) - G(\mathbf{0}) \geq (1 - \frac{1}{e})(G(X^*) - G(\mathbf{0})), \text{ for } \alpha \geq 1$$

Overview

- ❑Problem Formulation
- ❑Deterministic Strategy
- ❑Stationary Randomized Strategy
- ❑Evaluation

Stationary Randomized Strategy

- We consider a time-slotted system.
- At each time slot, a random caching strategy is sampled from a joint distribution over the feasible set:

$$\mu(X) = \prod_{v \in V} \mu_v(x_{v1}, \ldots, x_{v|\mathcal{C}|})$$

- Our problem becomes

$$\begin{aligned} \text{Maximize:} \quad & \sum_{(i,p) \in \mathcal{R}} U(\mathbb{E}_\mu[\lambda_{(i,p)} F_{(i,p)}(X)]) \\ \text{s.t.} \quad & \text{supp}(\mu) \in \mathcal{D}_1, \end{aligned}$$

Distribution

Thm:

There exists a polynomial method that produces a distribution which is within $(1 - 1/e)^{1-\alpha}$ from the optimal in expectation.

This approximation factor is better than 1-1/e for utility functions with $\alpha < 1$

L-method

Utility of expected caching gain rate

$$\text{Maximize:} \quad \sum_{(i,p)\in\mathcal{R}} U(\mathbb{E}_\mu[\lambda_{(i,p)} F_{(i,p)}(X)])$$

$$\text{s.t.} \quad \text{supp}(\mu) \in \mathcal{D}_1,$$

L-method

Optimal solution: Y*

Utility of expected caching gain rate $\xrightarrow{\text{Equivalence}}$ Maximizing total utility over Y

$$\mathbb{E}_{\mu_v}[x_{vi}] = \mathbf{P}_{\mu_v}[x_{vi}] = y_{vi}, \text{ for } v \in V, \ i \in \mathcal{C}$$

$$\begin{aligned} \text{Maximize:} \quad & G(Y) = \sum_{(i,p)\in\mathcal{R}} U(\lambda_{(i,p)} F_{(i,p)}(Y)) \\ \text{s.t.} \quad & \sum_{i\in\mathcal{C}} y_{vi} \leq c_v \quad \text{for all } v \in V, \\ & y_{vi} \in [0,1] \quad \text{for all } v \in V, \ i \in \mathcal{C}. \end{aligned}$$

L-method

Optimal solution: Y*

Utility of expected caching gain rate

Equivalence

Maximizing total utility over Y

$$F_{(i,p)}(Y) = \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \left(1 - \prod_{k'=1}^{k} (1 - y_{p_{k'}i})\right)$$

$$\Downarrow$$

Convex function

$$L_{(i,p)}(Y) = \sum_{k=1}^{|p|-1} w_{p_{k+1}p_k} \min\left\{1, \sum_{k'=1}^{k} y_{p_{k'}i}\right\}$$

Convex approximation

Fractional Y

L-method

Optimal solution: Y*

Utility of expected caching gain rate

Equivalence

Maximizing total utility over Y

$$
\begin{aligned}
\text{Maximize:}\quad & G(Y) = \sum_{(i,p)\in\mathcal{R}} U(\lambda_{(i,p)} F_{(i,p)}(Y)) \\
\text{s.t.}\quad & \sum_{i\in\mathcal{C}} y_{vi} \le c_v \quad \text{for all } v \in V, \\
& y_{vi} \in [0,1] \quad \text{for all } v \in V,\ i \in \mathcal{C}.
\end{aligned}
$$

$$\Downarrow$$

$$
\begin{aligned}
\text{Maximize:}\quad & H(Y) = \sum_{(i,p)\in\mathcal{R}} U(\lambda_{(i,p)} L_{(i,p)}(Y)) \\
\text{s.t.}\quad & \sum_{i\in\mathcal{C}} y_{vi} \le c_v \quad \text{for all } v \in V, \\
& y_{vi} \in [0,1] \quad \text{for all } v \in V,\ i \in \mathcal{C}.
\end{aligned}
$$

