Cache Networks of Counting Queues

Yuanyuan Li, *Member, IEEE,* and Stratis Ioannidis, *Member, IEEE*

***Abstract*—We consider a cache network in which intermediate nodes equipped with caches can serve content requests. We model this network as a universally stable queuing system, in which packets carrying identical responses are consolidated before being forwarded downstream. We refer to resulting queues as `M/M/1C` or *counting queues*, as consolidated packets carry a counter indicating the packet's multiplicity. Cache networks comprising such queues are hard to analyze; we propose two approximations: one via `M/M/∞` queues, and one based on `M/M/1C` queues under the assumption of Poisson arrivals. We show that, in both cases, the problem of jointly determining (a) content placements and (b) service rates admits a poly-time, $1-1/e$ approximation algorithm. We also show that our analysis, with respect to both algorithms and associated guarantees, extends to (a) counting queues over items, rather than responses, as well as to (b) queuing at nodes and edges, as opposed to just edges. Numerical evaluations indicate that our proposed approximation algorithms yield good solutions in practice, significantly outperforming competitors.**

***Index Terms*—DR-submodularity, cache networks, Jackson networks**

I. Introduction

We consider a network of caches, in which intermediate nodes store requested contents and can serve content requests. Cache networks are a natural abstraction for many applications, including information-centric networks [2]–[5], content delivery networks [6]–[8], and peer-to-peer networks [9], [10]. A series of recent efforts focus on the problem of *cache network design*, describing algorithms for placing contents in caches in order to minimize routing costs [11]–[15].

In most prior work, routing costs are modeled via a linear function of traffic in each edge, which does not capture, e.g., delays due to queuing. A recent paper by Mahdian et al. [15] addresses this limitation by considering so-called *Kelly cache networks*, i.e., cache networks whose links are associated with `M/M/1` queues. This formulation has several advantages. First, it allows the authors to capture queuing costs in their cache network design. Second, the system is a Kelly network [16] and, hence, its steady-state distribution is easy to characterize. Unfortunately, these modelling advantages come at the expense of realism. In `M/M/1` queues, packets carrying *the same content* are queued and served separately. This would not happen in a real network, in which packets carrying identical content would be used to serve multiple requests. As a side-effect of this modeling distortion, networks studied by Mahdian et al. can become unstable: queue sizes can grow to infinity, congested with packets containing identical content.

This is an extended version of a paper that appeared in the IEEE International Conference on Computer Communications (INFOCOM 2020) [1]. The authors are with the Electrical and Computer Engineering Department, Northeastern University, Boston, MA 02115 USA (e-mail: yuanyuanli@ece.neu.edu; ioannidis@ece.neu.edu).

In this work, we address this problem by considering a different type of queue, originally introduced by Abolhassani et al. [17], which we refer to as a *counting queue*. When packets containing identical content arrive in such a queue, they merge, resulting in a single packet carrying the same content. The header of this packet contains a counter with the "cardinality" of merged packets it represents. Merged packets are forwarded towards the request source, serving multiple requests via a single response. Counting queues, which we denote by `M/M/1C`, capture real-life behavior more accurately than `M/M/1` queues. They also lead to networks that are *universally stable*: the merging of packets prevents queues (and counters) to grow to infinity, irrespective of demand [17].

Nevertheless, by considering networks of `M/M/1C` queues, we suffer a reversal of fortune in comparison to Mahdian et al. [15]: though we gain realism, we lose tractability, as the resulting system is *not* a Kelly network; thus, steady-state distributions do not have product form and are hard to describe. As a result, steady-state behavior and the routing cost optimization that it corresponds to are difficult to characterize in a closed form. One of the main contributions of our work is to address this challenge directly, providing both analytical and experimental evidence that `M/M/1C` queues are well-approximated by `M/M/∞` queues; the latter are indeed easy to analyze, enabling us to produce algorithms for the cache design problem with approximation guarantees. In particular, we make the following contributions:

- We introduce the `M/M/1C` queues by Abolhassani et al. [17] into the cache network setting by Mahdian et al. [15], aiming to capture network behavior with greater realism. In contrast to `M/M/1` queues, resulting networks are *not* Kelly networks, and intermediate queue arrivals are *not* Poisson.
- We show that `M/M/∞` queues approximate `M/M/1C` queues; we show this both experimentally and analytically, through a mutual stochastic dominance (c.f. Thm. 1).
- Motivated by the above observations, we study two cache network design problems, each serving as an approximation of a cache network with counting queues. Both problems optimize content placement and service assignment decisions *jointly*. In the first problem, $\textsc{MinCost}_{\texttt{M/M/}\infty}$, we approximate counting queues with `M/M/∞` queues; in the second problem, $\textsc{MinCost}_{\texttt{M/M/1C}}$, we use `M/M/1C` steady-state distributions, assuming however Poisson arrivals in intermediate queues.
- We show that both problems are NP-hard (c.f. Thm. 2), and construct a $1-1/e$ poly-time approximation algorithm for the joint optimization of item placements and service assignments (c.f. Thm. 4).

