How Bad is Selfish Caching?

Qian Ma
Department of Electrical and Computer Engineering, Northeastern University
qianma@northeastern.edu

Edmund Yeh
Department of Electrical and Computer Engineering, Northeastern University
eyeh@ece.neu.edu

Jianwei Huang
The Chinese University of Hong Kong, Shenzhen
The Chinese University of Hong Kong
jwhuang@ie.cuhk.edu.hk

ABSTRACT

Caching networks can reduce the routing costs of accessing contents by caching contents closer to users. However, cache nodes may belong to different entities and behave selfishly to maximize their own benefits, which often lead to performance degradation for the overall network. In this paper, we model the selfish behaviors of cache nodes as selfish caching games on arbitrary directed graphs with heterogeneous content popularity. We study the existence of a pure strategy Nash equilibrium (PSNE) in selfish caching games, and analyze its efficiency in terms of social welfare. We show that a PSNE does not always exist in arbitrary-topology caching networks. However, if the network does not have a mixed request loop, i.e., a directed loop in which each edge is traversed by at least one content request, we show that a PSNE always exists and can be found in polynomial time. We then show that the efficiency of Nash equilibria, captured by the price of anarchy (PoA), can be arbitrarily poor if we allow arbitrary content request patterns. However, when cache nodes have homogeneous request patterns, we show that the PoA is bounded even allowing arbitrary topologies. We further analyze the selfish caching games for cache nodes with limited computational capabilities, and show that an approximate PSNE exists with bounded PoA in certain cases of interest.

CCS CONCEPTS

• **Networks** → **Network algorithms**; *Network economics.*

KEYWORDS

Selfish caching games, directed graphs, price of anarchy

ACM Reference Format:
Qian Ma, Edmund Yeh, and Jianwei Huang. 2019. How Bad is Selfish Caching?. In *The Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc '19), July 2–5, 2019, Catania, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3323679.3326499

E. Yeh gratefully acknowledges support from National Science Foundation grant NeTS-1718355, as well as from research grants by Intel Corp. and Cisco Systems. This work is also supported by the General Research Fund CUHK 14219016 from Hong Kong UGC, and the Presidential Fund from the Chinese University of Hong Kong, Shenzhen.

Mobihoc '19, July 2–5, 2019, Catania, Italy

ACM ISBN 978-1-4503-6764-6/19/07…$15.00
https://doi.org/10.1145/3323679.3326499

1 INTRODUCTION

Caching networks can reduce the routing costs for accessing contents by caching the requested contents as close to the requesting users as possible. Prevailing caching networks include content delivery networks (CDN) [5, 12], information-centric networks (ICN) [46], femtocell networks [38], web caching networks [26], and peer-to-peer networks [10]. There has been extensive previous work (e.g., [3, 20, 34]) on how to optimally allocate contents to available caches. However, most existing work assumes that cache nodes are altruistic and cooperate with each other to optimize an overall network performance objective such as social welfare.

In practice, cache nodes may belong to different entities [44]. For example, in wireless community mesh networks such as Google WiFi [1] and Guifi [42], individual users contribute their wireless routers (as caches) to the community. On the Internet, different operators and providers deploy their own caching infrastructures and services. Examples include AT&T Content Delivery Network Service, Google Global Cache, Netflix Open Connect, and Akamai.

In caching networks where different entities operate their own caches, cache nodes may behave selfishly to maximize their own benefits. For example, in multi-hop wireless community mesh networks [14], a cache node has an incentive to cache the content items to minimize its own routing cost, which may not always maximize the social welfare. This motivates us to study the selfish caching behaviors through a game-theoretic approach.

To our best knowledge, this is the first paper to examine *selfish caching games* on arbitrary directed graphs with heterogeneous content popularity. We focus on the pure strategy Nash equilibrium (PSNE),[1] and address two fundamental questions.

- *First, is a PSNE guaranteed to exist in any selfish caching game?*
- *Second, if a PSNE exists, does it have a guaranteed efficiency in terms of social welfare?*

The short answers to the above two questions are "No" and "No". In other words, the selfish caching game can be arbitrarily bad.

In this paper, we characterize the conditions under which (i) a PSNE exists, and (ii) a PSNE has a guaranteed efficiency. We capture the efficiency of PSNE by the *price of anarchy* (PoA), which is the ratio of the social welfare achieved by the worst PSNE to that achieved by a socially optimal strategy [31, 37]. The analysis of PSNE and PoA takes into account the asymmetric and node-specific interdependencies among cache nodes, which reflect the network topology and content request patterns. Our analysis will help the network designer understand when the network behaves with certain performance guarantees, and how to create these conditions in the network.

[1] The main reason for implementing PSNE in practice is simplicity [43].

We analyze the selfish caching game in two scenarios. We first consider a scenario where all contents have equal sizes, which corresponds to practical applications such as video-on-demand services using harmonic broadcasting that divide each video into segments of equal size [23]. We then consider a scenario where contents have unequal sizes, which corresponds to practical applications such as video streaming services over HTTP (e.g., Netflix and Hulu) that split each video into segments of lengths from 2 to 10 seconds [19].

Our primary contributions are:

- *Selfish Caching Game*: To the best of our knowledge, this is the *first* work that studies the selfish caching game on directed graphs with arbitrary topologies and heterogeneous content popularity, which provides fundamental understanding of selfish behavior in general caching networks.
- *Pure Strategy Nash Equilibrium (PSNE)*: For selfish caching games with equal-sized content items, we first show that a PSNE does not always exist. We then show that a PSNE exists if the network does not have a mixed request loop, i.e., a directed loop in which each edge is traversed by at least one content request. Furthermore, we propose a polynomial-time algorithm to find a PSNE.
- *Price of Anarchy*: We show that the PoA in general can be arbitrarily poor if we allow arbitrary content request patterns. However, when cache nodes have homogeneous request patterns, we show that the selfish caching game is an α-scalable valid utility game and the PoA is bounded in arbitrary-topology caching networks.
- *Approximate PSNE*: For selfish caching games with unequal-sized content items, each node's payoff maximization problem is NP-hard. When cache nodes have limited computational capability, we show that their selfish caching behaviors lead to an approximate PSNE with bounded PoA in certain cases of interest.

The rest of the paper is organized as follows. In Section 2, we review related literature. In Section 3, we introduce our system model. In Section 4, we model the selfish caching game and analyze the PSNE. In Section 5, we study the PoA. In Section 6, we analyze selfish caching games with unequal-sized content items. In Section 7, we provide simulation results. We conclude in Section 8. **Due to space constraints, some of the proofs are presented in the online technical report [29].**

2 RELATED WORK

There has been a rich body of previous work on caching, many of which are summarized in an excellent recent survey [32]. In the following, we introduce related work regarding caching optimization and selfish caching game, respectively.

Caching Optimization. There is considerable recent literature on a variety of caching optimization problems, including proactive caching [40, 41], optimal caching under unknown content popularities [15, 48], distributed adaptive algorithms for optimal caching [3, 20, 34], caching at the edges [6, 25, 28, 49, 50], TTL (time-to-live) caches [4, 22], optimal caching in evolving networks [35], joint caching and routing optimization [2, 13], optimal cache partitioning [8], and collaborative caching [16, 21, 30, 36, 39, 47]. All the above work assumes that all cache nodes aim to maximize the social welfare. However, cache nodes operated by different entities may behave selfishly to maximize their own benefits.

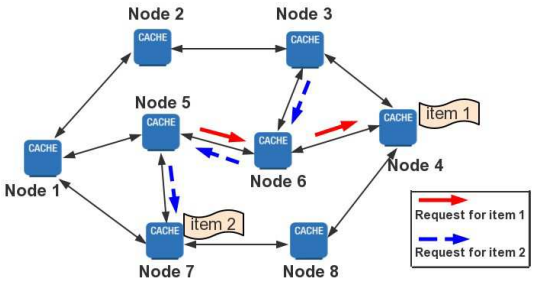

Figure 1: A caching network with $|V| = 8$ nodes and $|\mathcal{I}| = 2$ content items, where node 4 (node 7, respectively) is the designated server of item 1 (item 2, respectively). The request forwarding paths are fixed in our model. For example, the path of node 5 requesting item 1 is $p^{(5,1)} = (5, 6, 4)$, and the path of node 3 requesting item 2 is $p^{(3,2)} = (3, 6, 5, 7)$.

Selfish Caching Game. There are few papers studying the selfish caching behaviors in simple settings. In [9], Chun *et al.* studied the selfish caching game on undirected graphs with a single content item, assuming homogeneous content popularity across users. In [17], Goemans *et al.* studied the content market sharing game, where users get rewards for caching content items. They assumed that any node which caches a requested item can serve the request equally, without considering network topology. Authors in [26] and [33] studied a distributed selfish replication game in an undirected complete graph, where the distance between any two nodes is the same. In [18], Gopalakrishnan *et al.* studied the capacitated selfish replication game in an undirected network, where users are equally interested in some content items.