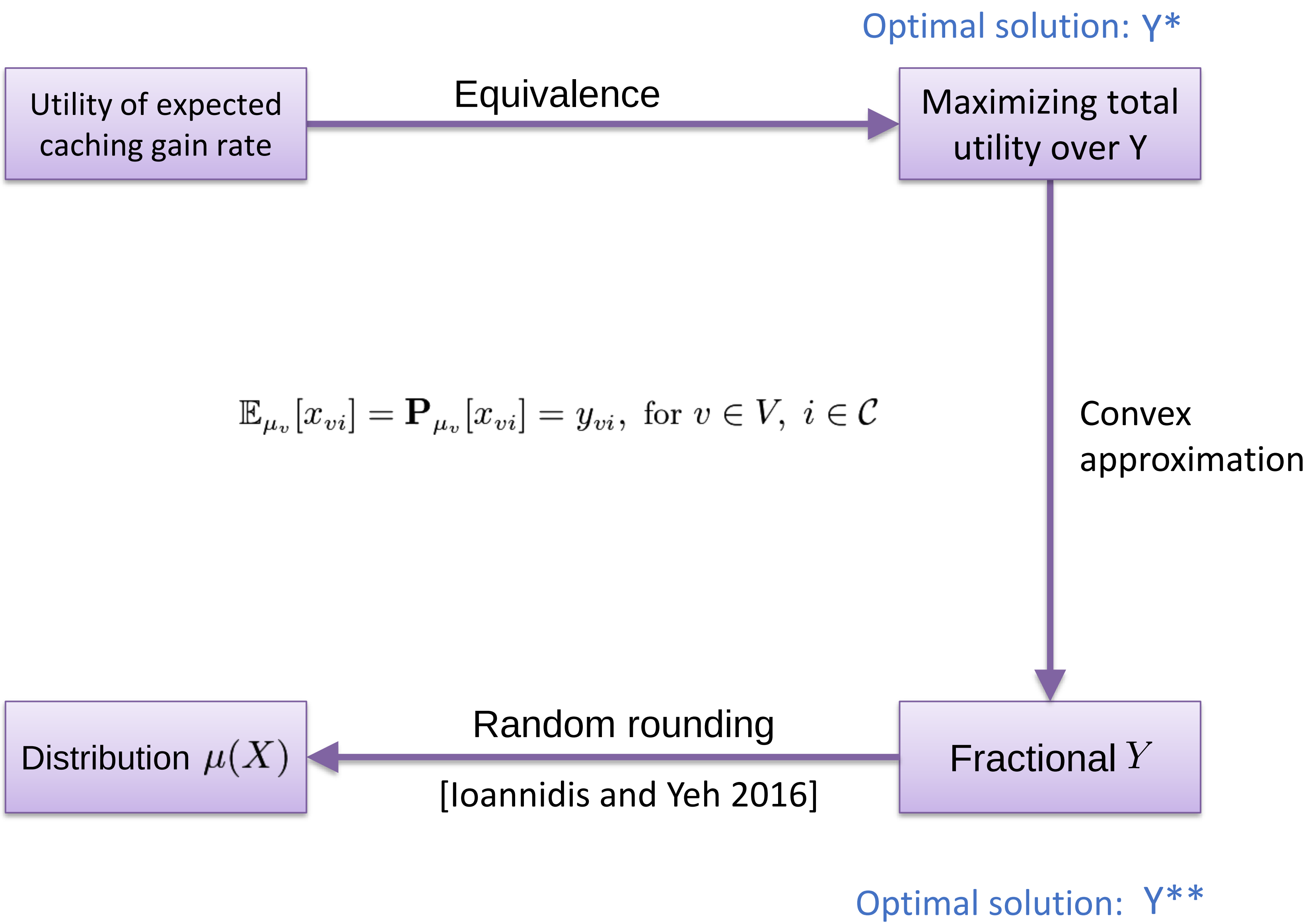
Convex approximation

Fractional Y

Convex Optimization

Optimal solution: Y**

L-method


Optimal solution: Y*
Utility of expected caching gain rate
Equivalence
Maximizing total utility over Y
$\mathbb{E}_{\mu_v}[x_{vi}] = \mathbf{P}_{\mu_v}[x_{vi}] = y_{vi}, \text{ for } v \in V, \ i \in \mathcal{C}$
Convex approximation
Distribution $\mu(X)$
Random rounding
[Ioannidis and Yeh 2016]
Fractional Y
Optimal solution: Y**