- Finally, we conduct extensive experiments over multiple topologies: our joint item placements and service rate assignments significantly outperform competitors.

From a technical standpoint, our algorithm solves a mixed integer problem with a non-convex objective; this requires showing that both $\text{MINCOST}_{\text{M/M/}\infty}$ and $\text{MINCOST}_{\text{M/M/1C}}$ exhibit an important underlying structural property (c.f. Theorem 3): their objectives are continuous *Diminishing Returns (DR) submodular* [18] w.r.t. both content placements *and* service assignments *jointly*, a result that is non-obvious.

The remainder of this paper is structured as follows. We review related work in Sec. II. We introduce M/M/1C and M/M/∞ queues, and formulate our two approximate problems in Sec. III. Sec. IV contains our approximation algorithms. We extend our analysis, algorithms and guarantees in Sec. V. Our experiments are in Sec. VI, and we conclude in Sec. VII.

II. Related Work

Cache Network Topology. Cache networks have been intensely studied both experimentally and theoretically. Several works [19]–[28] model the network as a bipartite graph, in which requests fetch contents in one hop, and proposed algorithms do not readily generalize to arbitrary topologies. Multi-hop networks are studied by a series of recent papers [11]–[14], [29]–[31], which are generally harder to analyze. We elaborate on these below.

Optimization Objectives. Several works assign a cost to each edge in the network, and aim at making caching decisions that minimize expected routing costs. This objective has been studied in the context of femtocaching systems [11], arbitrary cache networks [12], [14], small cell networks [13], parallel computing frameworks [31] and in proactive (i.e. predictive) cache networks [25], to name a few. By assuming a fixed cost per edge, all above works assume costs are linear functions of traffic. As such, they cannot be used to model costs in queuing systems like the ones we study. Content placements that maximize the number of requests served by caches are studied in hierarchical caching networks [32], in cellular netorks with moving users [26], in arbitrary congestible networks [30], and in multi-cell mobile edge computing networks with storage, computation, and communication constraints [28]. To minimize the expected delay experienced by all the requester, Domingues et al. [33] study the interplay between content search and content placement and Poularakis et al. [27] study the content placement of layered-video. Yeh et al. [5] focus on maximizing throughput, i.e., user demand rate satisfied by the network. Our model, objective, and constraints, significantly depart from the ones considered in the above works.

Our work is closest to, and inspired by, recent work Mahdian et al. [15]. As discussed in the introduction, they consider a cache network in which each edge is associated with an M/M/1 queue. Resulting costs are not linear, capture queuing, and the objective is submodular and therefore optimizable via the continuous greedy algorithm of Calinescu et al. [34]. We depart by considering M/M/1C (counting) queues and M/M/∞ approximations thereof, and optimizing item placement and routing decisions *jointly*: as a result, our optimization requires tools beyond classic submodularity.

M/M/1C Queues. Abolhassani et al. [17] introduce M/M/1C queues and prove their universal stability, in a simpler setting than the one we consider here. In particular, the authors study a multicast single-cell wireless network in which users issue requests for a finite set of items. These requests are queued while waiting for an item, forming M/M/1C queues: whenever the base station broadcasts a requested data item, all requests in a queue are served simultaneously, just like in our setting. The authors prove that this system is always stable for all loading factors (given by the ratio of the request rate to the service rate). They also study work-conserving multicasting policies, particularly characterizing the scaling delay gains of the First-Come-First-Serve policy. Though this setting is similar to ours, we depart in several significant aspects. We consider a *network* of queues, whereas they model a single base station cell as a set of parallel M/M/1C queues: this corresponds to just a single "edge" in our setting. Linking queues/edges together to form a network gives rise to significant challenges in our setting, that Abolhassani et al. do not face. In particular, since the departure process is not Poisson, our linked queue network is not a Kelly network. It is therefore hard to formulate its steady state distribution in a closed form, which is not the case for the single-edge setting they consider. Second, we have a different objective: they minimize the aggregate average number of requests in the system; in contrast, we minimize not just the number of responses, but more generally the total queuing cost, which is a function of the number of responses. Finally, the NP-hardness of our problem is due to the networked setting, and the combinatorial challenge we need to address does not appear in the single-edge setting by Abolhassani et al.

Joint Optimization. Dehgan et al. [24], Poularakis et al. [28], Ioannidis and Yeh [14] and Liu et al. [30] consider the joint optimization of caching and routing in networks; the first two in particular study routing in the bipartite setting, while the last two do so in arbitrary topologies. Caching and routing decisions are formulated as binary variables in those works. Our joint optimization of caching and service rates is fundamentally different, not only because it contains both continuous and integer variables; it is also not amenable to standard submodularity approaches, as is [14], but requires the use of continuous DR-submodular optimization [18] instead to attain a $1 - 1/e$ approximation guarantee. Zafari et al. [35] jointly optimize data compression rate and data placement in a tree topology, posing this as a mixed integer problem; they solve this by a spatial branch-and-bound search strategy, which comes with no poly-time approximation guarantees.