The analysis in the above literature is applicable to undirected graphs, and some are restricted to homogeneous content popularity. In this work, we study the selfish caching game on directed graphs with arbitrary topologies and heterogeneous content popularity.

3 SYSTEM MODEL

We consider a network of selfish caches, represented by a directed caching graph $G(V, E)$ with an *arbitrary topology*, where V is the set of cache nodes and E is the set of bidirectional edges which enable ARQ with asymmetric edge costs (see an example in Figure 1). Each cache node requests one or more content items (e.g., movies) from the set $\mathcal{I} = \{1, \ldots, |\mathcal{I}|\}$. For each content item $i \in \mathcal{I}$, there is a fixed set of *designated server nodes* $\mathcal{D}^i \subseteq V$, $|\mathcal{D}^i| > 0$, that store i in their permanent storage (outside of their caches). We consider equal-sized content items in Sections 3–5, which correspond to applications such as video-on-demand services using harmonic broadcasting that divide each video into segments of equal size [23]. We will consider the case of unequal-sized items in Section 6.

3.1 Caching Strategies

Each node $s \in V$ has a cache of capacity $c_s \in \mathbb{N}$, i.e., node s can store exactly c_s equal-sized content items. We denote the caching strategy of node $s \in V$ by $\boldsymbol{x}_s = \{x_{si} : \forall i \in \mathcal{I}\} \in \{0, 1\}^{|\mathcal{I}|}$, where

$$x_{si} \in \{0, 1\}, \text{ for all } i \in \mathcal{I},$$

indicates whether node s stores content item i, and satisfies

$$\sum_{i\in\mathcal{I}} x_{si} \le c_s, \text{ for all } s \in V.$$

We let $\boldsymbol{x}_{-s} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{s-1}, \boldsymbol{x}_{s+1}, \ldots, \boldsymbol{x}_{|V|}\}$ denote the caching strategy of nodes other than node s, and let $\boldsymbol{x} = \{\boldsymbol{x}_s, \boldsymbol{x}_{-s}\}$ denote the global caching strategy. Given $\boldsymbol{x}_s$, we let $Z_s = \{i : x_{si} = 1, i \in \mathcal{I}\}$ denote the set of items cached by node $s \in V$.

3.2 Content Requests

We describe each content request by a pair (s, i), where the request source[2] $s \in V$ requests content item $i \in \mathcal{I}$. We assume that each request (s, i) arrives according to a stationary ergodic process with arrival rate $\lambda_{(s,i)} \ge 0$ for all $s \in V$ and $i \in \mathcal{I}$, which reflects *heterogeneous content popularity* across items and request nodes.

Request (s, i) is forwarded over a pre-determined fixed *request forwarding path*[3] $p^{(s,i)}$, from request source s to one of content item i's designated server nodes in $\mathcal{D}^i$. Specifically, the path $p^{(s,i)}$ of length $K \le |V|$ is a sequence $(p_1, \ldots, p_K)$ of nodes $p_k \in V$ such that $p_1 = s$, $p_K \in \mathcal{D}^i$, and $(p_k, p_{k+1}) \in E$ for all $k \in \{1, \ldots, K-1\}$. We require that $p^{(s,i)}$ contains no loops ($p_k \ne p_l$ for all $1 \le k < l \le K$) and no node other than the terminal node on $p^{(s,i)}$ is a designated server for content item i ($p_k \notin \mathcal{D}^i$ for all $1 \le k < K$). For request (s, i), we let $V_{(s,i)} = \{v : v \in p^{(s,i)}, v \ne s, v \notin \mathcal{D}^i\}$ denote the set of intermediate nodes on path $p^{(s,i)}$. We denote $V_s = \cup_{i\in\mathcal{I}} V_{(s,i)}$ as the set of intermediate nodes on all the request forwarding paths of node s.[4]

Request (s, i) travels along path $p^{(s,i)}$ until either (i) the request reaches a node $v \in p^{(s,i)}$ such that node v caches content item i, i.e., $x_{vi} = 1$ or, (ii) if $x_{vi} = 0$ for all $v \in p^{(s,i)} \setminus \{p_K\}$, the request reaches $p_K \in \mathcal{D}^i$. Having found the closest copy of content item i, the network generates a *response message* carrying the requested content item i. The response message is propagated in the reverse direction along the request forwarding path, i.e., from the closest node with content item i back to the request source node s.

3.3 Routing Costs

Transferring a content item across edge $e = (u, v) \in E$ incurs a cost (e.g., delay or financial expense) denoted by $w_{uv} \ge 0$.[5] Since the size of each request message is relatively small compared with the content item, we assume that costs are only due to content item transfers, and the costs of forwarding requests are negligible [20]. To serve the request (s, i), the routing cost depends on the caching decision x_{si} of the request source node s, as well as the caching decisions $x_{vi}, \forall v \in V_{(s,i)}$, of all the intermediate nodes on the request forwarding path $p^{(s,i)}$. Specifically, the routing cost of transferring item i over the reverse direction of $p^{(s,i)}$ is

$$h_{(s,i)}\left(x_{si}, \{x_{vi} : v \in V_{(s,i)}\}\right) = \sum_{k=1}^{|p^{(s,i)}|-1} w_{p_{k+1}p_k} \prod_{k'=1}^{k} \left(1 - x_{p_{k'}i}\right)$$
$$= \sum_{k=1}^{|p^{(s,i)}|-1} w_{p_{k+1}p_k} \left(1 - x_{si}\right) \prod_{k'=2}^{k} \left(1 - x_{p_{k'}i}\right).$$

Note that $h_{(s,i)}(\cdot)$ includes the cost on edge (p_{k+1}, p_k), i.e., $w_{p_{k+1}p_k}$, if and only if none of the nodes from p_1 to p_k on path $p^{(s,i)}$ has cached content item i. For example, in Figure 1, $p^{(3,2)} = (3, 6, 5, 7)$ and the routing cost of request $(3, 2)$ depends on x_{32}, x_{62} and x_{52}. If $(x_{32}, x_{62}, x_{52}) = (0, 0, 1)$, then $h_{(3,2)}(x_{32}, x_{62}, x_{52}) = w_{63} + w_{56}$.

Table 1: Key Notation

$G(V, E)$	Caching graph, with nodes in V and edges in E
c_s	Cache capacity of node $s \in V$
w_{uv}	Cost on edge $(u, v) \in E$
$\mathcal{I}$	Set of content items
$\mathcal{D}^i$	Set of designated servers for content item $i \in \mathcal{I}$
(s, i)	Request for item i from node s
$\lambda_{(s,i)}$	Arrival rate of request (s, i)
$p^{(s,i)}$	Request forwarding path of request (s, i)
$V_{(s,i)}$	The set of intermediate nodes on path $p^{(s,i)}$
V_s	The set of intermediate nodes on node s' paths
x_{si}	Caching strategy of node $s \in V$ for item $i \in \mathcal{I}$
$\boldsymbol{x}_s$	Caching strategy of node $s \in V$
Z_s	The set of content items cached by node $s \in V$
$\boldsymbol{x}$	Global caching strategy of all nodes
$h_{(s,i)}$	The routing cost to serve request (s, i)
h_s	The routing cost of node s
g_s	The caching gain of node s
G	The aggregate caching gain in the network

3.4 Selfish Caching Behavior

Each selfish cache node $s \in V$ seeks a caching strategy to optimize its own benefit, i.e., minimizing the aggregate expected cost for serving all its own requests, calculated as follows:

$$h_s\left(\boldsymbol{x}_s, \{\boldsymbol{x}_v : v \in V_s\}\right) = \sum_{i\in\mathcal{I}} \lambda_{(s,i)} \cdot h_{(s,i)}\left(x_{si}, \{x_{vi} : v \in V_{(s,i)}\}\right). \quad (1)$$

For notation simplicity, we write $h_s(\cdot)$ as $h_s(\boldsymbol{x}_s, \boldsymbol{x}_{-s})$. In the absence of caching, i.e., $\boldsymbol{x} = \boldsymbol{0}$, the aggregate expected cost of node s is:

$$h_s(\boldsymbol{0}) = \sum_{i\in\mathcal{I}} \lambda_{(s,i)} \sum_{k=1}^{|p^{(s,i)}|-1} w_{p_{k+1}p_k}.$$

We define the *caching gain* of node s as

$$g_s(\boldsymbol{x}_s, \boldsymbol{x}_{-s}) = h_s(\boldsymbol{0}) - h_s(\boldsymbol{x}_s, \boldsymbol{x}_{-s}). \quad (2)$$

Intuitively, the caching gain is the cost reduction enabled by caching. Since $h_s(\boldsymbol{0})$ is a constant, minimizing the aggregate expected cost in (1) is equivalent to maximizing the caching gain in (2). Hence, the caching gain in (2) serves as node s' payoff function.

4 SELFISH CACHING GAME

In this section, we model the interactions among selfish cache nodes by a selfish caching game on directed graphs. We construct an example where the pure strategy Nash equilibrium (PSNE) does not exist for such a game. We then identify the condition under

[2] We consider a request source to be a point of aggregation which combines many network users. While a single user may request a given content item only once over a time period, an aggregation point is likely to submit many requests for a given content item over a time period.