Overview

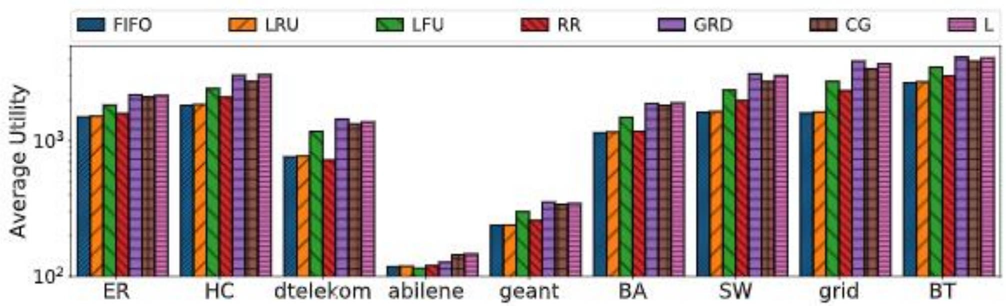
- ❑Problem Formulation
- ❑Deterministic Strategy
- ❑Stationary Randomized Strategy
- ❑Evaluation

Performance Evaluation

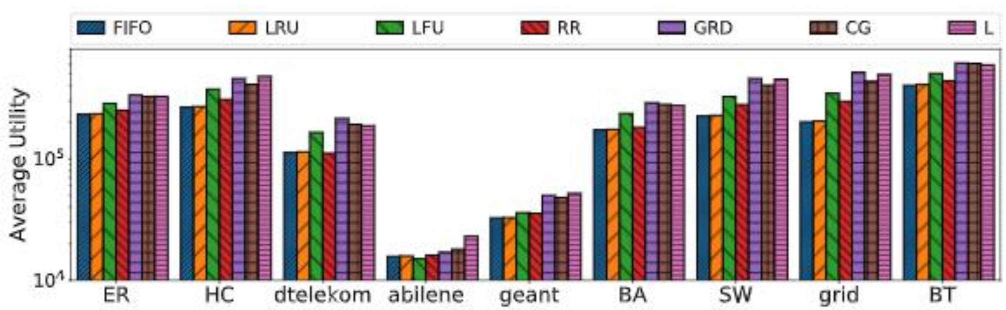
We show the algorithms outperform the baseline caching algorithms (LRU, LFU, FIFO, RR) over different synthetic and real-world topologies.

Graph	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert \mathcal{C} \rvert$	$\lvert \mathcal{R} \rvert$	$\lvert Q \rvert$	c_v
hypercube	128	896	300	1K	20	3
balanced-tree	341	680	300	1K	20	3
grid-2d	100	360	300	1K	20	3
erdos-renyi	100	1042	300	1K	20	3
small-world	100	491	300	1K	20	3
barabasi-albert	100	768	300	1K	20	3
geant	22	66	10	100	10	2
abilene	9	26	10	100	4	2
dtelekom	68	546	300	1K	20	3