TTL Caches. Time-to-Live (TTL) caches have drawn attention recently due to their connection to classic replacement policies. TTL caches assign a timer to each content, and an eviction occurs when a timer expires. Che et al. [36] show that the hit probability of LRU caches can be approximated through a TTL-based eviction scheme by introducing the notion of characteristic time. This approach has been refined and extended to other traditional replacement policies [37], [38] as well as more general requests arrival processes [39], thus providing a general framework for analyzing different replacement policies. Several papers study the approximate

and exact behavior of, e.g., hit probabilities and cache occupancies, in both individual TTL-caches as well as networks thereof [40], [41]. TTL-like mechanisms, referred to as reinforced counters, are used for content placement. Domingues et al. [33], [42] propose analytical models to study how to turn reinforced counters, while routing is based on random walks. We depart in considering queuing-based, network-wide metrics, rather than hit rates, but also by studying off-line, centralized algorithms.

Approximation Networks. Jackson networks [43] are classic queuing networks comprising reversible queues serving packets of a single type. Many classic queues, including M/M/1 and M/M/∞, are reversible. Kelly networks [16] generalize Jackson networks to queues serving packets of multiple types. Both leverage reversibility to show that steady state distributions have a product form. This property makes the characterization of expected steady state rewards and costs tractable. Product-form steady-state distributions arise also in settings where service times are not exponentially distributed. For example, symmetric queues (e.g., M/D/1 and M/E_k/1) exhibit his property [16].

In general, if properties like reversibility or symmetry are not satisfied, the steady state distribution of the network does not have a closed form. Approximations similar to the ones we employ are used often in this case; the most famous example are so-called *loss networks*. Dating back to Erlang's work in 1925, loss networks model the probability of serving a telephone call in a circuit-switched network. More broadly, they are used to model stochastic resource allocation problems [44], and were famously analysed by Frank Kelly in his seminal 1991 paper [45]. The so-called fixed point approximation (see p. 337 in [45]) is an approximation, operating under an assumption that is very similar to the one we describe here. Namely, the approximation assumes that losses/call blocks at a link happen independently, effectively implying that traffic exiting a loss link is Poisson, even though it is not, and call blocks are not independent. This approximation serves as the basis for a significant body of work on analyzing loss networks for a variety of applications [46]–[49], despite the fact that it is indeed an approximation.

Submodular Maximization. Maximizing a submodular function subject to a matroid constraint is classic. Krause and Golovin [50] show that the greedy algorithm achieves a $1/2$ approximation ratio. Calinescu et al. [34] propose a *continuous greedy* algorithm improving the ratio to $1 - 1/e$, that applies a Frank-Wolfe [51] variant to the multilinear extension of the submodular objective. With the help of auxiliary potential functions, Filmus and Ward [52] run a non-oblivious local search after the greedy algorithm, and also produce a $1 - 1/e$ approximation ratio. Further improvements are made by Sviridenko et al. [53] for a more restricted class of submodular functions with bounded curvature. Bian et al. [54] [18] show that the same Frank-Wolfe variant can be used to maximize continuous DR-submodular functions within a $1 - 1/e$ ratio. One of our technical contributions is to show that the multilinear extension, in our case, which is a function of both randomized item placements *and* continuous service rates, is jointly DR-submodular in its input. We note that we depart from multilinear extensions considered in prior work [15], [34], [53], [55], that did not contain continuous variables beyond the ones due to randomization.

III. Problem Formulation

We consider a network of caches which store a finite number of items. Requests for items are generated and are routed through pre-determined paths. Upon hitting a cache which stores the requested item, a response carrying the item is back-propagated over the reverse path. This generates traffic over queues on network edges. We aim to minimize queuing costs by (a) placing items in caches and (b) assigning queue service rates across responses appropriately. In what follows, we describe this problem in detail.

A. System Model

Following Mahdian et al. [15], we consider a network modeled as a directed graph $G(V, E)$ with node set V. Each edge e in the graph is represented by $e = (u, v) \in E$, where $u, v \in V$. This directed graph is symmetric, i.e., if $(u, v) \in E$, then $(v, u) \in E$ as well.

1) Caches: Items of equal size are permanently stored in certain network nodes, called *designated servers*. Formally, for every item $i \in C$, where set C is the *item catalog*, we denote by $\mathcal{S}_i \subseteq V$ the set of designated servers storing i. Every node in V, including designated servers, has additional storage that is used to store more items from the catalog. Formally, each node v is associated with a cache of finite storage capacity $c_v \in \mathbb{N}$. We use a binary variable $x_{vi} \in \{0, 1\}$ indicating whether node $v \in V$ is caching item $i \in C$. Let vector $\boldsymbol{x} = [x_{vi}]_{v \in V, i \in C} \in \{0, 1\}^{|V||C|}$ be the global item placement vector. We denote the set of feasible placements by:

$$\mathcal{D} = \{\boldsymbol{x} \in \{0, 1\}^{|V||C|} : \sum_{i \in C} x_{vi} \leq c_v, \forall v \in V\}. \quad (1)$$