[3] In this paper, we assume that the request forwarding path is determined in a longer timescale compared with caching, which is similar as in the named data networks.

[4] Note that each cache node can play some or all of the following roles: a designated server of content items, a source of requests, and an intermediate node on request forwarding paths.

[5] We do not model the congestion effect on each edge, and hence ignore throughput issues. How to combine cost and throughput issues is an interesting open problem.

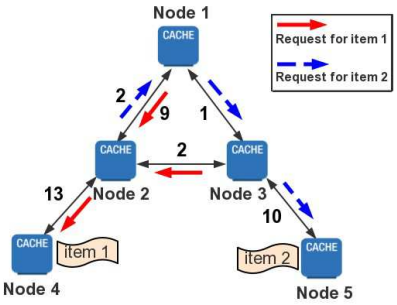

Figure 2: An example where PSNE does not exist. The caching network has $|V| = 5$ nodes and $|\mathcal{I}| = 2$ content items, where node 4 (node 5, respectively) is the designated server of item 1 (item 2, respectively). The cache capacity is 1 at each node. The request arrival rates satisfy $\lambda_{(v,i)} = \lambda_i, \forall v \in V, i \in \mathcal{I}$, where $\lambda_1 = 10$ and $\lambda_2 = 14$. The request forwarding paths are fixed, for example, $p^{(3,1)} = (3, 2, 4)$. The edge costs are asymmetric, i.e., $w_{12} = 9$ and $w_{21} = 2$.

which a PSNE exists, and propose a polynomial-time algorithm to find a PSNE under the condition.

4.1 Game Modeling

We define the selfish caching game as follows:

Game 1 (Selfish Caching Game on Directed Graphs).

- *Players: the set V of cache nodes on the caching graph G(V,E);*
- *Strategies: the caching strategy $\boldsymbol{x}_s = \{x_{si} : \forall i \in \mathcal{I}\}$ for each cache node $s \in V$, where $x_{si} \in \{0, 1\}$ and $\sum_{i\in\mathcal{I}} x_{si} \leq c_s$;*
- *Payoffs: the caching gain $g_s(\boldsymbol{x}_s, \boldsymbol{x}_{-s})$ for each $s \in V$.*

Since the selfish caching game is a finite game, there exists at least one mixed strategy Nash equilibrium (including pure strategy Nash equilibrium as a special case). However, since it is difficult to implement random caching strategies in practical caching networks, we focus on analyzing pure strategy Nash equilibria in this paper, as defined below.

Definition 4.1 (Pure Strategy Nash Equilibrium). A pure strategy Nash equilibrium of the selfish caching game is a caching strategy profile $\boldsymbol{x}^{\text{NE}}$ such that for every cache node $s \in V$,

$$g_s(\boldsymbol{x}_s^{\text{NE}}, \boldsymbol{x}_{-s}^{\text{NE}}) \geq g_s(\boldsymbol{x}_s, \boldsymbol{x}_{-s}^{\text{NE}}), \text{ for all feasible } \boldsymbol{x}_s. \tag{3}$$

4.2 An Example with No PSNE

In the following, we first show that the PSNE does not always exist.

Theorem 4.2. *There exists a selfish caching game for which the pure strategy Nash equilibrium does not exist.*

Proof. Figure 2 is an example with no PSNE. For node 4 (node 5, respectively), caching item 2 (item 1, respectively) is its dominant strategy. Now we analyze the selfish behaviors of nodes 1, 2, and 3. It is easy to verify that for all 8 feasible caching strategy profiles, there always exists one cache node that can improve its caching gain by changing its caching strategy unilaterally. For example, if all the three nodes cache item 1, then node 3 has the the incentive to cache item 2 to improve its caching gain assuming that the other two nodes do no change their caching strategies. Hence there is no strategy profile where everyone is achieving its maximum payoff assuming other nodes do not change their strategies. Hence, the PSNE does not exist. □

4.3 Existence of a PSNE

Deciding the existence of a PSNE for games on graphs is NP-hard in general [45]. However, we identify the condition under which a PSNE of the selfish caching game exists and can be found in polynomial time. To proceed, we first introduce the definition below.

Definition 4.3 (Mixed Request Loop). A mixed request loop on a directed graph is a directed loop $(p_1, p_2, \ldots, p_K, p_{K+1} = p_1)$ involving $3 \leq K \leq |V|$ nodes, where $p_k \in V$ for $1 \leq k \leq K$, $p_k \neq p_l$ for all $1 \leq k < l \leq K$, and at least one content request traverses edge $(p_k, p_{k+1}) \in E$ for all $1 \leq k \leq K$.

In Figure 2, $(1, 3, 2, 1)$ forms a mixed request loop, where requests for item 1 traverse edge $(3, 2)$, and requests for item 2 traverse edges $(2, 1)$ and $(1, 3)$.

Note that a loop on graph is not always a mixed request loop. For example, $(1, 3, 2, 1)$ in Figure 2 is a loop since they are connected. However, if $p^{(3,1)} = (3, 1, 2, 4)$ rather than $(3, 2, 4)$, then no request will traverse edge $(3, 2)$, and hence loop $(1, 3, 2, 1)$ will not be a mixed request loop.[6] Hence, we can avoid mixed request loops by properly choosing the request forwarding paths.

In the following, we will show that a PSNE exists in the selfish caching game on caching graphs with no mixed request loop.

Theorem 4.4. *A PSNE always exists in the selfish caching game on caching graphs with no mixed request loop.*

Proof. See Appendix B in the technical report [29]. □

Theorem 4.4 holds in caching networks with arbitrary topologies and heterogeneous content popularity. We prove the existence[7] of the PSNE by finding a PSNE in polynomial time.

4.4 Polynomial-Time Algorithm to Find a PSNE

In this section, we present a polynomial-time algorithm to find a PSNE for the selfish caching game. Specifically, for each selfish caching game, we can define a *state graph* [17] as follows.

Recall that given node s' caching strategy $\boldsymbol{x}_s$, the set $Z_s = \{i : x_{si} = 1, i \in \mathcal{I}\}$ is the set of content items cached by node $s \in V$. Hence we can use $\boldsymbol{x} = \{\boldsymbol{x}_s : \forall s \in V\}$ and $Z = \{Z_s : \forall s \in V\}$ interchangeably to represent the caching strategy profile.

Definition 4.5 (State Graph [17]). A state graph is a directed graph where each vertex corresponds to a strategy profile Z. There is a directed arc from vertex Z to vertex Z' with label v if the only difference between Z and Z' is the strategy of player v and the payoff of player v in Z is strictly less than its payoff in Z'.

A PSNE corresponds to a vertex on the state graph without any outgoing arc, i.e., a sink. Hence identifying a PSNE of the selfish caching game is equivalent to identifying a sink on the corresponding state graph.

[6] A figure for the new setting can be found in Appendix A in the technical report [29].
[7] The selfish caching game generally admits multiple PSNEs, depending on system parameters such as edge weights and request arrival rates.

Algorithm 1: Find PSNE on State Graph [17]

Input: $G(V, E)$, $\mathcal{I}$, w_{uv}, $\forall(u, v) \in E$, $\lambda_{(s,i)}$ and $p^{(s,i)}$, for all $s \in V$, $i \in \mathcal{I}$
Output: Z^{NE}

1 Set $Z = \emptyset$;
2 **repeat**
3 Randomly pick a node $s \in V$ where $|Z_s| < c_s$;
4 Add item i^* where $i^* \in \arg\max_{i \in \mathcal{I} \setminus Z_s} g_s(Z_s \cup \{i\}, Z_{-s})$ to node s, i.e., $Z_s \leftarrow Z_s \cup \{i^*\}$;
5 **while** $\exists v \in V, j \notin Z_v, t \in Z_v$, *such that* $g_v(Z_v \cup \{j\} \setminus \{t\}, Z_{-v}) > g_v(Z_v, Z_{-v})$ **do**
6 Set $Z_v \leftarrow Z_v \cup \{j\} \setminus \{t\}$;
7 **end**
8 **until** $\forall s \in V$ *satisfies* $|Z_s| = c_s$;
9 Set $Z^{\text{NE}} = Z$;

We propose a polynomial-time algorithm (Algorithm 1 [17]) to find a sink on the state graph. The algorithm proceeds in rounds. The first round starts at the vertex $Z = \emptyset$, corresponding to the strategy profile where none of the cache nodes cache any content item (Line 1 of Algorithm 1). In each round, the first arc traversed on the state graph corresponds to an *add arc* where a player, say s, changes from Z_s to $Z_s \cup \{i^*\}$. Intuitively, player s adds only one content item i^* to its cache, where we select i^* among all possible content items not currently in Z_s to maximize player s' caching gain (Lines 3-4 of Algorithm 1). After the first arc, subsequent arcs in the same round correspond to *change arcs*. Specifically, a change arc corresponds to a player, say v, replacing Z_v by $Z_v \cup \{j\} \setminus \{t\}$, where $j \notin Z_v$ and $t \in Z_v$. Intuitively, player v replaces content item t for content item j if $g_v(Z_v \cup \{j\} \setminus \{t\}, Z_{-v}) > g_v(Z_v, Z_{-v})$ (Lines 5-7 of Algorithm 1). When the current vertex on the state graph has no change arcs, one round ends. For the vertex where a round ends, if there is an add arc outgoing from it, a new round starts; otherwise, it is a sink and the algorithm terminates. Such a sink corresponds to the PSNE.