Note that the y-axis uses log-scale



(a) $\alpha = 0.8$



(b) $\alpha = 2$

Caching algorithms

- ☐ GRD
- ☐ CG
- ☐ L
- ☐ FIFO
- ☐ LRU
- ☐ LFU
- ☐ RR

proposed ones

Effect of Fairness: Content Allocation

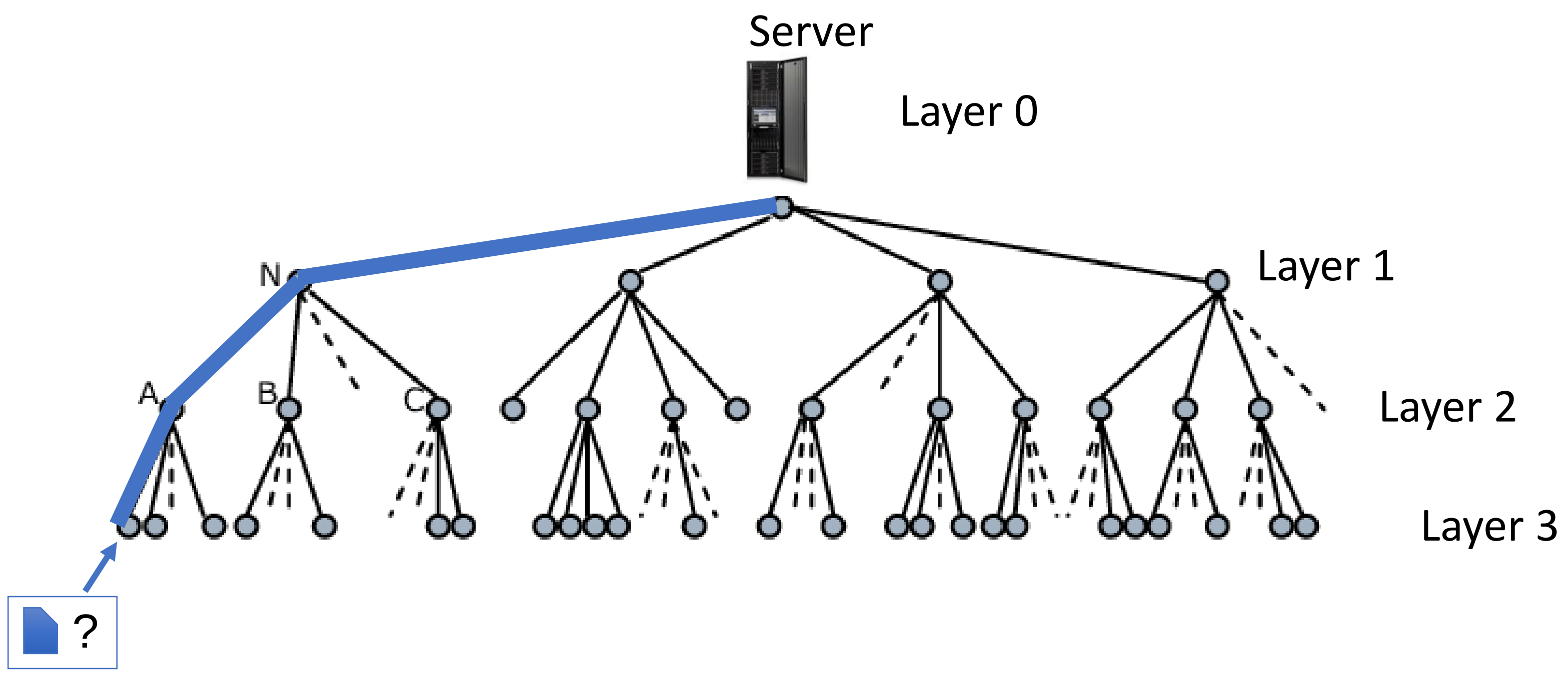
- Why we want to use this fair caching model? What will this model give us?

- We show that content items with **different popularities** are more fairly stored in the network if we consider the fair caching scheme.

Effect of Fairness: Content Allocation

4-ary balanced tree of height 4:

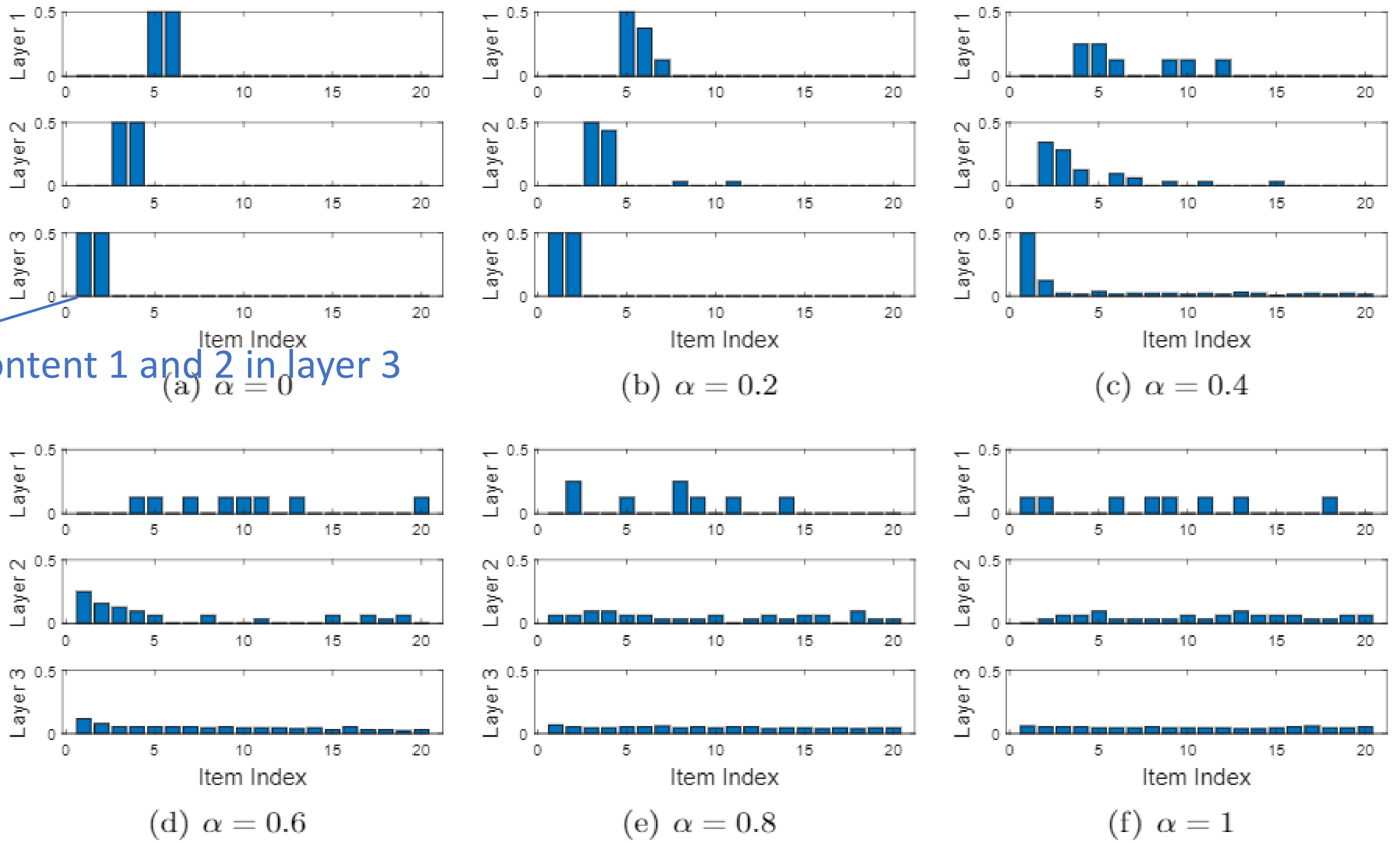
- Requests are generated u.a.r. at the leaf nodes
- The root is the server of all content items
- Content popularities follow Zipf distribution. **The content with smaller index has higher popularity**



Effect of Fairness: Content Allocation

4-ary tree with height 3:

- Requests enter the network from the leaf nodes
- The root is the server of all content items
- Content popularities follow Zipf distribution. **The content with smaller index has higher popularity**
- The height of a bar is the fraction of total cache space used to store a content



Only cache content 1 and 2 in layer 3

(a) $\alpha = 0$ (b) $\alpha = 0.2$ (c) $\alpha = 0.4$

(d) $\alpha = 0.6$ (e) $\alpha = 0.8$ (f) $\alpha = 1$

Summary

- Fair caching model: utility maximization problem
- We study several polynomial offline solutions
- Content items are more fairly stored in the network
- Future direction: Distributed and adaptive algorithms?

Thank you!

Submodularity

$$U(\cdot) \quad \circ \quad \lambda_{(i,p)} F_{(i,p)}(X) \quad \rightarrow \quad U(\lambda_{(i,p)} F_{(i,p)}(X))$$

non-decreasing concave	non-decreasing submodular	non-decreasing submodular

$$G(X) = \sum_{(i,p)\in\mathcal{R}} U(\lambda_{(i,p)} F_{(i,p)}(X))$$

monotone and submodular

$\sum_{i\in\mathcal{C}} x_{vi} \leq c_v$, for all $v \in V$

$x_{vi} \in \{0,1\}$, for all $v \in V$ and $i \in \mathcal{C}$

matroid constraints

We are maximizing a monotone submodular function under matroid constraints

Greedy Algorithm

Greedy algorithm produces a solution within 1/2 approximation factor [Calinescu et al. 2007].

Main idea: In each iteration, select an item to put in the cache of one of the nodes such that the overall utility increment is maximized.

