Parameterized Complexity of Caching in Networks

Robert Ganian[1], **Fionn Mc Inerney**[2], **Dimitra Tsigkari**[2]

[1]Algorithms and Complexity Group, TU Wien, Vienna, Austria
[2]Telefónica Scientific Research, Barcelona, Spain
rganian@gmail.com, fmcinern@gmail.com, dimitra.tsigkari@telefonica.com

Abstract

The fundamental caching problem in networks asks to find an allocation of contents to a network of caches with the aim of maximizing the cache hit rate. Despite the problem's importance to a variety of research areas—including not only content delivery, but also edge intelligence and inference—and the extensive body of work on empirical aspects of caching, very little is known about the exact boundaries of tractability for the problem beyond its general NP-hardness. We close this gap by performing a comprehensive complexity-theoretic analysis of the problem through the lens of the parameterized complexity paradigm, which is designed to provide more precise statements regarding algorithmic tractability than classical complexity. Our results include algorithmic lower and upper bounds which together establish the conditions under which the caching problem becomes tractable.

1 Introduction

Caching is one of the most important tasks that need to be handled by modern-day content delivery networks (CDNs), and its role is expected to grow even further in the future, e.g., in the context of wireless communications (Liu et al. 2016; Paschos et al. 2018) and artificial intelligence. For the latter, it is crucial to cache: trained models for inference requests (Salem et al. 2023; Zhu et al. 2023; Yu et al. 2024), information that can accelerate the training of large models (Lindgren et al. 2021; Zhang et al. 2024), and trained model parts in distributed learning paradigms (Thapa et al. 2022; Tirana et al. 2024). Typically, in a caching network, one needs to decide where to store the contents in order to maximize metrics related to, e.g., network performance or user experience, with cache hit rate being the most predominant one (Paschos, Iosifidis, and Caire 2020). While many recent works have investigated dynamic caching policies that either decide on caching before time-slotted requests or via the eviction policy (Paschos et al. 2019; Bhattacharjee, Banerjee, and Sinha 2020; Rohatgi 2020; Paria and Sinha 2021; Mhaisen et al. 2022), in this article we focus on proactive caching, which decides how to fill the caches based on anticipated requests. Proactive caching is of crucial importance in edge caching (Bastug, Bennis, and Debbah 2014; Tadrous and Eryilmaz 2015) and is applied in contemporary caching architectures like Netflix's CDN (Netflix 2024b). In practice, content requests in video streaming services can be steered through trends or recommendation systems, leading to accurate predictions on the content popularity (Zhou, Khemmarat, and Gao 2010; Netflix Tech Blog 2016).

In the proactive setting, the caching problem in a network of caches (with possibly overlapping coverage) can be naturally represented as a bipartite graph between the set of users and the network of caches (of limited capacities), a set of contents, and information on the users' anticipated requests; the task is to allocate the contents to caches to maximize the cache hit rate. This problem was proven to be NP-hard in the seminal work that formalized this setting (Shanmugam et al. 2013). The subsequent numerous variants of this base problem (Poularakis, Iosifidis, and Tassiulas 2014; Dehghan et al. 2016; Krolikowski, Giovanidis, and Di Renzo 2018; Ricardo et al. 2021; Tsigkari and Spyropoulos 2022) were all shown to be NP-hard by a reduction from it.

In the above works, the theoretical intractability was typically overcome via the use of heuristics or approximation algorithms. In fact, despite the extensive body of work on the caching problem, our understanding of its foundational complexity-theoretic aspects is still in its infancy. Specifically, very little is known about the problem's "boundaries of tractability", i.e., the precise conditions under which it becomes tractable. This gap is in stark contrast with recent substantial advances made in this direction for problems arising in artificial intelligence and machine learning such as Bayesian network learning (Ordyniak and Szeider 2013; Ganian and Korchemna 2021; Grüttemeier and Komusiewicz 2022), data completion (Ganian et al. 2018, 2020; Eiben et al. 2021), and resource allocation (Bliem, Bredereck, and Niedermeier 2016; Deligkas et al. 2021; Eiben et al. 2023).

The aim of this article is to close the aforementioned gap by carrying out a comprehensive study of the complexity-theoretic aspects of caching in its modern network-based formalization through the lens of *parameterized complexity* (Downey and Fellows 2013; Cygan et al. 2015), which provides more refined tools to identify the conditions under which an NP-hard problem becomes tractable when compared to classical complexity. In the parameterized setting, the running times of algorithms are studied with respect to the input size n, but also an integer parameter k, which intuitively captures well-defined structural properties of the

input. The most desirable outcome here is *fixed-parameter tractability* (FPT), which means that the problem can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ for a computable function f. A weaker notion of this is XP-tractability, which means that the problem can be solved in time $n^{f(k)}$. The least favorable outcome is paraNP-hardness, which means that the problem remains NP-hard even when k is a constant. Lastly, excluding fixed-parameter tractability is done by proving the problem is W[1]-*hard* via a parameterized reduction.

Our Contributions. We consider three fundamental variants of the NETWORK-CACHING problem, which differ solely on the modeling and encoding of the content sizes. The main challenges of our investigation arise from the various aspects of the problem: when solving an instance one needs to (1) adhere to (knapsack-like) capacity constraints while (2) taking into account the graph structure of the caching network, and also (3) deal with the large variety of naturally occurring parameterizations. In particular, we study the complexity of these variants with respect to six parameters: the number C of caches, the maximum capacity K of any cache, the number S of contents, the number U of users, the maximum degree Δ of the network, and the maximum number λ of contents that any user may request.[1]

We first consider the most computationally challenging variant of the problem, where the content and cache sizes are encoded in their "exact form" (as binary-encoded numbers). Interestingly, the inherent hardness of this variant—denoted HETNC-B—makes it easier to perform a detailed analysis compared to the two other variants. Note that the parameter K is incompatible with this model (bounding the cache capacities translates the problem to the unary-encoded setting investigated later). We begin by establishing the problem is FPT under the combined parameterizations[2] of $C+S$ (Theorem 1) and $U+S$ (or the complexity-theoretically equivalent $U+\lambda$) (Corollary 2); as HETNC-B is the most general of the three models, these tractability results carry over to the other two. Then, we show that HETNC-B is paraNP-hard under all remaining parameterizations (Theorems 7 and 10 and Corollary 11). The clean cut between tractability and intractability for HETNC-B is depicted in Fig. 1 (top).