In the following theorem, we show that Algorithm 1 can find a sink on the state graph in polynomial time.

Theorem 4.6. *For the selfish caching game on arbitrary-topology caching graphs with no mixed request loop, Algorithm 1 computes a PSNE in polynomial time by traversing a path of length at most* $|V||\mathcal{I}|^2(|V|-2)^2$ *on the corresponding state graph.*

Proof. See Appendix C in the technical report [29]. □

At any given vertex of the state graph, Algorithm 1 only requires one to find the next arc to traverse, which takes $O(|V|)$ time. Hence, the total maximum running time of Algorithm 1 is $O(|V|^2|\mathcal{I}|^2(|V|-2)^2)$. Furthermore, different random choices of the next arc to traverse in Algorithm 1 correspond to different outcomes if there is more than one PSNE in the selfish caching game.

Since each cache node maximizes its own benefit, a PSNE of the selfish caching game does not in general optimize the social welfare. We will quantify the efficiency of the Nash equilibria in terms of social welfare next.

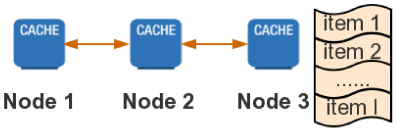

Figure 3: An example where PoA approaches 0. Consider a caching network with $|V| = 3$ nodes where node 3 is the designated server of content items in set $\mathcal{I} = \{1, 2, \ldots, I\}$. The request arrival rates at node 1 satisfy $\lambda_{(1,1)} > 0, \lambda_{(1,i)} = \lambda_{(1,1)} - \epsilon > 0$, where $\epsilon > 0$, for $i \in I \setminus \{1\}$. Node 2 does not generate request, i.e., $\lambda_{(2,i)} = 0, \forall i \in I$. The cache capacities are $c_1 = 1$ and $c_2 = I - 1$.

5 PRICE OF ANARCHY

To evaluate the efficiency of Nash equilibria, we analyze the price of anarchy (PoA) [31], i.e., the ratio of the social welfare achieved by the worst Nash equilibrium to that achieved by a socially optimal strategy.

In this paper, we define the social welfare as the aggregate caching gain in the network. Specifically, the social welfare maximization problem is

$$\begin{aligned} &\max\ G(\boldsymbol{x}) \triangleq \textstyle\sum_{s \in V} g_s(\boldsymbol{x}_s, \boldsymbol{x}_{-s}) \\ &\text{s.t.}\ \ \textstyle\sum_{i \in \mathcal{I}} x_{si} \le c_s,\ x_{si} \in \{0,1\},\ \forall s \in V, i \in \mathcal{I}. \end{aligned} \tag{4}$$

Problem (4) is NP-hard [20]. Hence it is challenging to calculate the socially optimal caching strategy and analyze the PoA in general.

In the following, we first show that the PoA can be arbitrarily poor if we allow any content request patterns. Under reasonable constraints of request patterns and paths, however, we can show that the PoA is bounded in general caching networks. Furthermore, for given caching networks with known network topology and parameters, we can derive a better bound for PoA.

5.1 An Example with an Arbitrarily Poor PoA

In the following lemma, we show that the PoA can be arbitrarily close to 0, indicating that the selfish caching behaviors can lead to unboundedly poor performance in terms of social welfare.

Lemma 5.1. *There exists a selfish caching game for which the PoA is arbitrarily close to* 0.

Proof. We construct an example where PoA approaches 0, as shown in Figure 3. In this example, the socially optimal caching strategy is for node 1 to cache content item 1 and for node 2 to cache content items 2 to I. The optimal social welfare, i.e., aggregate caching gain, is[8]

$$G^{\text{SO}} = \lambda_{(1,1)}(w_{21} + w_{32}) + \textstyle\sum_{i=2}^{I} \lambda_{(1,i)} w_{32}.$$

There may exist more than one PSNE, one of which is that node 1 caches item 1 and node 2 caches none of the content items (since node 2 has no request of its own). The social welfare achieved by this PSNE is $G^{\text{NE}} = \lambda_{(1,1)}(w_{21} + w_{32})$. We have

$$\frac{G^{\text{NE}}}{G^{\text{SO}}} = \frac{1}{1 + \sum_{i=2}^{I} \frac{\lambda_{(1,i)} w_{32}}{\lambda_{(1,1)}(w_{21}+w_{32})}}.$$

[8]The superscript "SO" represents socially optimal.

When $w_{32} \gg w_{21}$ and $\epsilon \to 0$, we have $\frac{\lambda_{(1,i)}w_{32}}{\lambda_{(1,1)}(w_{21}+w_{32})} \to 1$ and $\frac{G^{\text{NE}}}{G^{\text{SO}}} \to \frac{1}{I}$, which goes to 0 as I becomes very large. Since PoA measures the worst case ratio between any PSNE and the social optimal solution, the PoA will be no larger than $G^{\text{NE}}/G^{\text{SO}}$ and hence can be arbitrarily close to 0. □

5.2 Bound on PoA

In this section, we show that under reasonable constraints of request patterns and paths, the selfish caching game belongs to a class of games that we call *α-scalable valid utility games,* and the PoA is bounded by the length of the longest request forwarding path in the network.

Recall that $Z_s = \{i : x_{si} = 1, i \in \mathcal{I}\}$ represents the set of content items cached by node $s \in V$. For convenience, we express the caching gain of node s as $g_s(Z_s, Z_{-s})$, and the aggregate caching gain of the network as $G(Z) = \sum_{s\in V} g_s(Z_s, Z_{-s})$.

We first define the *valid utility games* (for general games not restricted to selfish caching games), introduced by Vetta in [43].

Definition 5.2 (Valid Utility Game [43]). A game (with social function $\gamma(\cdot)$ and individual payoff functions $f_s(\cdot), \forall s \in V$)[9] is a valid utility game if the following three properties are satisfied:

(1) The social function $\gamma(\cdot)$ is non-decreasing and submodular. Mathematically, for every content item $i \in \mathcal{I}$ and for any subsets Z, Z' such that $Z \subseteq Z'$,

$$\gamma(Z) \le \gamma(Z'), \tag{5}$$

$$\gamma(Z \cup \{i\}) - \gamma(Z) \ge \gamma(Z' \cup \{i\}) - \gamma(Z'). \tag{6}$$

(2) The sum of players' payoff functions $f_s(\cdot)$ for any strategy profile $\boldsymbol{x}$ should be no larger than the social function $\gamma(\cdot)$:

$$\textstyle\sum_{s\in V} f_s(\boldsymbol{x}_s, \boldsymbol{x}_{-s}) \le \gamma(\boldsymbol{x}). \tag{7}$$

(3) The payoff of a player is no less than the difference between the social function when the player participates and that when it does not participate

$$f_s(\boldsymbol{x}_s, \boldsymbol{x}_{-s}) \ge \gamma(\boldsymbol{x}_s, \boldsymbol{x}_{-s}) - \gamma(\boldsymbol{0}, \boldsymbol{x}_{-s}). \tag{8}$$

Vetta in [43] proved that the PoA of a valid utility game is bounded by 2. In the following, we define a new class of games called *α-scalable valid utility games,* which generalizes the notation of valid utility games.

Definition 5.3 (α-Scalable Valid Utility Game). A game is an α-scalable valid utility game if it satisfies the two properties in (5), (6), and (7), and the payoff of a player is no less than the product of a positive constant α and the difference between the social function when the player participates and that when it does not participate:

$$f_s(\boldsymbol{x}_s, \boldsymbol{x}_{-s}) \ge \frac{1}{\alpha} \cdot \Big[\gamma(\boldsymbol{x}_s, \boldsymbol{x}_{-s}) - \gamma(\boldsymbol{0}, \boldsymbol{x}_{-s})\Big]. \tag{9}$$

Note that the valid utility game is a special case of the α-scalable valid utility games with $\alpha = 1$. We show that the selfish caching game is an α-scalable valid utility game with $\alpha = \max_{v\in V, i\in\mathcal{I}} |p^{(v,i)}| - 1$ when the following two properties are satisfied.

[9]Note that the social function can be any objective that the network aims to optimize, and may not be the summation of individual players' payoff functions.

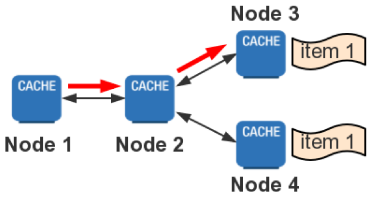

Figure 4: An example that satisfies the path overlap property. Here, $p^{(2,1)} = (2, 3)$ and $p^{(1,1)} = (1, 2, 3)$.