2) Requests and Responses: A set of nodes $Q \subseteq V$, called *query nodes*, generate requests to fetch items from C. Let $\mathcal{R}$ be the set of response types. Each request has a unique type $r = (i^r, p^r) \in \mathcal{R}$, determined by (a) the item $i^r \in C$ being requested , and (b) the *path* $p^r \subseteq V$ followed by the request. We make the following assumptions on path $p^r, r \in \mathcal{R}$: (a) p^r is a sequence of adjacent nodes, e.g. p^r_1, p^r_2, ..., p^r_K, where $(p^r_i, p^r_{i+1}) \in E$, (b) $p^r_1 \in Q$, i.e., the first node of path is a query node, (c) $p^r_K \in \mathcal{S}_{i^r}$, i.e., the last node of path is a designated server, and (d) the path p^r is simple, i.e., it does not contain repeated nodes. For $v \in p^r$, let $k_{p^r}(v) \in \{1, 2, ..., K\}$ be the position of node v in path p^r, i.e., $k_{p^r}(v) = k$ iff $p^r_k = v$.

Since assuming all items have the same size, we should view an item as a piece/chunk of a file. Moreover, a more realistic scenario would involve multiple items being requested simultaneously: this would indicate that all pieces of file are requested at once. Our model can extend to address this; we discuss this further in Sec. V-A. However, for the sake of notational simplicity, we first focus on single item requests, as in, e.g., [12], [15].

Requests of type r are generated according to an exogenous Poisson process with rate $\lambda^r \geq 0, r \in \mathcal{R}$. Then, they follow

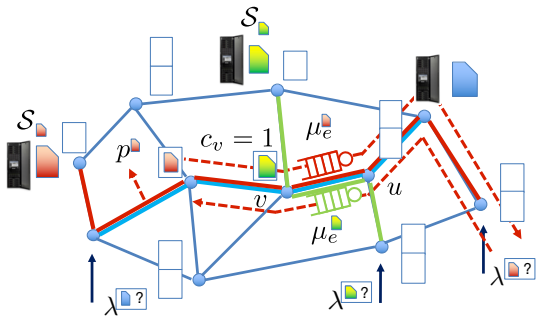


Fig. 1: An example of a network of counting queues. Three request types r, each for an item $i^r \in \{\texttt{red}, \texttt{green}, \texttt{blue}\}$ are generated with rates λ^r, following distinct paths p^r to designated servers $\mathcal{S}_{i^r}$. Node v, with capacity $c_v = 1$, is shown in the center of the graph; all three paths pass through v, and the green and red response flows both pass through edge $e = (v, u)$. Hence, the service rate μ_e is split across the green and red response types.

path p^r; when the request reaches a node storing item i^r, a response is generated. This response carries item i^r to query node $p_1^r \in \mathcal{Q}$ following the reverse path. Given all the paths $\{p^r, r \in \mathcal{R}\}$, for every edge $e \in E$, we denote by $\mathcal{R}_e$ the set of response types passing through edge e, i.e., for $e = (v, u)$,

$$\mathcal{R}_{(v,u)} = \{r \in \mathcal{R} : (u, v) \in p^r\}.$$

3) Queuing Costs: We assume requests are negligible, but responses incur traffic in the network. We model this traffic as follows. Every edge $e \in E$ is associated with service rate $\mu_e \in \mathbb{R}_+$. The service rate in an edge is split across response types. For every type $r \in \mathcal{R}_e$, there exists a queue with service rate μ_e^r. Assume that the minimum service rate for all queues is some small $\epsilon \in \mathbb{R}_+$. Let vector $\boldsymbol{\mu} = [\mu_e^r]_{e \in E, r \in \mathcal{R}_e} \in \mathbb{R}_+^{\sum_e |\mathcal{R}_e|}$ be the global service rate assignment vector. We denote the set of feasible assignments by:

$$\mathcal{D}_\mu = \{\boldsymbol{\mu} \in \mathbb{R}_+^{\sum_e |\mathcal{R}_e|} : \mu_e^r \geq \epsilon, \sum_{r \in \mathcal{R}_e} \mu_e^r \leq \mu_e, \forall e \in E, r \in \mathcal{R}_e\}. \quad (2)$$

Let $n_e^r \in \mathbb{N}$ be the queue size. We assume that traffic cost is a function of n_e^r and denote by $c_e^r(n_e^r) : \mathbb{N} \to \mathbb{R}_+$ the cost of response type r on edge e.

Our overall model is illustrated in Fig. 1. The global service rate assignment $\boldsymbol{\mu}$ and the global item placement $\boldsymbol{x}$ are design parameters: we wish to determine $\boldsymbol{x}$ and $\boldsymbol{\mu}$ jointly, to minimize expected traffic costs in steady state. The expected steady state distribution of queue sizes depends on the type of queues we use to model the network. In Section III-C, we describe the queues we consider formally, and in Section III-D we give a more precise definition of the optimization problem we solve. Before we do so, however, we briefly discuss how our work relates to Mahdian et al. [15] and the challenges we face in solving the corresponding optimization problem.