Next, we consider the unary variant of the problem, where one still distinguishes between contents of different sizes, but only on a more categorical scale (e.g., by distinguishing between full movies, longer shows, and shorter shows in the setting of video streaming). Here, the sizes of the contents are encoded in unary, which affects the complexity of the associated HETNC-U problem. On the positive side, in addition to $C+S$ and $U+S$, we identify a third fixed-parameter tractable fragment of the problem, specifically $C+K$ (Theorem 3). Moreover, we obtain XP-tractability for HETNC-U with respect to C alone (Theorem 3), and complement this result by establishing W[1]-hardness (and hence, ruling out fixed-parameter tractability) even when combining C with

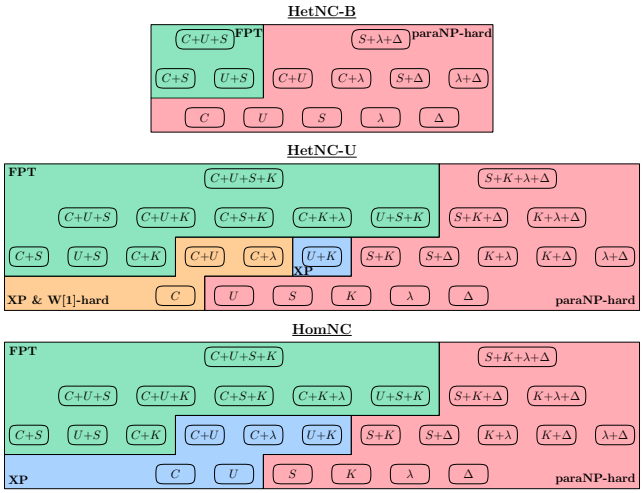

Figure 1: Complexity landscapes of HETNC-B (top), HETNC-U (middle), and HOMNC (bottom). We (mostly) omit parameterizations including $C+\Delta$, $U+\Delta$, $U+\lambda$, or $S+\lambda$, as the first two, the third, and the last are complexity-theoretically equivalent to $C+U$, $U+S$, and S, respectively. Indeed, for any pair of combined parameters claimed to be equivalent, each of them can be bounded by a computable function of the other. To prove the first two equivalences, note that $\Delta \leq C+U$, $U \leq C \cdot \Delta$, and $C \leq U \cdot \Delta$. For the latter two, note that $S \leq U \cdot \lambda$ and $\lambda \leq S$.

either U or λ (Theorem 8 and Corollary 9). All fragments not covered by these cases are then paraNP-hard (Theorems 7 and 8), except for $U+K$ for which we establish XP-tractability (Theorem 4). The obtained complexity landscape of HETNC-U is illustrated in Fig. 1 (middle).

Finally, we turn to the simplest, totally uniform case where each content has the same size; we denote the associated problem HOMNC. We first demonstrate the challenging nature of even this variant by strengthening the previously known lower bound (Shanmugam et al. 2013) and establishing NP-hardness when $S=\lambda=2$, $K=1$, and, additionally, $\Delta=3$ (Theorem 7). Furthermore, unlike for HETNC-U, here we establish XP-tractability with respect to U alone (Corollary 6). For the lower bounds, all paraNP-hardness results, other than with respect to U alone, can be shown to carry over to the simpler setting of HOMNC. Our results for HOMNC are displayed in Fig. 1 (bottom).

To complete the complexity landscapes for HETNC-U and HOMNC, it remains to complement some of the obtained XP algorithms (those depicted in blue in Fig. 1) with W[1]-hardness results to exclude fixed-parameter tractability. In this respect, we show that resolving any of the open cases for HOMNC by establishing W[1]-hardness (which we conjecture to hold) immediately implies the W[1]-hardness of all the open cases depicted in Fig. 1. In fact, we show that all 5 "open" parameterizations for HOMNC are complexity-theoretically equivalent to each other (Theorem 14). Hence, while settling these 5 open questions seems to require novel ideas and techniques beyond today's state of the art, resolv-

[1]The problem definitions and motivation for the employed parameterizations are deferred to Section 2.

[2]Parameterizing by a set of parameters is equivalent to parameterizing by their sum (Cygan et al. 2015).

ing any of them settles the complexity of the others.

As our final contribution, we complement the complexity-theoretic landscapes depicted in Fig. 1 by considering whether tractability can be achieved by exploiting finer structural properties of the network. Indeed, for graph-theoretic problems, it is typical to also consider structural parameterizations of the input graph such as their *treewidth* (Robertson and Seymour 1986), *treedepth* (Nesetril and de Mendez 2012), *feedback edge number* (Ganian and Korchemna 2021; Bredereck et al. 2022), or *vertex cover number* (Bodlaender, Groenland, and Pilipczuk 2023; Chalopin et al. 2024). It is also common to study such problems when the graph is restricted to being *planar*, i.e., it can be drawn on a plane such that none of its edges cross. This is particularly important for NETWORK-CACHING, as planar networks will have much less overlapping cache coverage. As our last set of results, we classify the complexity of HETNC-B, HETNC-U, and HOMNC with respect to the above structural restrictions; in particular, we show that almost none of these restrictions help achieve tractability, even when combined with those considered in Fig. 1.

2 Setup and Problem Definitions

We use standard graph-theoretic terminology (Diestel 2012). For any positive integer n, let $[n] = \{1, \dots, n\}$. We consider a set $\mathcal{C}$ of $C := |\mathcal{C}|$ caches and a set $\mathcal{U}$ of $U := |\mathcal{U}|$ users, each of which has access to a subset of caches (which could be determined by, e.g., routing or other network policies). This naturally defines a bipartite graph $G = (\mathcal{C}, \mathcal{U}, \mathcal{E})$, where $\mathcal{E}$ are the edges between the two independent sets $\mathcal{C}$ and $\mathcal{U}$, as depicted in Fig. 2. This is the predominant setting in the literature (Shanmugam et al. 2013; Paria and Sinha 2021; Mhaisen et al. 2022; Tsigkari and Spyropoulos 2022; Salem et al. 2023). For all $u \in \mathcal{U}$, the subset of caches that u has access to is the neighborhood of u, denoted by $N(u)$. For all $c \in C$, the subset of users that c serves is $N(c)$. The maximum degree of the network is $\Delta := \max_{v \in V(G)} |N(v)|$. The users have access to a catalog $\mathcal{S}$ of $S := |\mathcal{S}|$ contents, where the size of a content $s \in \mathcal{S}$ is denoted by $\sigma(s)$; as is common in the literature (Shanmugam et al. 2013; Blaszczyszyn and Giovanidis 2015; Paschos et al. 2019) and, without loss of generality, the content sizes are positive integers. In our analysis, we consider both the cases of contents of equal (homogeneous) and of variable (heterogeneous) sizes, where the former is equivalent to rescaling and fixing $\sigma(s) = 1$ for all $s \in \mathcal{S}$.

For each cache $c \in \mathcal{C}$, we denote by $\kappa(c) \in \mathbb{Z}^+$ its capacity, and we denote the maximum capacity of a cache (over all caches) by $K := \max_{c \in \mathcal{C}} \kappa(c)$. As a cache with capacity equal to $\sum_{s \in \mathcal{S}} \sigma(s)$ can store all the contents in $\mathcal{S}$, without loss of generality, we assume that $K \leq \sum_{s \in \mathcal{S}} \sigma(s)$, which, in the case of HOMNC, implies that $K \leq S$. However, in real systems, K is far from this upper bound. Indeed, in the case of Netflix, the cache capacity may be as little as 0.3% of the catalog size (Paschos et al. 2016; Netflix 2024a).