Definition 5.4 (Homogeneous Request Pattern Property). The request arrival processes for content item $i \in \mathcal{I}$ at different nodes are the same, i.e.,

$$\lambda_{(s,i)} = \lambda_i, \forall s \in V, i \in \mathcal{I}. \tag{10}$$

The homogeneous request pattern property implies that each content item has a global popularity. Note that even under the homogeneous request pattern property, the popularity of different content items can be different, i.e., $\lambda_i \ne \lambda_j, i \ne j, i, j \in \mathcal{I}$.

Definition 5.5 (Path Overlap Property). If node s is on path $p^{(v,i)}$, then starting from node s, path $p^{(v,i)}$ overlaps with path $p^{(s,i)}$, i.e.,

$$s \in p^{(v,i)} \Rightarrow p^{(s,i)} \subseteq p^{(v,i)}. \tag{11}$$

Figure 4 shows an example that satisfies the path overlap property. Note that the path overlap property is naturally satisfied when each node chooses a unique shortest path to fetch content items.

Theorem 5.6. *The selfish caching game with the homogeneous request pattern and path overlap properties on caching graphs with no mixed request loop is an α-scalable valid utility game where*

$$\alpha = \max_{v\in V, i\in\mathcal{I}} |p^{(v,i)}| - 1. \tag{12}$$

Proof. See Appendix D in the technical report [29]. □

In the following theorem, we show that when the selfish caching game is an α-scalable valid utility game, the PoA is bounded by the length of the longest request forwarding path in the network.

Theorem 5.7. *When the selfish caching game is an α-scalable valid utility game, the PoA satisfies*

$$PoA \ge \frac{1}{1+\alpha} = \frac{1}{\max_{v\in V, i\in\mathcal{I}} |p^{(v,i)}|}. \tag{13}$$

Proof. See Appendix E in the technical report [29]. □

The above performance guarantee is true for general caching networks with an arbitrary topology. However, given a caching network with a known topology and network parameters, we can further explore the network structure and derive a better bound for PoA. This is achieved by characterizing the discrete curvature of the social function, as discussed next.

5.3 PoA and the Discrete Curvature of the Social Function

To understand how the discrete curvature [43] of the social function will affect our PoA analysis, we first introduce the discrete derivative. For a set function $G(\cdot)$, we define the discrete derivative at Y in the direction Z as

$$G'_Z(Y) = G(Y \cup Z) - G(Y).$$

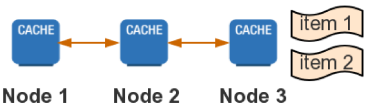

Figure 5: An example to calculate the value of $\delta(G)$. We assume $\lambda_{(v,i)} = \lambda_i, \forall v \in V, i \in \mathcal{I}$. According to (14), we have $\delta(G) = \frac{1}{1+w_{21}/w_{32}} \in [0,1]$. When $w_{21}/w_{32} \to 0$, we have $\delta(G) \to 1$; when $w_{21}/w_{32} \to \infty$, we have $\delta(G) \to 0$.

We define the *discrete curvature* of a non-decreasing, submodular social function $G(\cdot)$ to be

$$\delta(G) = \max_{\forall s \in V: G'_{Z_s}(\emptyset) > 0} \frac{G'_{Z_s}(\emptyset) - G'_{Z_s}(\mathcal{I}^{|V|} - Z_s)}{G'_{Z_s}(\emptyset)} \in [0,1], \quad (14)$$

where $\mathcal{I}^{|V|} - Z_s$ represents the caching strategy profile (which can be infeasible) under which node s caches content items in set $\mathcal{I} \setminus Z_s$ while all other nodes cache all content items in set $\mathcal{I}$. Figure 5 shows an example to calculate the value of $\delta(G)$.

Given the discrete curvature of the social function, we can obtain a better bound for the PoA of the selfish caching game.

Theorem 5.8. *Given the discrete curvature $\delta(G)$ of the social function, for any selfish caching game with the homogeneous request pattern and path overlap properties on caching graphs with no mixed request loop, the PoA satisfies*

$$PoA \geq \frac{1}{\alpha + \delta(G)}. \quad (15)$$

Proof. See Appendix F in the technical report [29]. □

Since $\delta(G) \in [0,1]$ exploits the curvature property of the given network structure, the performance guarantee in (15) is better than the one in (13).

6 SELFISH CACHING GAMES WITH UNEQUAL-SIZED ITEMS

In this section, we analyze more general selfish caching games with unequal-sized items, which correspond to the practical applications such as video streaming services over HTTP (e.g., Netflix and Hulu) that split each video into segments of lengths from 2 to 10 seconds [19]. We show that the caching gain maximization problem for each cache node is NP-hard. We further generalize the model by considering that each cache node has limited computational capability to solve its caching gain maximization problem. This may lead to an approximate PSNE. We analyze the existence and efficiency of an approximate PSNE under these two generalizations.

6.1 Game Modeling

Let L_i denote the size of content item i, for all $i \in \mathcal{I}$. We define the selfish caching game with unequal-sized items as follows:

Game 2 (Selfish Caching Game with Unequal-Sized Items).

- *Players: the set V of cache nodes on the caching graph G(V,E);*
- *Strategies: the caching strategy $\boldsymbol{x}_s = \{x_{si} : \forall i \in \mathcal{I}\}$ for each cache node $s \in V$, where $x_{si} \in \{0,1\}$ and $\sum_{i \in \mathcal{I}} L_i x_{si} \leq c_s$;*
- *Payoffs: the caching gain $g_s(\boldsymbol{x}_s, \boldsymbol{x}_{-s})$ for each $s \in V$.*

In the following, we will show that for each cache node $s \in V$, given fixed $\boldsymbol{x}_{-s}$, its caching gain maximization problem is equivalent to a knapsack problem. For node $s \in V$, the caching gain in (2) can be equivalently written as

$$g_s(\boldsymbol{x}_s, \boldsymbol{x}_{-s}) = \sum_{i \in \mathcal{I}} \lambda_{(s,i)} \sum_{k=1}^{|p^{(s,i)}|-1} w_{p_{k+1}p_k} \left(1 - \prod_{k'=2}^{k} (1 - x_{p_{k'}i})\right) + \sum_{i \in \mathcal{I}} x_{si} \cdot \lambda_{(s,i)} \sum_{k=1}^{|p^{(s,i)}|-1} w_{p_{k+1}p_k} \prod_{k'=2}^{k} (1 - x_{p_{k'}i}). \quad (16)$$

Given fixed $\boldsymbol{x}_{-s}$, the first term in (16) is a constant, while the second term in (16) depends on $\boldsymbol{x}_s$. Define weight

$$q_{si}(\boldsymbol{x}_{-s}) = \lambda_{(s,i)} \sum_{k=1}^{|p^{(s,i)}|-1} w_{p_{k+1}p_k} \prod_{k'=2}^{k} (1 - x_{p_{k'}i}). \quad (17)$$

Intuitively, $q_{si}(\boldsymbol{x}_{-s})$ represents the routing cost for request (s,i) under $\boldsymbol{x}_{-s}$ if $x_{si} = 0$. Given fixed $\boldsymbol{x}_{-s}$, the caching gain maximization problem of node s is equivalent to the following knapsack problem:

$$\max_{\boldsymbol{x}_s} \sum_{i \in \mathcal{I}} x_{si} \cdot q_{si}(\boldsymbol{x}_{-s}) \quad \text{s.t.} \ \sum_{i \in \mathcal{I}} L_i x_{si} \leq c_s, x_{si} \in \{0,1\}, \forall i \in \mathcal{I}. \quad (18)$$

Solving the knapsack problem (18) is NP-hard.[10] In practice, each cache node has limited computational capability in a short time period (e.g., minutes or hours for which the request patterns remain unchanged [15]), and can only solve the knapsack problem (18) to an approximate solution $\hat{\boldsymbol{x}}_s = \{\hat{x}_{si} : \forall i \in \mathcal{I}\}$. There is extensive literature on the polynomial-time approximation algorithms for the knapsack problem [24]. We present one such algorithm in Lines 4-12 of Algorithm 2, which achieves a 1/2 approximation ratio (see Section 9.4.2 of [24]).

Now we consider the general case where each cache node can obtain only a $1/\beta$ approximate solution with $\beta > 1$ for problem (18). This leads to the β-approximate PSNE of Game 2.