B. Choice of Queues and Challenges

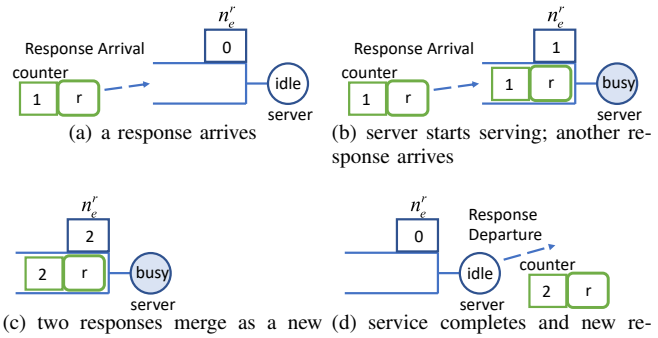
Mahdian et al. [15] consider a network very similar to the one we have proposed here, assuming queuing at edges happens via M/M/1 queues: all responses are served individually by the edge server. This is convenient from a modeling perspective, because M/M/1 queues form a Kelly network [16]. However, transmitting same-type responses individually over the same queue is both inefficient and impractical. If two responses of the same type are present in a queue, it suffices to transmit only one of them: requests pending at the same downstream source can be satisfied simultaneously by the same response. Transmitting responses individually leads to larger queue sizes, thereby incurring larger queuing costs, but also larger delays (by Little's Theorem [56]). In fact, the system considered by Mahdian et al. becomes unstable when the load of an M/M/1 queue is above one. This motivates us to introduce a new type of queue we call a *counting queue*. In such queues, responses of the same type *merge*: this is more realistic, but also reduces overall traffic, leading to a universally stable system.

Despite these advantages, introducing counting queues gives rise to two significant challenges when attempting to minimize the expected queuing costs. First, in contrast to Mahdian et al. [15], counting queues are *not reversible*, *do not form a Kelly network, and are thus hard to analyze in steady state*. As a result, our objective (namely, the aggregate expected steady-state queuing cost) is hard to compute efficiently and in a closed form. The second challenge arises from the combinatorial nature of the item placement space, combined with the continuous nature of the service rate assignment space. To summarize, the problem we are trying to solve is a mixed integer problem, with a non-convex objective that we cannot compute efficiently. This motivates us to introduce approximations, which we discuss in more detail below.

C. Queue Types

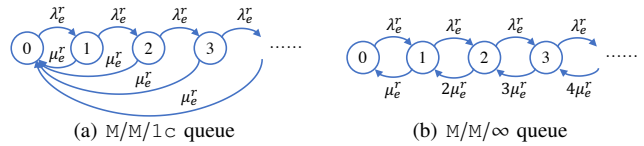
We introduce counting (or M/M/1c) queues [17] in this subsection, and describe how they relate to (and can be approximated by) M/M/∞ queues.

1) M/M/1c Queue: To model realistic behavior, a *counting queue* behaves as follows. When a response of type r arrives at an empty queue on edge e, it experiences immediate service with rate μ_e^r. A subsequent response of type r arriving before the server is finished is *not* queued: instead, it merges with the response in the server, and both are served simultaneously with rate μ_e^r. In practice, this is implemented as follows: every response is associated with a counter initialized to one by the designated server generating it. Whenever two responses of type r are collocated in an edge e, they merge into a new response of type r, with a counter equal to the sum of its constituent counters. Note that, as service times are exponential, by the memoryless property [57], the residual service time after a merge remains exponential with rate μ_e^r. After being served, this merged response departs. This whole process is depicted in Fig. 2. We formally refer to such a queue as an M/M/1c queue ('c' is for 'counter'). We consider the size n_e^r of an M/M/1c queue to be equal to the counter value of the merged response in the queue's server. Assuming Poisson arrivals of responses with counters equal to one, the queue size process $\{n_e^r(t); t \geq 0\}$ is a Markov process whose



(a) a response arrives (b) server starts serving; another response arrives (c) two responses merge as a new response (d) service completes and new response departs

Fig. 2: M/M/1c queue. When a response arrives in an idle M/M/1c queue, it is served immediately. If two responses meet in a queue before the end of service, they merge as a new response with counter value equal to the sum of their respective counters. The new response continues being served as before. After the service, the merged response departs the queue and queue is again idle. Queue size n_e^r equals the counter value of the response in this queue, and 0 if the queue is empty.



(a) M/M/1c queue (b) M/M/∞ queue

Fig. 3: M/M/1c and M/M/∞ transitions, assuming Poisson arrivals. M/M/1c is not reversible, while M/M/∞ is.

transition diagram is described in Fig. 3(a), and its steady state distribution is given by the following lemma. The proof is in Appendix A.

Lemma 1. *Assume that response arrivals follow a Poisson process with rate λ_e^r, and each response's counter is 1. Then, the steady state distribution of the* M/M/1c *queue is*

$$\boldsymbol{P}_{\mathtt{M/M/1c}}(n_e^r = n) = \left(\frac{\rho_e^r}{\rho_e^r + 1}\right)^n \frac{1}{\rho_e^r + 1}, \quad \textit{for all } n \in \mathbb{N}, \tag{3}$$

where $\rho_e^r = \frac{\lambda_e^r}{\mu_e^r}$ is load of response type r on edge e.