A user requests contents following a probability distribution. Specifically, a user $u \in \mathcal{U}$ requests content $s \in \mathcal{S}$ with probability $p_{us} \in [0, 1]$, such that $\sum_{s \in \mathcal{S}} p_{us} = 1$ for all

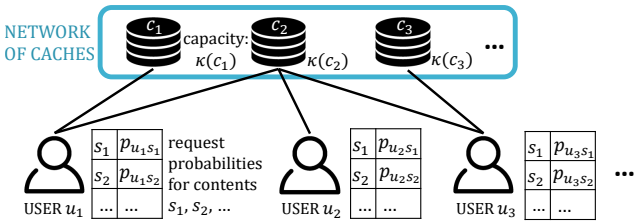

Figure 2: Illustration of a bipartite graph between users and subsets of the caches in our setting.

$u \in \mathcal{U}$. The values p_{us} are considered to be known and represented as rational numbers; these may be based on past user requests, trending contents, etc. We denote by λ the maximum number of contents any user may request, i.e., $\lambda := \max_{u \in \mathcal{U}} |\{s \in \mathcal{S} | \ p_{us} > 0\}|$. A request for content s by user u is served by a cache adjacent to user u where the requested content is stored. If the content is not stored in any of the adjacent caches, then the request is served by a large cache outside of $\mathcal{C}$ containing all contents; this will induce traffic at the backhaul network with a potential impact on user quality and retrieval costs (Shanmugam et al. 2013; Poularakis et al. 2014). Thus, it is important to optimize the caching allocation in the "smaller" caches in proximity of the users; these are updated during off-peak hours (Netflix 2024b) and the cache placement is static between two updates. Moreover, each user $u \in \mathcal{U}$ is characterized by a priority weight $w(u) \in \mathbb{Q}^+$ that may, e.g., capture the guarantee of high quality of service (QoS) for premium users in streaming platforms (Disney+ 2023; Netflix 2023).

Let a *caching allocation* $Z : \mathcal{C} \to 2^{\mathcal{S}}$ be a mapping that assigns a subset of contents to each cache. If Z complies with the size constraints of the individual caches, i.e., $\forall c \in \mathcal{C} : \sum_{s \in Z(c)} \sigma(s) \leq \kappa(c)$, then it is *feasible*. For a user $u \in \mathcal{U}$, the set of contents stored in its adjacent small caches (i.e., in $N(u)$) under a caching allocation Z is defined as the set $H(u) := \{s \in \mathcal{S} \mid \exists c \in N(u) : s \in Z(c)\}$. Potential requests for contents in $H(u)$ will induce cache hits. We can now define the *cache hit rate* (Shanmugam et al. 2013; Paschos, Iosifidis, and Caire 2020) achieved by a caching allocation Z to be $CH(Z) := \sum_{u \in \mathcal{U}} \sum_{s \in H(u)} w(u) \cdot p_{us}$.

As is common in complexity-theoretic studies, we consider the decision variant of the problem in question; we note that all of our algorithms can also solve the corresponding optimization problem and are constructive, i.e., they can output the optimal feasible caching allocation. We can now define the problem archetype studied in this paper.

NETWORK-CACHING

Input: A bipartite graph $G = (\mathcal{C}, \mathcal{U}, \mathcal{E})$, the cache capacities $\kappa(c_1) \dots, \kappa(c_C)$, a set $\mathcal{S}$ of contents of sizes $\sigma(s_1), \dots, \sigma(s_S)$, the request probabilities p_{us} for all $u \in \mathcal{U}$ and $s \in \mathcal{S}$, the user weights $w(u)$ for all $u \in \mathcal{U}$, and $\ell \in \mathbb{Q}^+$.

Question: Is there a feasible caching allocation Z such that $CH(Z) \geq \ell$?

As the complexity of NETWORK-CACHING depends on

how the sizes of contents are encoded and represented, we distinguish between the following three variants of it:

- In HOMOGENEOUS NETWORK-CACHING (HOMNC), the sizes of the contents are the same, and thus, without loss of generality, are equal to 1, i.e., $\sigma(s_1) = \ldots = \sigma(s_S) = 1$. This restriction has been considered in past works (Blaszczyszyn and Giovanidis 2015; Paschos et al. 2019), and can be justified on the basis that contents could in practice be partitioned into equal-sized chunks and cached independently, see (Maggi et al. 2018).
- In HETEROGENEOUS NETWORK-CACHING (UNARY) (HETNC-U), the numbers $\sigma(s_1), \ldots, \sigma(s_S)$ are encoded in unary. For our complexity-theoretic analysis, this is equivalent to assuming that the sizes of contents are not much larger than the input size. HETNC captures the situation where we do not represent the sizes of contents by their exact bit size, but rather by a rough categorical scale. As an example from the video-on-demand setting, movies are typically roughly twice as long as 1-hour shows, which are then roughly twice as long as 30-minute shows, and hence, we can represent the sizes of these items by the integers 4, 2, and 1, respectively.
- In HETEROGENEOUS NETWORK-CACHING (BINARY) (HETNC-B), the numbers $\sigma(s_1), \ldots, \sigma(s_S)$ are encoded in binary, meaning that they can be exponentially larger than the input size. This situation may arise, e.g., if one were to simply use the exact sizes of contents.

As mentioned earlier, we study three fundamental variants of NETWORK-CACHING with respect to all combinations of the parameterizations discussed in Section 1. As C, K, S, and U are all natural input parameters of this problem, it is only logical to consider parameterizations by them. On the other hand, we also consider the auxiliary parameters Δ and λ, as one could naturally think that this could lead to tractability. Indeed, if each user only has access to a bounded number of caches and, additionally, each cache only serves a bounded number of users, then, intuitively, this should render the problem significantly easier. Further, if each user only requests (with non-zero probability) a bounded number of contents, then, instinctively, this should also simplify the problem. However, as our first lower bound result (Theorem 7), we surprisingly show that this is not the case.

3 Establishing Upper Bounds: Algorithms

In this section, we provide all the tractability results delimiting the boundaries of tractability depicted in Fig. 1, i.e., algorithms. We first analyze HETNC-B, as any tractability results for this problem also carry over to the other two. We use a direct branching argument to show:

Theorem 1. HETNC-B *is* FPT *parameterized by* $C + S$.

Proof. We apply a brute-force algorithm that computes all the possible caching allocations and finds a feasible one with the largest cache hit rate. For each cache, it branches over the 2^S possibilities of storing contents in that cache. Thus, there are $\mathcal{O}(2^{SC})$ possible caching allocations. For each caching allocation, it takes $\mathcal{O}(C \cdot S)$ time to test if it is feasible and $\mathcal{O}(U \cdot C \cdot S)$ time to compute its cache hit rate. Thus, the algorithm's total runtime is $\mathcal{O}(2^{SC} \cdot C \cdot S \cdot U)$. Its correctness follows since it enumerates all caching allocations. □

With Theorem 1 in hand, we can provide a second fixed-parameter tractable fragment for HETNC-B.

Corollary 2. HETNC-B *is* FPT *parameterized by* $U + S$.