Algorithm 1: Greedy

Input: $G : V \times \mathcal{C} \to \mathbb{R}_+$

$S \leftarrow \emptyset$

while $S^+ := \{(v, i) \in V \times \mathcal{C} : S \cup \{(v, i)\} \in \mathcal{D}_1\} \neq \emptyset$ **do**

$\quad (v^*, i^*) \leftarrow \arg\max_{(v,i) \in S^+} \left(G\left(S \cup \{(v, i)\right) - G(S)\right)$

$\quad S \leftarrow S \cup \{(v^*, i^*)\}$

end

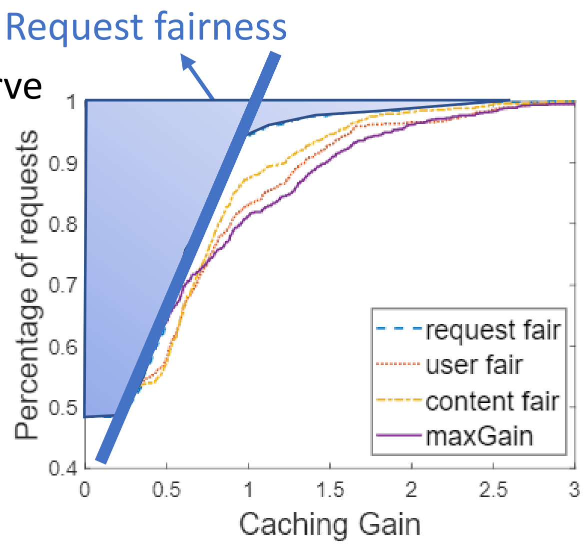
return S

Effect of Fairness: Caching Gain

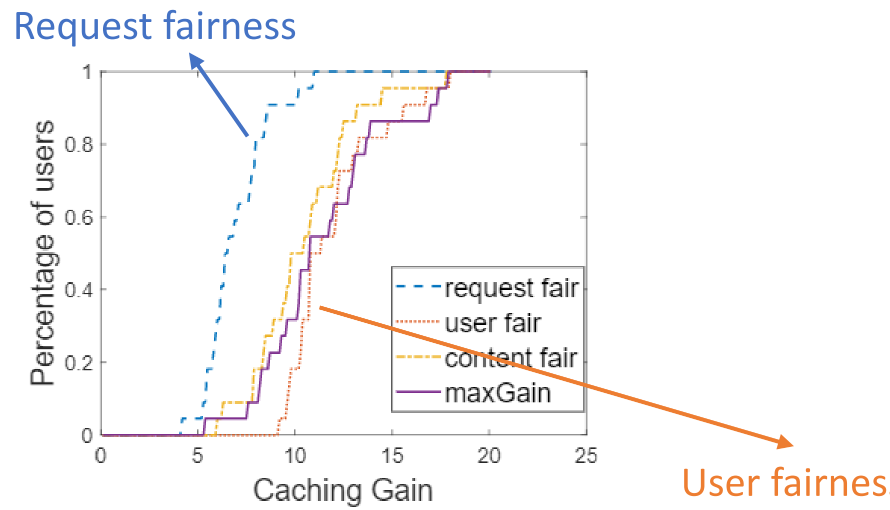
- We confirm that caching gain is more fairly distributed across different requests, content items, and users when we consider **request fairness**, **content fairness**, **and user fairness,** respectively.

Effect of Fairness: Caching Gain

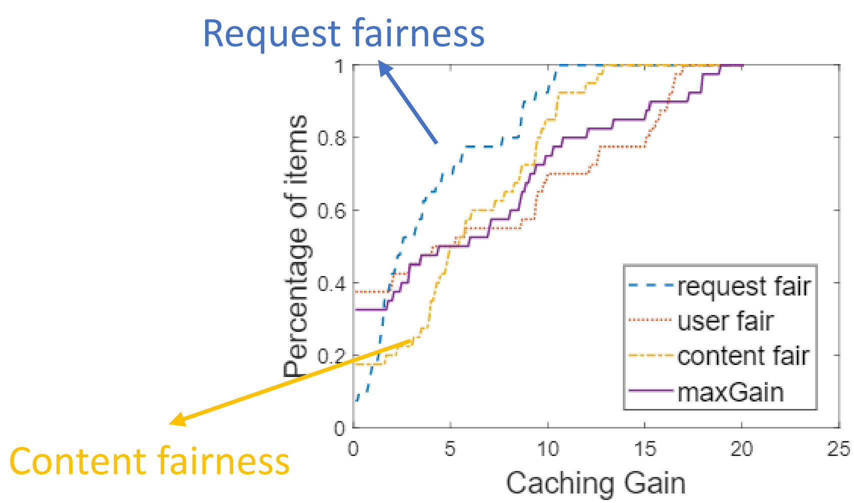
- Sharpness
- Area Above Curve (AAC)



(a) CDFs across requests



(b) CDFs across users



(c) CDFs across content items

L-method
α = 2
GEANT topology