Note that this queue is *universally stable*, i.e., positive recurrent for all $\rho_e^r > 0$. However, Lemma 1 only holds for edges directly adjacent to a designated server. This is because intermediate queues, further from a designated server, satisfy neither of the two assumptions of Lemma 1: (a) arrivals are *not* Poisson, and (b) counters of responses may be *larger than 1*. Overall, the entire system is *not* a Kelly network, and its steady state distribution is difficult to describe in a closed form.

2) M/M/∞ *Queue:* The above state of affairs motivates us to approximate counting queues with M/M/∞ queues [58]. Recall that an M/M/∞ queue has infinite servers. Just as in an M/M/1c queue, incoming responses are not queued but are immediately served with service rate μ_e^r. However, the service times of responses collocated in an M/M/∞ queue are independent, while in M/M/1c they are tightly coupled: in fact, all responses are served simultaneously. Again, $\{n_e^r(t); t \geq 0\}$ is a Markov process whose transition diagram is described in Fig. 3(b), and its steady state distribution is given by the following classic lemma (see, e.g., [59]):

Lemma 2. *Assume that response arrivals follow a Poisson process with rate λ_e^r. Then, the steady state distribution of* M/M/∞ *queue is*

$$\boldsymbol{P}_{\mathtt{M/M/\infty}}(n_e^r = n) = \frac{(\rho_e^r)^n}{n!} e^{-\rho_e^r}, \quad \textit{for all } n \in \mathbb{N}, \tag{4}$$

where $\rho_e^r = \frac{\lambda_e^r}{\mu_e^r}$ is load of response type r on edge e.

Note that responses here do not merge and, hence, implicitly all have counter value one. A significant advantage of M/M/∞ queues is that they are reversible [57]. Hence, networks of such queues form a Jackson network [16], [43], [57]. In steady state, departures from these queues are Poisson by Burke's theorem [60]; as a result, in contrast to M/M/1c queues, the steady state distribution in Eq. (4) applies to *every edge* in the network, for an appropriately defined arrival rate λ_e^r (see Eq. (8) in Sec. III-D below).

3) Approximating M/M/1c *Queues with* M/M/∞ *Queues:* There are several reasons why M/M/∞ queues are good approximations of M/M/1c queues. First, observe that both queues are universally stable. Second, they exhibit the same aggregate service rate: when n_e^r customers are in the queue, the aggregate service rate in both is $n_e^r \mu_e^r$; put differently, in both queues the aggregate service rate grows linearly with the queue size. Finally, queue sizes of M/M/1c and M/M/∞ queues are related through a notion of mutual stochastic dominance. In particular, it is easy to confirm from Lemmas 1 and 2 that the two queues have the same expectation, i.e.,

$$\mathbb{E}_{\mathtt{M/M/1c}}[n_e^r] = \mathbb{E}_{\mathtt{M/M/\infty}}[n_e^r] = \rho_e^r. \tag{5}$$

More generally, all moments of the two queues are coupled through the following relationship:

Theorem 1. *Let $m^k_{\mathtt{M/M/1c}}(\rho_e^r) = \mathbb{E}_{\mathtt{M/M/1c}}[(n_e^r)^k]$ and $m^k_{\mathtt{M/M/\infty}}(\rho_e^r) = \mathbb{E}_{\mathtt{M/M/\infty}}[(n_e^r)^k]$ be the k-th moment of n_e^r in* M/M/1c *and* M/M/∞ *queues, respectively. Then, for all $\rho_e^r \geq 0$,*

$$m^k_{\mathtt{M/M/\infty}}(\rho_e^r) \leq m^k_{\mathtt{M/M/1c}}(\rho_e^r) \leq k! \cdot m^k_{\mathtt{M/M/\infty}}(\rho_e^r). \tag{6}$$

The proof can be found in Appendix B .This theorem immediately implies that, for any polynomial cost function $c_e^r(n_e^r)$, the expected costs under the two queues are within a multiplicative constant (not depending on ρ_e^r) of each other.[1] This, along with the ability to describe the joint steady state distribution across edges, motivates our approximation of M/M/1c queues by M/M/∞ queues.

D. Cache Cost Minimization

As discussed above, given item placements $\boldsymbol{x} \in \mathcal{D}$ and service rate assignments $\boldsymbol{\mu} \in \mathcal{D}_\mu$, the network of M/M/∞ queues is a Jackson network. Formally, in steady state, arrivals

[1] This result can be extended to continuous functions using, e.g., the Stone-Weierstrass Theorem [61].