Proof Sketch. We apply a simple preprocessing procedure where, for each set of $S + 1$ or more caches with the same neighborhood in G, we keep the S caches with the largest capacities and delete the rest. This takes $\mathcal{O}(C \log(C) \cdot U)$ time, and results in a graph G'. It remains to show that (G, ℓ) is a yes-instance if and only if (G', ℓ) is. □

We now have all the tractability results needed for Fig. 1 (top). For HETNC-U, we contrast the situation for the binary case by designing a dynamic programming algorithm whose running time is XP when parameterized by C alone, and even FPT when parameterized by $C + K$.

Theorem 3. HETNC-U *is* XP *parameterized by* C *and* FPT *parameterized by* $C + K$.

Proof Sketch. Let $c_1, \ldots, c_C$ be the caches and $s_1, \ldots, s_S$ the contents of the catalog. Let A be a C-dimensional array where, for all $i \in [C]$, the i^{th} dimension of the array has size $\kappa(c_i) + 1$. We use a zero-based indexing for A, i.e., the indices start from 0. Essentially, each entry in A will correspond to the best possible cache hit rate that can be achieved when the caches have the remaining capacities corresponding to the coordinates of the entry, with the value of the i^{th} coordinate corresponding to the remaining capacity of the cache c_i for all $i \in [C]$. Initially, we set all the entries of A to -1 except for one entry that we set to 0, whose coordinates correspond to all the caches having their full capacities remaining (i.e., $A[\kappa(c_1)] \ldots [\kappa(c_C)] = 0$). Essentially, -1 is a placeholder that represents that a caching allocation resulting in the represented cache capacities cannot yet exist, which is the case here as the capacities of the caches cannot be reduced without storing contents in them. Then, we will update the entries of A through a dynamic programming approach considering the contents one by one and all the possibilities of adding the content to the caches. In order to perform this update properly, we also have a second array A' with $A' = A$ initially. The algorithm proceeds as follows.

For $i = 1$ to $i = S$, do the following. For each non-negative entry $A[j_1] \ldots [j_C]$ in A and each of the 2^C subsets $\mathcal{C}' \subseteq \mathcal{C}$, possibly update the entries of A' as follows. Let $CH(\mathcal{C}', s_i)$ be the cache hits obtained only from storing s_i in the caches of $\mathcal{C}'$. For all $t \in [C]$, let $q_t = 1$ if $c_t \in \mathcal{C}'$, and otherwise, let $q_t = 0$. If, for all $t \in [C]$, it holds that $j_t - (q_t \cdot \sigma(s_i)) \geq 0$, and

$$A[j_1] \ldots [j_C] + CH(\mathcal{C}', s_i) >$$

$$A'[j_1 - (q_1 \cdot \sigma(s_i))] \ldots [j_C - (q_C \cdot \sigma(s_i))], \text{then set}$$

$$A'[j_1 - (q_1 \cdot \sigma(s_i))] \ldots [j_C - (q_C \cdot \sigma(s_i))] =$$

$$A[j_1] \ldots [j_C] + CH(\mathcal{C}', s_i).$$

This ends the two innermost For loops. Before incrementing i by 1, set $A = A'$. Once the algorithm is finished, let ℓ' be

the largest entry in A'. Note that ℓ' is the maximum cache hit rate possible among all feasible caching allocations. Hence, we correctly conclude that the input is a yes-instance if and only if $\ell' \geq \ell$. The total runtime is at most $\mathcal{O}((K+1)^C \cdot 2^C \cdot C \cdot U \cdot S)$, and all that remains is to argue correctness. □

The ideas from the above algorithm can be extended to show that HETNC-U is XP parameterized by $U + K$. This is done by grouping caches together into types based on the users they serve and their capacities, and keeping track of the number of caches of each type that are left as we fill them.

Theorem 4. HETNC-U *is* XP *parameterized by* $U + K$.

Proof Sketch. Let $s_1, \ldots, s_S$ be the contents of the catalog. Furthermore, let $\mathcal{U}_1, \ldots, \mathcal{U}_{2^U}$ be the distinct subsets of $\mathcal{U}$ and, for all $r \in [2^U]$, let $\mathcal{C}_r := \{c \mid N(c) = \mathcal{U}_r \text{ and } c \in \mathcal{C}\}$. Further, for all integers $0 \leq y \leq [K]$, let $\mathcal{C}_r^y$ be the set of caches in $\mathcal{C}_r$ whose remaining capacities (after possibly storing some contents) are equal to y. Thus, there are $T := 2^U \cdot (K+1)$ sets of the form $\mathcal{C}_r^y$ or, in other words, "types" of caches. Let A be a T-dimensional array where, for each $i \in [T]$, the i^{th} dimension of the array corresponds to a cache type and has size $C + 1$. Specifically, for all $r \in [2^U]$ and integers $0 \leq y \leq K$, the cache of type $\mathcal{C}_r^y$ corresponds to the $((r-1) \cdot (K+1) + y + 1))^{\text{th}}$ dimension of A, and this dimension's coordinate corresponds to the number of caches of type $\mathcal{C}_r^y$ remaining. We use a zero-based indexing for A, that is, the indices start from 0. Essentially, each entry in A will correspond to the best possible cache hit rate that can be achieved when the numbers of each type of cache remaining correspond to the coordinates of the entry. Initially, we set all of the entries of A to -1 except for one entry that we set to 0, whose coordinates correspond to the initial numbers of each type of cache before any contents are stored in caches. The -1 is a placeholder and plays an analogous role as in the proof of Theorem 3. Then, we will update the entries of A through a dynamic programming approach considering the contents one by one and all the possibilities of adding the content to types of caches. In order to do this update properly, we will also have a second array A' with $A' = A$ initially. The algorithm proceeds as follows.

For $i = 1$ to $i = S$, do the following. For each non-negative entry $A[j_1] \ldots [j_T]$ in A and each of the 2^T subsets of $\mathcal{C}' \subseteq \{\mathcal{C}_r^y \mid r \in [2^U] \text{ and } 0 \leq y \leq K\}$, possibly update the entries of A' as follows. Let $CH(\mathcal{C}', s_i)$ be the cache hits obtained only from storing s_i in the types of caches of $\mathcal{C}'$. When storing s_i in the types of caches of $\mathcal{C}'$, this changes the number of certain types of caches remaining, possibly increasing or decreasing it for some. To simplify matters, without loss of generality, from $j_1, \ldots, j_T$, after storing the content s_i in the types of caches of $\mathcal{C}'$, let $j_1', \ldots, j_T'$ be the resulting numbers of each cache type remaining. If, for all $t \in [T]$, it holds that $j_t' \geq 0$, and

$$A[j_1] \ldots [j_T] + CH(\mathcal{C}', s_i) > A'[j_1'] \ldots [j_T'], \text{ then set}$$

$$A'[j_1'] \ldots [j_T'] = A[j_1] \ldots [j_T] + CH(\mathcal{C}', s_i).$$

This ends the two innermost For loops. Before incrementing i by 1, set $A = A'$. Once the algorithm is finished, let ℓ' be the largest entry in A'. Note that ℓ' is the maximum cache hit rate possible among all feasible caching allocations. Hence, we correctly conclude that the input is a yes-instance if and only if $\ell' \geq \ell$. The total runtime of the algorithm is at most $\mathcal{O}((C+1)^T \cdot 2^T \cdot (C \cdot U + T) \cdot S) = \mathcal{O}((C+1)^{f(U,K)} \cdot S)$, for some computable function f. □

We now have all the tractability results we need to establish Fig. 1 (middle): the upper bounds follow from Theorems 1, 3, and 4 plus Corollary 2. Finally, we establish that HOMNC is XP parameterized by U through the following observation that allows us to bound the number of caches by a function of the number of users, after which the XP algorithm provided in Theorem 3 can be applied. The reason why this only works for HOMNC is that it amalgamates caches with the same neighborhood into a single cache with a larger capacity, which cannot be safely done (i.e., without changing the outcome of the problem) in the other settings due to the variable content sizes.