6.2 Existence of an Approximate PSNE

A β-approximate PSNE is a strategy profile for which no player can improve its caching gain by a factor more than β of its current caching gain by unilaterally changing its strategy.[11]

Definition 6.1 (β-Approximate PSNE [7]). A pure strategy profile $\boldsymbol{x}^{\beta-NE}$ is a β-approximate PSNE if no player can find an alternative pure strategy with a payoff which is more that β times its current payoff. That is for any player $s \in V$,

$$g_s(\boldsymbol{x}'_s, \boldsymbol{x}^{\beta-NE}_{-s}) \leq \beta \cdot g_s(\boldsymbol{x}^{\beta-NE}_s, \boldsymbol{x}^{\beta-NE}_{-s}), \text{ for all feasible } \boldsymbol{x}'_s. \quad (19)$$

Next, we show that a β-approximate PSNE exists when the following property is satisfied:

Definition 6.2 (Cloud Property). All content items are stored in the same designated server node, i.e.,

$$|\mathcal{D}^i| = 1 \text{ and } \mathcal{D}^i = \mathcal{D}^j, \forall i \neq j, i, j \in \mathcal{I}. \quad (20)$$

[10] When $L_i = L_j$, $\forall i, j \in \mathcal{I}$, problem (18) is a max-weight knapsack problem, which is easy to solve and corresponds to the scenario with equal-sized items in Sections 3–5.

[11] An alternative notion of approximate PSNE (see, e.g., [11]) is based on an additive error, rather than the multiplicative error. Our definition is equally natural, and indeed more in line with the notion of price of anarchy in game theory [7, 31, 43].

Algorithm 2: Find β-Approximate PSNE

Input: $G(V, E)$, $\mathcal{I}$, w_{uv}, $\forall(u, v) \in E$, $\lambda_{(s,i)}$ and $p^{(s,i)}$, for all $s \in V$, $i \in \mathcal{I}$
Output: $\boldsymbol{x}^{\beta-NE}$

1 Classify nodes into sets $\mathcal{V}^m$ for $0 \le m \le |V|-2$;
2 **for** $m = 0 : |V|-2$ **do**
3 **for** $s \in \mathcal{V}^m$ **do**
4 Set $\hat{\boldsymbol{x}}_s = \boldsymbol{0}$, i.e., $\hat{x}_{si} = 0$, $\forall i \in \mathcal{I}$;
5 Relax problem (18) to a linear programming problem by relaxing $\boldsymbol{x}_s \in \{0,1\}^{|\mathcal{I}|}$ to $\widetilde{\boldsymbol{x}}_s \in [0,1]^{|\mathcal{I}|}$;
6 Compute an optimal solution $\widetilde{\boldsymbol{x}}_s^*$ of the LP-relaxation;
7 Set $I_s = \{i : \widetilde{x}_{si}^* = 1\}$ and $F_s = \{i : 0 < \widetilde{x}_{si}^* < 1\}$;
8 **if** $\sum_{i\in I_s} q_{(s,i)}(\boldsymbol{x}_{-s}) > \max_{i\in F_s} q_{(s,i)}(\boldsymbol{x}_{-s})$ **then**
9 Set $\hat{x}_{si} = 1$, $\forall i \in I_s$;
10 **else**
11 Set $\hat{x}_{sj} = 1$ for $j = \arg\max_{i\in F_s} q_{(s,i)}(\boldsymbol{x}_{-s})$;
12 **end**
13 **end**
14 **end**
15 Set $\boldsymbol{x}^{\beta-NE} = \hat{\boldsymbol{x}}$;

Furthermore, for each cache node $s \in V$, its request forwarding path for different content items is the same, i.e.,

$$p^{(s,i)} = p^s, \forall i \in \mathcal{I}. \tag{21}$$

In practice, a network in which all content items are stored in the cloud server satisfies (20). Note that (21) is naturally satisfied when each node chooses the unique shortest path to fetch content items. Furthermore, (21) is naturally satisfied in a tree topology.

In the following, we show that a β-approximate PSNE exists in Game 2 with the cloud property and the path overlap property in (11).[12] Note that in the caching graph satisfying the cloud property and the path overlap property, there is no mixed request loop.

Theorem 6.3. *A β-approximate PSNE always exists in Game 2 with the cloud property and the path overlap property.*

Proof. See Appendix G in the technical report [29]. □

The result holds in arbitrary-topology networks with heterogeneous content popularity. However, the complexity for finding an approximate PSNE may grow exponentially with the number of nodes and their strategies in general.

6.3 Polynomial-Time Algorithm to Find an Approximate PSNE

In this section, we propose a polynomial-time algorithm to find an approximate PSNE of Game 2.

With the cloud property in (20) and (21), and given designated server node u, the caching gain of each cache node $s \in V$ depends on not only $\boldsymbol{x}_s$ but also $\{\boldsymbol{x}_v : v \in V_s\}$, where $V_s = \{v : v \in p^s, v \ne s, v \ne u\}$ is the set of intermediate nodes on node s' request forwarding path p^s. We group nodes with the same number of intermediate nodes into one set, i.e., denote the set of nodes with

[12] The cloud property and the path overlap property are sufficient conditions for existence. Analyzing the sufficient and necessary conditions for existence of (approximate) PSNE on graphs is an open problem, and we will consider it in the future work.

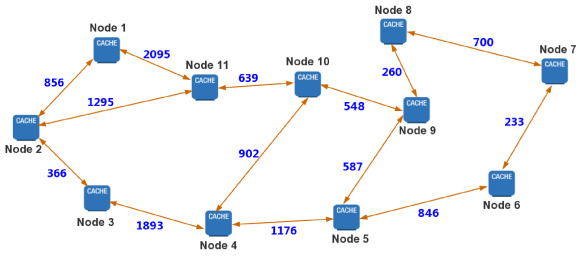

Figure 6: Abilene network.

$|V_s| = m$ by $\mathcal{V}^m = \{s \in V : |V_s| = m\}$ where $0 \le m \le |V|-2$. Note that since node u stores all content items, its caching strategy does not affect other nodes.

If the path overlap property in (11) is satisfied, we know that if node $v \in V_s$, then $s \notin V_v$. That is, if node v in on path p^s, then node s is not on path p^v. Hence, for each node $s \in \mathcal{V}^m$ with $|V_s| = m$ intermediate nodes in set $V_s = \{v : v \in p^s, v \ne s, v \ne u\}$, every intermediate node $v \in V_s$ has a smaller number of intermediate nodes, i.e., $|V_v| < m$. This motivates us to find the equilibrium strategies for nodes in sets $\mathcal{V}^m$, $0 \le m \le |V|-2$, according to the increasing order of m.

We propose a polynomial-time algorithm (Algorithm 2) to find a β-approximate PSNE. Specifically, We find the equilibrium strategies of nodes in sets $\mathcal{V}^m$ for $0 \le m \le |V|-2$ sequentially (Lines 1-2 of Algorithm 2). For example, for node $s \in \mathcal{V}^0$ such that $V_s = \emptyset$ (Line 3 of Algorithm 2), its β-approximate equilibrium strategy $\boldsymbol{x}_s^{\beta-NE}$ is the β-approximate solution to problem (18), calculated by Lines 4-12 of Algorithm 2. Note that for a node $s \in \mathcal{V}^0$, $q_{si}(\boldsymbol{x}_{-s}) = q_{si}$ is a constant independent of other nodes' strategies. For node $s \in \mathcal{V}^m$ with $1 \le m \le |V|-2$ (Line 3 of Algorithm 2), its equilibrium strategy is the β-approximate solution to problem (18), calculated by Lines 4-12 of Algorithm 2. Note that its equilibrium strategy depends only on the caching strategies of nodes $v \in V_s$ in its intermediate node set with $|V_v| < m$, and hence $q_{si}(\boldsymbol{x}_{-s}) = q_{si}(\{\boldsymbol{x}_v^{\beta-NE} : v \in V_s\})$. We continue this sequential process until all nodes decide their β-approximate equilibrium caching strategies. The resulting caching strategy profile is a β-approximate PSNE of Game 2.

In the following theorem, we show that Algorithm 2 can find a β-approximate PSNE of Game 2 in polynomial time.

Theorem 6.4. *For Game 2 with the cloud property and the path overlap property, Algorithm 2 computes a β-approximate PSNE in $O(|V||\mathcal{I}|)$ time.*

Proof. See Appendix H in the technical report [29]. □

However, the approximate PSNE does not in general optimize the social welfare. We next analyze the PoA of Game 2.

6.4 Price of Anarchy

we show that the PoA for the β-approximate PSNE is bounded under the homogeneous request pattern property in (10).

Theorem 6.5. *For Game 2 with the cloud property, the path overlap property, and the homogeneous request pattern property, the PoA for*

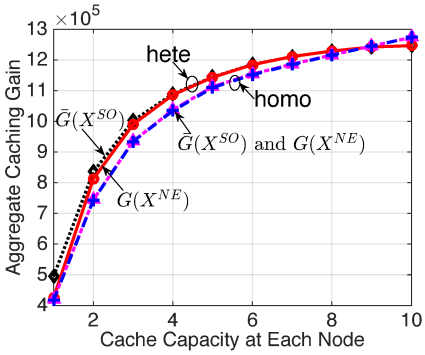

Figure 7: $G(\cdot)$ vs. c_v, under both heterogeneous and homogeneous request patterns.