TABLE I: Notation Summary

	Response type based counting queues
$\mathbb{R}, \mathbb{N}$	Sets of real and natural numbers
$\mathbb{R}_+, \mathbb{N}_+$	Sets of non-negative reals and positive naturals
$G(V, E)$	Network graph, with nodes V and edges E
$k_{p^r}(v)$	Position of node v in path p^r
c_v	Cache capacity at node $c \in V$
x_{vi}	Integer variable indicating $v \in V$ stores $i \in C$
$\boldsymbol{x}$	Global item placement vector of x_{vi}s in $\{0,1\}^{\lvert V\rvert\lvert C\rvert}$
$\mathcal{R}$	Set of types of requests
$\mathcal{R}_e$	Set of types of responses passing through e
λ_r	Request arrival rate for type $r \in \mathcal{R}$
μ_e	Service rate of edge $e \in E$
μ_e^r	Service rate of type $r \in \mathcal{R}_e$ over edge e
ϵ	The minimum service rate of all μ_e^r
$\boldsymbol{\mu}$	Global service rates vector of μ_e^rs in $\mathbb{R}_+^{\sum_e \lvert\mathcal{R}_e\rvert}$
ρ_e^r	Load of type r over edge e
$\mathcal{D}$	Set of feasible item placements $\boldsymbol{x}$
$\mathcal{D}_\mu$	Set of feasible service rates $\boldsymbol{\mu}$
$C_{\mathtt{M/M/\infty}}$	Cost function for $\mathtt{M/M/\infty}$ queues
$C_{\mathtt{M/M/1C}}$	Cost function for $\mathtt{M/M/1C}$ queues
C	Generalized non-decreasing and convex cost function
F	Caching gain of decision $\{\boldsymbol{x}, \boldsymbol{\mu}\}$ over $\{\mathbf{0}, \boldsymbol{\epsilon}\}$
$\tilde{\mathcal{D}}$	Convex hull of $\mathcal{D}$
y_{vi}	Probability that v stores i
$\boldsymbol{y}$	Vector of marginal probabilities y_{vi}s in $[0,1]^{\lvert V\rvert\lvert C\rvert}$
G	Multilinear extension with marginals $\boldsymbol{y}$
$[\boldsymbol{x}]_{+(v,i)}$	Vector $\boldsymbol{x}$ with the (v,i)-th coordinate set to 1
$[\boldsymbol{x}]_{-(v,i)}$	Vector $\boldsymbol{x}$ with the (v,i)-th coordinate set to 0
	Item based counting queues
$\mathcal{I}_e$	Set of items passing through e
$\mathcal{D}_{\mu'}$	Set of feasible service rates $\boldsymbol{\mu}$

of responses of type r on edge $e = (v,u)$ where $(u,v) \in p^r$ are Poisson with rate:

$$\lambda_e^r = \lambda_e^r(\boldsymbol{x}) = \lambda^r \prod_{k'=1}^{k_{p^r}(u)} (1 - x_{p^r_{k'} i^r}). \tag{7}$$

Intuitively, this states that responses of type r pass through edge $(v,u) \in E$ iff all path predecessors of node v do not store item i^r, i.e., $x_{v'i^r} = 0$ for all $v' : k_{p^r}(v') < k_{p^r}(v)$.

Let the state of a network of $\mathtt{M/M/\infty}$ queues be $\mathbf{n} = [n_e^r]_{e\in E, r\in\mathcal{R}_e}$, where n_e^r is the number of responses of type r on edge e. Then, for each $r \in \mathcal{R}$, the corresponding network is a Jackson network and, in particular, the steady state joint distribution has following product form:

$$\pi(\mathbf{n}) = \prod_{e\in E} \prod_{r\in\mathcal{R}} \pi_e^r(n_e^r), \tag{8}$$

where $\pi_e^r(n_e^r) = \frac{(\rho_e^r)^{n_e^r}}{(n_e^r)!} e^{-\rho_e^r}$, $n_e^r \in \mathbb{N}$, and the load of response type $r \in \mathcal{R}_e$ on $e = (v,u) \in E$ is:

$$\rho_e^r = \rho_e^r(\boldsymbol{x}, \mu_e^r) = \frac{\lambda^r}{\mu_e^r} \prod_{k'=1}^{k_{p^r}(u)} (1 - x_{p^r_{k'} i^r}). \tag{9}$$

Hence, in steady state, the expected cost of response type $r \in \mathcal{R}_e$ on edge $e \in E$, according to Lemma 2, is:

$$\mathbb{E}_{\mathtt{M/M/\infty}}[c_e^r(n_e^r)] = \sum_{n=0}^{\infty} c_e^r(n) \cdot e^{-\rho_e^r} \frac{(\rho_e^r)^n}{n!}, \tag{10}$$

Given a cache network represented by graph $G(V,E)$, service rate capacities μ_e, $e \in E$, storage capacities c_v, $v \in V$, a requests set $\mathcal{R}$ and arrival rates λ_r, $r \in \mathcal{R}$, we formulate the cache cost minimization problem under $\mathtt{M/M/\infty}$ queues as follows:

$$\text{MinCost}_{\mathtt{M/M/\infty}}$$

$$\min_{\boldsymbol{x},\boldsymbol{\mu}} : C_{\mathtt{M/M/\infty}}(\boldsymbol{x},\boldsymbol{\mu}) = \sum_{e\in E} \sum_{r\in\mathcal{R}_e} \mathbb{E}_{\mathtt{M/M/\infty}}[c_e^r(n_e^r)], \tag{11a}$$

$$\text{s.t.} : \boldsymbol{x} \in \mathcal{D},\ \boldsymbol{\mu} \in \mathcal{D}_\mu, \tag{11b}$$

where $\mathcal{D}$ is defined by Eq. (1) and $\mathcal{D}_\mu$ is defined by Eq. (2).