Observation 5. *Any instance* (G, ℓ) *of* HOMNC *with* U *users can be reduced in polynomial time to an equivalent instance* (G', ℓ) *of* HOMNC *with at most* 2^U *caches.*

Theorem 3 together with Observation 5 immediately yields the following corollary.

Corollary 6. HOMNC *is* XP *parameterized by* U.

Corollary 6 is the only additional tractability result needed for the landscape in Fig. 1 (bottom).

4 Establishing Lower Bounds: Hardness

In this section, we establish the lower bounds required for the landscapes depicted in Fig. 1. As we are proving hardness results, it is advantageous to first consider parameterizations of HOMNC as they will carry over to HETNC-U and HETNC-B. To simplify our exposition, we denote all of the instances constructed in our reductions as a bipartite graph G together with the target cache hit rate ℓ.

We first strengthen the known result (Shanmugam et al. 2013) that HOMNC is NP-hard even if $S = \lambda = 2$ and $K = 1$, by showing that the same holds even if additionally restricted to networks where $\Delta = 3$. We prove this via a reduction from the NP-hard MONOTONE NAE-3-SAT-B3 problem (Kratochvíl and Tuza 2002), in which each variable appears in at most 3 clauses and only in its positive form.

Theorem 7. HOMNC *is* NP*-hard, even if* $S = \lambda = 2$*,* $K = 1$*, and* $\Delta = 3$*.*

Proof Sketch. Let $x_1, \ldots, x_n$ and $C_1, \ldots, C_m$ be the variables and clauses, respectively, of an input formula ϕ of MONOTONE NAE-3-SAT-B3. We construct an instance (G, ℓ) of HOMNC from the incidence graph G_ϕ of ϕ as follows. Each variable vertex x_i, $i \in [n]$, in G_ϕ corresponds to a cache c_{x_i} of capacity 1 in G. There are only two contents in the catalog: True and False. Each clause vertex C_j, $j \in [m]$, in G_ϕ corresponds to a user u_{C_j} (of weight 1) in G that requests True and False with equal probability. Thus, if a variable appears in a clause in ϕ, then the variable's corresponding cache is adjacent to the clause's corresponding user in G. Lastly, set $\ell := U$. This completes the construction of (G, ℓ). Now it only remains to verify correctness. □

Theorem 7 settles the complexity of our three problems with respect to all our considered parameters except for C and U. For these two parameters, the complexity differs depending on the problem and, as we showed in Section 3, HOMNC is XP parameterized by C or U. For HETNC-U, the three additional lower bound results we need to establish the landscape in Fig. 1 (middle) are W[1]-hardness with respect to $C + \lambda$ and $C + U$, and paraNP-hardness when parameterized by U alone. For the latter two lower bounds, we provide a simple reduction from UNARY BIN PACKING, which is not only NP-hard, but also W[1]-hard when parameterized by the number of bins (Jansen et al. 2013).

Theorem 8. HETNC-U *is* NP*-hard and also* W[1]*-hard parameterized by C, even if $U = 1$.*

For the last lower bound in the unary case, a direct adaptation of the reduction used in Theorem 8 yields that HETNC-U is W[1]-hard parameterized by C when each user requests a single content.

Corollary 9. HETNC-U *is* NP*-hard and also* W[1]*-hard parameterized by C, even if $\lambda = 1$.*

When the sizes of the contents are encoded in binary, one can establish a much stronger notion of intractability (paraNP-hardness) for HETNC when parameterized by C. Indeed, as was observed for other variants of NETWORK-CACHING (Poularakis et al. 2019), HETNC-B admits a trivial polynomial-time reduction from the weakly NP-hard 0-1 KNAPSACK problem (Garey and Johnson 1979).

Theorem 10. HETNC-B *is* NP*-hard, even if $U = C = 1$.*

A straightforward corollary of the proof of Theorem 10 yields that HETNC-B is NP-hard, even if there is only one cache and each user requests a single content.

Corollary 11. HETNC-B *is* NP*-hard, even if $C = \lambda = 1$.*

For some parameterizations, our results do not resolve whether they are FPT or W[1]-hard. In the final technical Section 6, we show that these open cases are interreducible.

5 Structural Parameters

In this section, we discuss an alternative approach towards identifying tractable fragments of HOMNC, specifically by exploiting well-established structural properties of the network. As we have seen, HETNC-B is NP-hard even if the network consists of a single edge, and HETNC-U is NP-hard even in star networks. Thus, they both remain paraNP-hard when parameterized by not only the fundamental structural parameter treewidth (Robertson and Seymour 1986), but also by essentially all other established graph parameters including, e.g., treedepth (Nesetril and de Mendez 2012), the feedback edge number (Ganian and Korchemna 2021; Bredereck et al. 2022), and the vertex cover number (Bodlaender, Groenland, and Pilipczuk 2023; Chalopin et al. 2024). Moreover, this implies that both problems remain NP-hard on planar networks—a property which is particularly relevant in the studied setting (see also Section 1). However, none of the hardness results presented thus far rule out tractability for HOMNC with respect to these structural parameters or planarity. As our next result, we show that most established structural parameters like treewidth, treedepth, and feedback edge number, as well as planarity, do not help even when dealing with the simpler homogeneous setting.

Theorem 12. HOMNC *is* NP*-hard even if $\lambda = 2$ and G is a star whose edges have each been subdivided once. Moreover, in this case it is also* W[1]*-hard parameterized by K.*

Proof Sketch. We reduce from the MAXIMUM k-VERTEX COVER problem: given a graph G and two positive integers k and t, is there a subset $V' \subseteq V(G)$ of k vertices such that at least t edges in G contain a vertex from V'? MAXIMUM k-VERTEX COVER is NP-hard and W[1]-hard parameterized by k (Cai 2008). From an instance (G, k, t) of MAXIMUM k-VERTEX COVER, we construct an instance (G', ℓ) of HOMNC as follows. For each edge $xy \in E(G)$, there is a user u_{xy} (of weight 1) in G' that only requests the contents x and y with equal probability (i.e., $p_{u_{xy}x} = p_{u_{xy}y} = 0.5$ and no other contents are requested), and a cache c_{xy} of capacity 1 that is adjacent only to u_{xy}. There is also one cache c of capacity k that is adjacent to every user in G'. Lastly, set $\ell := (U + t)/2$. This completes the construction of (G', ℓ). At this point, it remains to verify correctness. □

The previous theorem rules out tractability even on planar networks, but does not do so when Δ and K are constants. In our next result, we rule out tractability for HOMNC, even when restricted to planar networks where Δ, λ, and K are all fixed constants. We establish this via a reduction from the NP-hard PLANAR 3-SAT-E3 problem (Middendorf and Pfeiffer 1993), in which each variable appears in 3 clauses.