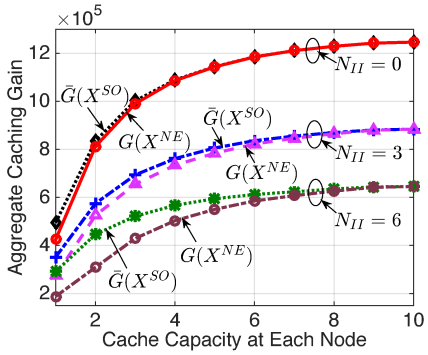

Figure 8: $G(\cdot)$ vs. c_v, under different no. of Type-II nodes N_{II}.

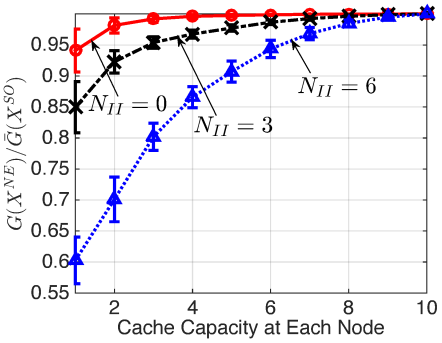

Figure 9: $G(\boldsymbol{x}^{\mathrm{NE}})/\bar{G}(\boldsymbol{x}^{\mathrm{SO}})$ vs. c_v, under different N_{II}, for 100 trials.

the β-approximate PSNE satisfies

$$PoA^{\beta} \geq \frac{1}{1+\alpha\cdot\beta} = \frac{1}{1+\beta\cdot\left(\max_{v\in V, i\in\mathcal{I}}|p^{(v,i)}|-1\right)}. \quad (22)$$

Proof. See Appendix I in the technical report [29]. □

Due to each cache node's limited computational capability, the performance guarantee for the approximate PSNE derived in (22) is worse than the one derived in (13) for the equal-sized item case.

7 SIMULATION RESULTS

We perform simulations on the Abilene network shown in Figure 6. Simulation results show that the performance of Nash equilibria improves with the cache capacity at each node, while it degrades with the number of nodes that do not generate content requests. Furthermore, adding extra cache nodes to the existing network can make the performance of Nash equilibria worse.

Upper Bound of the Optimal Social Welfare. Since the social welfare maximization problem (4) is NP-hard, we calculate an upper bound for the optimal social welfare. Specifically, we relax problem (4) by relaxing the binary caching strategy $\boldsymbol{x} = \{x_{si} \in \{0,1\} : \forall s \in V, i \in \mathcal{I}\}$ to be a continuous caching probability strategy $\boldsymbol{\phi} = \{\phi_{si} \in [0,1] : \forall s \in V, i \in \mathcal{I}\}$ where $\sum_{i\in\mathcal{I}}\phi_{si} \leq c_s, \forall s \in V$, while keeping the objective function unchanged. The relaxed problem is

$$\max G(\boldsymbol{\phi}) \text{ s.t. } \textstyle\sum_{i\in\mathcal{I}}\phi_{si} \leq c_s,\ \phi_{si}\in[0,1],\ \forall s\in V, i\in\mathcal{I}. \quad (23)$$

The relaxation objective function $G(\boldsymbol{\phi})$ is not concave, so (23) is not a convex optimization problem. We approximate $G(\boldsymbol{\phi})$ by $L(\boldsymbol{\phi})$ below [20]:

$$L(\boldsymbol{\phi}) = \textstyle\sum_{s\in V, i\in\mathcal{I}}\lambda_{(s,i)}\sum_{k=1}^{|p^{(s,i)}|-1} w_{p_{k+1}p_k}\min\left\{1, \sum_{k'=1}^{k}\phi_{p_{k'}i}\right\}.$$

Note that $L(\boldsymbol{\phi})$ is concave, and we can solve the following convex optimization problem in polynomial time.

$$\max L(\boldsymbol{\phi}) \text{ s.t. } \textstyle\sum_{i\in\mathcal{I}}\phi_{si} \leq c_s,\ \phi_{si}\in[0,1],\ \forall s\in V, i\in\mathcal{I}. \quad (24)$$

We have the following result:

Lemma 7.1. *Let $\boldsymbol{x}^*$, $\boldsymbol{\phi}^*$, and $\boldsymbol{\phi}^{**}$ be the optimal solutions to problems* (4), (23), *and* (24), *respective. Then:*

$$G(\boldsymbol{x}^*) \leq G(\boldsymbol{\phi}^*) \leq L(\boldsymbol{\phi}^*) \leq L(\boldsymbol{\phi}^{**}). \quad (25)$$

Proof. See Appendix J in the technical report [29]. □

Hence, $L(\boldsymbol{\phi}^{**})$ serves as an upper bound for the optimal social welfare $G(\boldsymbol{x}^*)$. In the following, we define $\bar{G}(\boldsymbol{x}^{\mathrm{SO}}) = L(\boldsymbol{\phi}^{**})$.[13]

Experiment Setup. For the Abilene network shown in Figure 6, we take all edge costs from the Abilene network configuration [27].[14] We consider a set $\mathcal{I} = \{1,\ldots,10\}$ of content items, where node 1 is the designated server of the first 6 content items and node 2 is the designated server of the remaining 4 content items. Each node chooses the shortest path to fetch every content item, following which there is no mixed request loop on the graph. We generate the arrival rates $\lambda_{(s,i)}, \forall s \in V, i \in \mathcal{I}$ uniformly at random in the interval $[0, 10]$.

Results. Figure 7 shows the aggregate caching gain $G(\boldsymbol{x}^{\mathrm{NE}})$ and $\bar{G}(\boldsymbol{x}^{\mathrm{SO}})$ under different cache capacities at each node, for the case with heterogeneous request patterns $\lambda_{(s,i)}$ (the upper two curves) and for the case with homogeneous request patterns $\lambda_{(s,i)} = \lambda_i, \forall s \in V, i \in \mathcal{I}$ (the lower two curves)[15], respectively. We can see that the gap between $G(\boldsymbol{x}^{\mathrm{NE}})$ and $\bar{G}(\boldsymbol{x}^{\mathrm{SO}})$ under homogeneous λ_i is smaller than the gap under heterogeneous $\lambda_{(s,i)}$. Thus, the homogeneous request pattern leads to better performance achieved by selfish caching behaviors in the Abilene network.

In practice, some cache nodes are intermediate routers which do not request for any content items. We define nodes with positive request rates as Type-I nodes (with a total number N_I), and nodes with no request as Type-II nodes (with a total number N_{II}). We show the impact of N_{II} in Figure 8. We can see that the gap between $G(\boldsymbol{x}^{\mathrm{NE}})$ and $\bar{G}(\boldsymbol{x}^{\mathrm{SO}})$ decreases with the cache capacity at each node, while the gap increases with N_{II}. This implies that the impact of the selfish behaviors is mitigated when the cache resource increases, and the selfish behaviors of Type-II nodes degrade the (relative) performance of Nash equilibria (since the selfish Type-II nodes will not cache content items at equilibrium).

To understand the impact of the randomness of request arrival rates $\lambda_{(s,i)}$, we perform simulations on 100 sets of randomly generated $\{\lambda_{(s,i)} : \forall s \in V, i \in \mathcal{I}\}$, and show the average ratios $G(\boldsymbol{x}^{\mathrm{NE}})/\bar{G}(\boldsymbol{x}^{\mathrm{SO}})$ of the 100 trials in Figure 9, where the error bars

[13] The superscript "SO" represents socially optimal.
[14] We assume that the edge costs are symmetric.
[15] We take $\lambda_i = \sum_{s\in V}\lambda_{(s,i)}/V$ given the heterogeneous $\lambda_{(s,i)}, \forall s \in v, i \in \mathcal{I}$.

represent the standard deviations. As is consistent with our observation from Figure 8, the performance of the Nash equilibria increases with the cache capacity, while decreases with N_{II}.

8 CONCLUSION

In this paper, we analyze selfish caching games on directed graphs, which can yield arbitrary bad performance. We show that a PSNE exists and can be found in polynomial time if there is no mixed request loop. We then show that with the homogeneous request pattern property and the path overlap property, the PoA is bounded in arbitrary-topology networks. We further show that the selfish caching game with unequal-sized items admits an approximate PSNE with bounded PoA in special cases.

There are several interesting directions to explore in the future, such as analyzing the sufficient and necessary conditions for existence of the (approximate) PSNE, analyzing the selfish caching games considering the congestion effect on each edge, analyzing the joint caching and routing decisions of selfish nodes though a two-stage game modeling, designing the incentive mechanism to incentivize selfish cache nodes to behave according to the socially optimal strategy, and analyzing the dynamic game between the cache nodes with repeated interactions in the long run considering the tit-for-tat property.

REFERENCES

[1] M. Afanasyev, T. Chen, G. M. Voelker, and A. C. Snoeren. 2010. Usage patterns in an urban WiFi network. *IEEE/ACM TON* 18, 5 (2010), 1359–1372.