Theorem 13. HOMNC *is* NP*-hard, even if G is planar, $K = 1$, $\lambda = 3$, and $\Delta = 5$.*

Proof Sketch. From an instance ϕ of PLANAR 3-SAT-E3, we construct an instance (G, ℓ) of HOMNC as follows. Let $x_1, \ldots, x_n$ and $C_1, \ldots, C_m$ be the variables and clauses in ϕ, respectively. For each of the literals $x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n$ in ϕ, there is a content in the catalog with the same name. We construct G from the planar incidence graph G_ϕ of ϕ as follows. Each variable vertex x_i, $i \in [n]$, in G_ϕ corresponds to a cache c_{x_i} of capacity 1 in G. Each clause vertex C_j, $j \in [m]$, in G_ϕ corresponds to a user u_{C_j} (of weight 1) in G that only requests (with non-zero probability) the contents corresponding to the literals that the clause C_j contains, and u_{C_j} requests these contents with equal probability. Thus, if a variable appears in a clause in ϕ, then the variable's corresponding cache is adjacent to the clause's corresponding user in G. Further, for each cache c_{x_i}, $i \in [n]$, there is an additional cache c'_{x_i} and two additional users u_{x_i} and $u_{\overline{x}_i}$. The user u_{x_i} ($u_{\overline{x}_i}$, resp.) only requests the content x_i ($\overline{x}_i$, resp.) with non-zero probability, i.e., $p_{u_{x_i}x_i} = 1$ ($p_{u_{\overline{x}_i}\overline{x}_i} = 1$, resp.). Both c_{x_i} and c'_{x_i} are adjacent to both u_{x_i} and $u_{\overline{x}_i}$. Since these are all the adjacencies of the cache c'_{x_i} and the users u_{x_i} and $u_{\overline{x}_i}$, their respective vertices can be placed arbitrarily close to c_{x_i}, thus maintaining that G is planar. For each user u_{C_j}, $j \in [m]$, if C_j contains 3 literals in ϕ, then there are two additional caches $c_{u_{C_j}}$ and $c'_{u_{C_j}}$ of capacity 1 that are adjacent only to u_{C_j}; $c'_{u_{C_j}}$ does not exist if C_j contains only 2 literals. Similarly, these are all the adjacencies of

these additional caches, and so, their respective vertices can be placed arbitrarily close to u_{C_j}, thus maintaining that G is planar. Lastly, set $\ell := U$. This completes the construction of (G, ℓ), and now it only remains to verify correctness. □

Neither of the previous results rules out tractability for HOMNC when parameterized by the vertex cover number of the graph, which is the minimum size of a subset S of the vertices such that every edge is incident to at least one vertex in S. In the next section dedicated to unifying the remaining open cases, we show that, for HOMNC, this parameterization is complexity-theoretically equivalent to several other ones we already considered. As a by-product, this yields XP-tractability for HOMNC parameterized by the vertex cover number, contrasting the lower bounds ruling out the use of essentially all other structural graph parameters.

6 Interreducibility: Linking the Open Cases

For several of the studied cases, we obtained XP algorithms, but lack the corresponding W[1]-hardness proofs which would rule out inclusion in FPT. In this section, for all parameterizations of HOMNC which we show to admit XP algorithms in Section 3 (in particular $C, U, C+U, C+\lambda$, and $U+K$) and also for the vertex cover number parameter mentioned at the end of the previous section, we prove the following: either HOMNC is W[1]-hard for each of these, or it is FPT for each of these. That is, all of the arising parameterized problems are equivalent and there is only a single open case left for HOMNC. We emphasize that these results are not trivial, as here it does not hold that bounding one parameter is immediately equivalent to bounding the other.

Theorem 14. *Let $\varkappa$ and $\varkappa'$ denote any two of the following six parameters: C, U, $U+K$, $C+U$, $C+\lambda$, and the vertex cover number $\mathtt{vc}(G)$. Then, there exists a parameterized reduction from* HOMNC *parameterized by $\varkappa$ to* HOMNC *parameterized by $\varkappa'$.*

Proof Sketch. First, note that we do not need to present parameterized reductions in the cases where $\varkappa \geq \varkappa'$ as the trivial reduction where the two instances are identical is valid in these cases. This implies that we do not need to consider the case where $\varkappa = C + U$ and $\varkappa' = \mathtt{vc}(G)$. We now provide a case analysis that can be shown to be exhaustive due to transitivity of reductions. Since each of the following reductions takes polynomial time and the equivalence of the two instances is immediate, we simply present the reductions.

Case 1: $\varkappa = U$ and $\varkappa' = C$. Covered in the proof of Obs. 5.

Case 2: $\varkappa = C$ and $\varkappa' = U$. From an instance (G, ℓ) of HOMNC with C caches, we obtain an equivalent instance (G', ℓ) of HOMNC with $U' \leq 2^C$ users as follows. For each set $U^* \subseteq \mathcal{U}$ of 2 or more users with the same neighborhood in G, delete all the users except for one, denote this user by u^*, and set the weight and content request probabilities of u^* such that any potential cache hits for U^* in G would result in the same amount of cache hits for u^* in G'. Specifically, $w(u^*) := \sum_{u \in U^*} w(u)$ and, for each $s \in \mathcal{S}$, $p_{u^* s} := \left(\sum_{u \in U^*} w(u) p_{us}\right) / w(u^*)$. As there are 2^C different subsets of caches in G, there are at most 2^C users in G'.

Case 3: $\varkappa = U$ and $\varkappa' = U + K$. From an instance (G, ℓ) of HOMNC with U users, we obtain an equivalent instance (G', ℓ) of HOMNC with a maximum cache capacity of $K' = 1$ and U users as follows. Let C be the number of caches in (G, ℓ). For each $i \in [C]$, replace the cache c_i in G by $\kappa(c_i)$ caches of capacity 1 (recall that $\kappa(c_i) \leq S$) with the same neighborhood as c_i.

Case 4: $\varkappa = C$ and $\varkappa' = C + \lambda$. From an instance (G, ℓ) of HOMNC with C caches, we obtain an equivalent instance (G', ℓ) of HOMNC with C caches in which each user requests only one content as follows. For each user u in G, delete u, and, for each content s requested with non-zero probability by u, add a user u_s with the same neighborhood as u and weight $w(u) \cdot p_{us}$ such that $p_{u_s s} = 1$.

To finish the proof, it suffices to expand on the ideas used above to establish **Case 5:** $\varkappa = \mathtt{vc}(G)$ and $\varkappa' = C + U$. □

We conjecture that HOMNC is W[1]-hard under these parameterizations, but believe that a proof requires novel techniques or insights into the problem. Moreover, as HETNC-U generalizes HOMNC, resolving this conjecture in the affirmative would also resolve the sole open case for HETNC-U.

7 Generalizations, Impact, and Conclusion

The FPT algorithms developed in Theorems 1 and 3 can be easily adapted to handle a more general framework associated with NETWORK-CACHING since they consider all the relevant feasible caching allocations. Indeed, for any objective function that can be computed in the desired FPT time when given a caching allocation, these algorithms can also compute the optimal value of that objective function along with its associated caching allocation. Moreover, most objective functions in the literature satisfy the above condition. For example, these algorithms can be trivially modified to deal with variants of NETWORK-CACHING where weights are added to the edges of the bipartite graph which represent the caching gain obtained from retrieving a content requested by the user from a specific cache (Ioannidis and Yeh 2016; Tsigkari and Spyropoulos 2022), and/or the objective function concerns other metrics such as QoS, streaming rate or energy consumption (Paschos, Iosifidis, and Caire 2020). Our hardness results also carry over to these variants as well as generalizations combining caching with other network-related decisions (Dehghan et al. 2016; Krolikowski, Giovanidis, and Di Renzo 2018; Ricardo et al. 2021). On the other hand, these hardness results cannot be lifted to non-discrete variants of NETWORK-CACHING, as these can typically be solved in polynomial time (Shanmugam et al. 2013).

All of our algorithms are deterministic and implementable in CDNs or inference delivery networks, and in fact it is reasonable to expect some of the studied parameters to achieve small values in practice (see also Section 2). However, we believe that a natural next step would be to find the theoretically fastest algorithms under the (Strong) Exponential Time Hypothesis. Further, designing informed heuristics based on our complexity analysis—as was successfully done in other fields (Bäckström et al. 2012; Rost, Döhne, and Schmid 2019; Komusiewicz, Schramek, and Sommer 2023)—would be an interesting alternate direction one could take.

Acknowledgements

This work was funded by the Austrian Science Fund (FWF) [10.55776/Y1329 and 10.55776/COE12], the WWTF Vienna Science and Technology Fund (Project 10.47379/ICT22029), the Spanish Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU through the project 6G-RIEMANN (TSI-063000-2021-147), the EU Horizon Europe TaRDIS project (grant agreement 101093006), and the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe and innovation programme under Grant Agreement No. 101139067 (ELASTIC). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union (EU). Neither the EU nor the granting authority can be held responsible for them.

References

Bäckström, C.; Chen, Y.; Jonsson, P.; Ordyniak, S.; and Szeider, S. 2012. The complexity of planning revisited—a parameterized analysis. In *Proc. of AAAI 2012*, volume 26, 1735–1741.

Bastug, E.; Bennis, M.; and Debbah, M. 2014. Living on the edge: The role of proactive caching in 5G wireless networks. *IEEE Communications Magazine*, 52(8): 82–89.

Bhattacharjee, R.; Banerjee, S.; and Sinha, A. 2020. Fundamental limits on the regret of online network-caching. *Proc. of the ACM on Measurement and Analysis of Computing Systems*, 4(2): 1–31.

Blaszczyszyn, B.; and Giovanidis, A. 2015. Optimal geographic caching in cellular networks. In *Proc. of IEEE ICC 2015*, 3358–3363.

Bliem, B.; Bredereck, R.; and Niedermeier, R. 2016. Complexity of efficient and envy-free resource allocation: few agents, resources, or utility levels. In *Proc. of IJCAI 2016*, 102–108.

Bodlaender, H. L.; Groenland, C.; and Pilipczuk, M. 2023. Parameterized Complexity of Binary CSP: Vertex Cover, Treedepth, and Related Parameters. In *Proc. of ICALP 2023*, volume 261 of *LIPIcs*, 27:1–27:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Bredereck, R.; Heeger, K.; Knop, D.; and Niedermeier, R. 2022. Parameterized complexity of stable roommates with ties and incomplete lists through the lens of graph parameters. *Inf. Comput.*, 289: 104943.

Cai, L. 2008. Parameterized Complexity of Cardinality Constrained Optimization Problems. *The Computer Journal*, 51(1): 102–121.

Chalopin, J.; Chepoi, V.; Mc Inerney, F.; and Ratel, S. 2024. Non-Clashing Teaching Maps for Balls in Graphs. In *Proc. of COLT 2024*, volume 247 of *PMLR*, 840–875.

Cygan, M.; Fomin, F. V.; Kowalik, Ł.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Dehghan, M.; Jiang, B.; Seetharam, A.; He, T.; Salonidis, T.; Kurose, J.; Towsley, D.; and Sitaraman, R. 2016. On the complexity of optimal request routing and content caching in heterogeneous cache networks. *IEEE/ACM Trans. on Networking*, 25(3): 1635–1648.

Deligkas, A.; Eiben, E.; Ganian, R.; Hamm, T.; and Ordyniak, S. 2021. The Parameterized Complexity of Connected Fair Division. In *Proc. of IJCAI 2021*, 139–145.

Diestel, R. 2012. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer.

Disney+. 2023. Disney+ pricing. https://help.disneyplus.com/article/disneyplus-price. Accessed: 2023-12-01.

Downey, R. G.; and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer.

Eiben, E.; Ganian, R.; Hamm, T.; and Ordyniak, S. 2023. Parameterized complexity of envy-free resource allocation in social networks. *Artificial Intelligence*, 315: 103826.

Eiben, E.; Ganian, R.; Kanj, I.; Ordyniak, S.; and Szeider, S. 2021. The Parameterized Complexity of Clustering Incomplete Data. In *Proc. of AAAI 2021*, volume 35, 7296–7304.

Ganian, R.; Kanj, I.; Ordyniak, S.; and Szeider, S. 2020. On the Parameterized Complexity of Clustering Incomplete Data into Subspaces of Small Rank. In *Proc. of AAAI 2020*, volume 34, 3906–3913.

Ganian, R.; Kanj, I. A.; Ordyniak, S.; and Szeider, S. 2018. Parameterized Algorithms for the Matrix Completion Problem. In *Proc. of ICML 2018*, volume 80 of *PMLR*, 1642–1651.

Ganian, R.; and Korchemna, V. 2021. The Complexity of Bayesian Network Learning: Revisiting the Superstructure. In *Proc. of NeurIPS 2021*, volume 34, 430–442.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Grüttemeier, N.; and Komusiewicz, C. 2022. Learning Bayesian networks under sparsity constraints: A parameterized complexity analysis. *Journal of Artificial Intelligence Research*, 74: 1225–1267.

Ioannidis, S.; and Yeh, E. 2016. Adaptive caching networks with optimality guarantees. *ACM SIGMETRICS Performance Evaluation Review*, 44(1): 113–124.

Jansen, K.; Kratsch, S.; Marx, D.; and Schlotter, I. 2013. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1): 39–49.

Komusiewicz, C.; Schramek, J.; and Sommer, F. 2023. On the Group Coverage Centrality Problem: Parameterized Complexity and Heuristics. In *Proc. of ACDA 2023*, 13–24.