[2] M. M. Amble, P. Parag, S. Shakkottai, and L. Ying. 2011. Content-aware caching and traffic management in content distribution networks. In *IEEE INFOCOM*.

[3] W. C. Ao and K. Psounis. 2015. Distributed caching and small cell cooperation for fast content delivery. In *ACM MobiHoc*. 127–136.

[4] S. Basu, A. Sundarrajan, J. Ghaderi, S. Shakkottai, and R. Sitaraman. 2018. Adaptive TTL-Based Caching for Content Delivery. *IEEE/ACM TON* (2018).

[5] S. Borst, V. Gupta, and A. Walid. 2010. Distributed caching algorithms for content distribution networks. In *IEEE INFOCOM*. 1–9.

[6] X. Cao, J. Zhang, and H. V. Poor. 2018. An Optimal Auction Mechanism for Mobile Edge Caching. In *IEEE ICDCS*. 388–399.

[7] S. Chien and A. Sinclair. 2011. Convergence to approximate Nash equilibria in congestion games. *Games and Economic Behavior* 71, 2 (2011), 315–327.

[8] W. Chu, M. Dehghan, D. Towsley, and Z.-L. Zhang. 2016. On allocating cache resources to content providers. In *ACM ICN*. 154–159.

[9] B. G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. H. Papadimitriou, and J. Kubiatowicz. 2004. Selfish caching in distributed systems: a game-theoretic analysis. In *ACM PODC*. 21–30.

[10] E. Cohen and S. Shenker. 2002. Replication strategies in unstructured peer-to-peer networks. In *ACM SIGCOMM*, Vol. 32. 177–190.

[11] C. Daskalakis, A. Mehta, and C. Papadimitriou. 2007. Progress in approximate Nash equilibria. In *ACM EC*. 355–358.

[12] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman. 2015. On the complexity of optimal routing and content caching in heterogeneous networks. In *IEEE INFOCOM*. 936–944.

[13] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman. 2015. On the complexity of optimal routing and content caching in heterogeneous networks. In *IEEE INFOCOM*. 936–944.

[14] R. Draves, J. Padhye, and B. Zill. 2004. Routing in multi-radio, multi-hop wireless mesh networks. In *ACM MobiCom*. 114–128.

[15] M. Garetto, E. Leonardi, and S. Traverso. 2015. Efficient analysis of caching strategies under dynamic content popularity. In *IEEE INFOCOM*. 2263–2271.

[16] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash. 2016. A provably efficient online collaborative caching algorithm for multicell-coordinated systems. *IEEE TMC* 15, 8 (2016), 1863–1876.

[17] M. Goemans, L. E. Li, V. S. Mirrokni, and M. Thottan. 2004. Market sharing games applied to content distribution in ad-hoc networks. In *ACM MobiHoc*. 55–66.

[18] R. Gopalakrishnan, D. Kanoulas, N. N. Karuturi, C. P. Rangan, R. Rajaraman, and R. Sundaram. 2012. Cache me if you can: capacitated selfish replication games. In *Latin American Symposium on Theoretical Informatics*. Springer, 420–432.

[19] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. 2012. Confused, timid, and unstable: picking a video streaming rate is hard. In *Proceedings of the 2012 Internet Measurement Conference*. ACM, 225–238.

[20] S. Ioannidis and E. Yeh. 2016. Adaptive caching networks with optimality guarantees. In *ACM SIGMETRICS*, Vol. 44. 113–124.

[21] A. T. Ip, J. Lui, and J. Liu. 2008. A revenue-rewarding scheme of providing incentive for cooperative proxy caching for media streaming systems. *ACM TOMM* 4, 1 (2008), 5.

[22] B. Jiang, P. Nain, and D. Towsley. 2018. On the Convergence of the TTL Approximation for an LRU Cache under Independent Stationary Request Processes. *ACM TOMPECS* 3, 4 (2018), 20.

[23] Li-Shen Juhn and Li-Ming Tseng. 1997. Harmonic broadcasting for video-on-demand service. *IEEE transactions on broadcasting* 43, 3 (1997), 268–271.

[24] H. Kellerer, U. Pferschy, and D. Pisinger. 2004. Introduction to NP-Completeness of knapsack problems. In *Knapsack problems*. Springer, 483–493.

[25] J. Kwak, Y. Kim, L. B. Le, and S. Chong. 2018. Hybrid content caching in 5G wireless networks: Cloud versus edge caching. *IEEE TWC* 17, 5 (2018), 3030–3045.

[26] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis. 2006. Distributed selfish replication. *IEEE TPDS* 17, 12 (2006), 1401–1413.

[27] A. Li, X. Yang, and D. Wetherall. 2009. SafeGuard: safe forwarding during route changes. In *PACM CoNEXT*. 301–312.

[28] X. Li, X. Wang, P.-J. Wan, Z. Han, and V. CM Leung. 2018. Hierarchical Edge Caching in Device-to-Device Aided Mobile Networks: Modeling, Optimization, and Design. *IEEE JSAC* (2018).

[29] Q. Ma, E. Yeh, and J. Huang. 2018. Online technical report. http://jianwei.ie.cuhk.edu.hk/publication/AppendixSelfishCacheMobihoc19.pdf

[30] P. Maillé, G. Simon, and B. Tuffin. 2015. Impact of revenue-driven CDN on the competition among network operators. In *IEEE CNSM*. 163–167.

[31] C. Papadimitriou. 2001. Algorithms, games, and the internet. In *ACM STOC*. 749–753.

[32] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire. 2018. The role of caching in future communication systems and networks. *arXiv preprint arXiv:1805.11721* (2018).

[33] G. G. Pollatos, O. A. Telelis, and V. Zissimopoulos. 2008. On the social cost of distributed selfish content replication. In *International Conference on Research in Networking*. Springer, 195–206.

[34] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas. 2018. Distributed Caching Algorithms in the Realm of Layered Video Streaming. *IEEE TMC* (2018).

[35] Z. Qin, X. Gan, L. Fu, X. Di, J. Tian, and X. Wang. 2018. Content Delivery in Cache-enabled Wireless Evolving Social Networks. *IEEE TWC* (2018).

[36] P. Rahimzadeh, C. Joe-Wong, K. Shin, Y. Im, J. Lee, and S. Ha. 2017. SVC-TChain: Incentivizing good behavior in layered P2P video streaming. In *IEEE INFOCOM*. 1–9.

[37] T. Roughgarden and É. Tardos. 2002. How bad is selfish routing? *J. ACM* 49, 2 (2002), 236–259.

[38] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire. 2013. Femtocaching: Wireless content delivery through distributed caching helpers. *IEEE TIT* 59, 12 (2013), 8402–8413.

[39] K. Shin, C. Joe-Wong, S. Ha, Y. Yi, I. Rhee, and D. S. Reeves. 2017. T-chain: A general incentive scheme for cooperative computing. *IEEE/ACM TON* 25, 4 (2017), 2122–2137.

[40] S. Shukla, O. Bhardwaj, A. A. Abouzeid, T. Salonidis, and T. He. 2017. Hold'em Caching: Proactive Retention-Aware Caching with Multi-path Routing for Wireless Edge Networks. In *ACM MobiHoc*. 24.

[41] J. Tadrous, A. Eryilmaz, and H. El Gamal. 2016. Joint smart pricing and proactive content caching for mobile services. *IEEE/ACM TON* 24, 4 (2016), 2357–2371.

[42] D. Vega, L. Cerda-Alabern, L. Navarro, and R. Meseguer. 2012. Topology patterns of a community network: Guifi. net. In *IEEE WiMob*. 612–619.

[43] A. Vetta. 2002. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *IEEE FOCS*. 416–425.

[44] L. Wang, G. Tyson, J. Kangasharju, and J. Crowcroft. 2017. Milking the cache cow with fairness in mind. *IEEE/ACM TON* 25, 5 (2017), 2686–2700.

[45] Yi-Kai Wang, Yitong Yin, and Sheng Zhong. 2014. Belief propagation for spatial spectrum access games. In *Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 225–234.

[46] E. Yeh, T. Ho, Y. Cui, M. Burd, R. Liu, and D. Leong. 2014. Vip: A framework for joint dynamic forwarding and caching in named data networks. In *ACM ICN*. 117–126.

[47] R. Yu, S. Qin, M. Bennis, X. Chen, G. Feng, Z. Han, and G. Xue. 2016. Enhancing software-defined RAN with collaborative caching and scalable video coding. In *IEEE ICC*. 1–6.

[48] J. Zhang, X. Lin, and X. Wang. 2018. Coded caching under arbitrary popularity distributions. *IEEE TIT* 64, 1 (2018), 349–366.

[49] T. Zhao, I-H. Hou, S. Wang, and K. Chan. 2018. ReD/LeD: An asymptotically optimal and scalable online algorithm for service caching at the edge. *IEEE JSAC* 36, 8 (2018), 1857–1870.

[50] X. Zhao, P. Yuan, and S. Tang. 2018. Collaborative edge caching in context-aware device-to-device networks. *IEEE TVT* 67, 10 (2018), 9583–9596.