Kratochvíl, J.; and Tuza, Z. 2002. On the complexity of bicoloring clique hypergraphs of graphs. *J. Algorithms*, 45: 40–54.

Krolikowski, J.; Giovanidis, A.; and Di Renzo, M. 2018. A decomposition framework for optimal edge-cache leasing. *IEEE Journal on Selected Areas in Communications*, 36(6): 1345–1359.

Lindgren, E.; Reddi, S.; Guo, R.; and Kumar, S. 2021. Efficient training of retrieval models using negative cache. In *Proc. of NeurIPS 2021*, volume 34, 4134–4146.

Liu, D.; Chen, B.; Yang, C.; and Molisch, A. F. 2016. Caching at the wireless edge: design aspects, challenges, and future directions. *IEEE Communications Magazine*, 54(9): 22–28.

Maggi, L.; Gkatzikis, L.; Paschos, G.; and Leguay, J. 2018. Adapting caching to audience retention rate. *Computer Communications*, 116: 159–171.

Mhaisen, N.; Sinha, A.; Paschos, G.; and Iosifidis, G. 2022. Optimistic no-regret algorithms for discrete caching. *Proc. of the ACM on Measurement and Analysis of Computing Systems*, 6(3): 1–28.

Middendorf, M.; and Pfeiffer, F. 1993. On the complexity of the disjoint paths problem. *Combinatorica*, 13(1): 97–107.

Nesetril, J.; and de Mendez, P. O. 2012. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer.

Netflix. 2023. Plans and Pricing. https://help.netflix.com/en/node/24926. Accessed 2023-12-01.

Netflix. 2024a. Open Connect appliances. https://openconnect.netflix.com/appliances. Accessed 2024-01-05.

Netflix. 2024b. Open Connect fill patterns. https://openconnect.zendesk.com/hc/en-us/articles/360035618071-Fill-patterns. Accessed 2024-01-05.

Netflix Tech Blog. 2016. Netflix and Fill. https://netflixtechblog.com/netflix-and-fill-c43a32b490c0. Accessed 2023-12-20.

Ordyniak, S.; and Szeider, S. 2013. Parameterized Complexity Results for Exact Bayesian Network Structure Learning. *Journal of Artificial Intelligence Research*, 46: 263–302.

Paria, D.; and Sinha, A. 2021. LeadCache: Regret-Optimal Caching in Networks. In *Proc. of NeurIPS 2021*, volume 34, 4435–4447.

Paschos, G.; Iosifidis, G.; and Caire, G. 2020. Cache optimization models and algorithms. *Foundations and Trends® in Communications and Information Theory*, 16(3–4): 156–345.

Paschos, G. S.; Bastug, E.; Land, I.; Caire, G.; and Debbah, M. 2016. Wireless Caching: Technical Misconceptions and Business Barriers. *IEEE Communications Magazine*, 54(8): 16–22.

Paschos, G. S.; Destounis, A.; Vigneri, L.; and Iosifidis, G. 2019. Learning to cache with no regrets. In *Proc. of IEEE INFOCOM 2019*, 235–243.

Paschos, G. S.; Iosifidis, G.; Tao, M.; Towsley, D.; and Caire, G. 2018. The role of caching in future communication systems and networks. *IEEE Journal on Selected Areas in Communications*, 36(6): 1111–1125.

Poularakis, K.; Iosifidis, G.; Argyriou, A.; Koutsopoulos, I.; and Tassiulas, L. 2019. Distributed Caching Algorithms in the Realm of Layered Video Streaming. *IEEE Trans. on Mobile Computing*, 18(4): 757–770.

Poularakis, K.; Iosifidis, G.; Argyriou, A.; and Tassiulas, L. 2014. Video delivery over heterogeneous cellular networks: Optimizing cost and performance. In *Proc. of IEEE INFOCOM 2014*, 1078–1086.

Poularakis, K.; Iosifidis, G.; and Tassiulas, L. 2014. Approximation algorithms for mobile data caching in small cell networks. *IEEE Trans. on Communications*, 62(10): 3665–3677.

Ricardo, G. I.; Tuholukova, A.; Neglia, G.; and Spyropoulos, T. 2021. Caching policies for delay minimization in small cell networks with coordinated multi-point joint transmissions. *IEEE/ACM Trans. on Networking*, 29(3): 1105–1115.

Robertson, N.; and Seymour, P. D. 1986. Graph Minors. II. Algorithmic Aspects of Tree-Width. *J. Algorithms*, 7(3): 309–322.

Rohatgi, D. 2020. Near-optimal bounds for online caching with machine learned advice. In *Proc. of SODA 2020*, 1834–1845.

Rost, M.; Döhne, E.; and Schmid, S. 2019. Parametrized complexity of virtual network embeddings: Dynamic & linear programming approximations. *ACM SIGCOMM Computer Communication Review*, 49(1): 3–10.

Salem, T. S.; Castellano, G.; Neglia, G.; Pianese, F.; and Araldo, A. 2023. Toward Inference Delivery Networks: Distributing Machine Learning With Optimality Guarantees. *IEEE/ACM Trans. on Networking*, 32(1): 859–873.

Shanmugam, K.; Golrezaei, N.; Dimakis, A. G.; Molisch, A. F.; and Caire, G. 2013. FemtoCaching: Wireless Content Delivery Through Distributed Caching Helpers. *IEEE Trans. on Information Theory*, 59(12): 8402–8413.

Tadrous, J.; and Eryilmaz, A. 2015. On optimal proactive caching for mobile networks with demand uncertainties. *IEEE/ACM Trans. on Networking*, 24(5): 2715–2727.

Thapa, C.; Arachchige, P. C. M.; Camtepe, S.; and Sun, L. 2022. Splitfed: When federated learning meets split learning. In *Proc. of AAAI 2022*, volume 36, 8485–8493.

Tirana, J.; Tsigkari, D.; Iosifidis, G.; and Chatzopoulos, D. 2024. Workflow Optimization for Parallel Split Learning. In *Proc. of IEEE INFOCOM 2024*.

Tsigkari, D.; and Spyropoulos, T. 2022. An approximation algorithm for joint caching and recommendations in cache networks. *IEEE Trans. on Network and Service Management*, 19(2): 1826–1841.

Yu, Z.; Li, H.; Fu, F.; Miao, X.; and Cui, B. 2024. Accelerating Text-to-Image Editing via Cache-Enabled Sparse Diffusion Inference. In *Proc. of AAAI 2024*, volume 38, 16605–16613.

Zhang, Z.; Shao, W.; Ge, Y.; Wang, X.; Gu, J.; and Luo, P. 2024. Cached Transformers: Improving Transformers with Differentiable Memory Cachde. In *Proc. of AAAI 2024*, volume 38, 16935–16943.

Zhou, R.; Khemmarat, S.; and Gao, L. 2010. The impact of YouTube recommendation system on video views. In *Proc. of IMC 2010*, 404–410.

Zhu, B.; Sheng, Y.; Zheng, L.; Barrett, C.; Jordan, M.; and Jiao, J. 2023. Towards Optimal Caching and Model Selection for Large Model Inference. In *Proc. of NeurIPS 2023*, volume 36, 59062–59094.