We can similarly define an optimization problem under $\mathtt{M/M/1C}$ queues. Supposing that the assumptions of Lemma 1 hold, the expected cost of an $\mathtt{M/M/1C}$ queue is:

$$\mathbb{E}_{\mathtt{M/M/1C}}[c_e^r(n_e^r)] = \sum_{n=0}^{\infty} c_e^r(n) \cdot \left(\frac{\rho_e^r}{\rho_e^r + 1}\right)^n \frac{1}{\rho_e^r + 1}. \tag{12}$$

where ρ_e^r are given by Eq. (9). We thus can again consider a cache cost minimization problem under $\mathtt{M/M/1C}$ queues, defined as:

$$\text{MinCost}_{\mathtt{M/M/1C}}$$

$$\min_{\boldsymbol{x},\boldsymbol{\mu}} : C_{\mathtt{M/M/1C}}(\boldsymbol{x},\boldsymbol{\mu}) = \sum_{e\in E} \sum_{r\in\mathcal{R}_e} \mathbb{E}_{\mathtt{M/M/1C}}[c_e^r(n_e^r)], \tag{13a}$$

$$\text{s.t.} : \boldsymbol{x} \in \mathcal{D},\ \boldsymbol{\mu} \in \mathcal{D}_\mu, \tag{13b}$$

We stress that *both* problems (11) and (13) are approximations of networks of counting queues. $\text{MinCost}_{\mathtt{M/M/\infty}}$ is clearly an approximation, as $\mathtt{M/M/\infty}$ queues are used instead of $\mathtt{M/M/1C}$ queues. The objective (11a) captures steady state costs in such a system accurately, as arrivals in intermediate queues are indeed Poisson. $\text{MinCost}_{\mathtt{M/M/1C}}$ is an approximation as the objective assumes Poisson arrivals and counters of size 1 at intermediate queues, neither of which are true for a real network of $\mathtt{M/M/1C}$ queues.

As we will see in Sec. VI, these approximations appear to perform well experimentally. Nevertheless, both problems are hard to solve; we prove the following in Appendix C:

Theorem 2. *Problems* (11) *and* (13) *are NP-hard.*

IV. Main Results

In this section, we show how to solve Problems (11) and (13) within a constant approximation, poly-time algorithm. Mahdian et al. [15] approach caching problems via submodular maximization. However, (11) and (13) can not be cast in this setting, as they have mixed constraints: we would like to determine not only item placements (integer variables) but also service rates (continuous variables). Nevertheless, we construct a $1 - 1/e$-approximation poly-time algorithm. A crucial step is that the so-called multilinear extensions of (11a) and (13a) are *jointly* DR-submodular [54] w.r.t. $\boldsymbol{x}$ and $\boldsymbol{\mu}$.

A. Cache Gain Maximization

We introduce the following assumption on cost functions:

Assumption 1. *For all $r \in \mathcal{R}$, $e \in E$, and $n \in \mathbb{N}_+$,*

$$c_e^r(n+1) - c_e^r(n) \geq c_e^r(n) - c_e^r(n-1) \geq 0.$$

Using this assumption, we establish the following property:

Lemma 3. *Under Assumption 1, the expected cost functions* $\mathbb{E}_{\mathrm{M/M/\infty}}[c_e^r(n_e^r)]$ *and* $\mathbb{E}_{\mathrm{M/M/1C}}[c_e^r(n_e^r)]$*, given by* (10) *and* (12)*, are non-decreasing and convex w.r.t. load* ρ_e^r*, given by* (9)*.*

The proof is in Appendix D. Motivated by Lemma 3, we consider a more general class of problems of the form:

MINCOST

$$\min_{\boldsymbol{x},\boldsymbol{\mu}} : C(\boldsymbol{x},\boldsymbol{\mu}) = \sum_{e\in E}\sum_{r\in\mathcal{R}_e} C_e^r(\rho_e^r(\boldsymbol{x},\mu_e^r)), \tag{14a}$$

$$\text{s.t.} : \boldsymbol{x}\in\mathcal{D},\ \boldsymbol{\mu}\in\mathcal{D}_\mu, \tag{14b}$$

where expected cost functions $C_e^r : \mathcal{D}\times\mathcal{D}_\mu \to \mathbb{R}_+$ are non-decreasing and convex. Clearly, by Lem. 3, an algorithm solving (14) can also solve both (11) and (13). Following [12], [14], we consider an equivalent maximization problem instead:

MAXGAIN

$$\max_{\boldsymbol{x},\boldsymbol{\mu}} : F(\boldsymbol{x},\boldsymbol{\mu}) = C(\mathbf{0},\boldsymbol{\epsilon}) - C(\boldsymbol{x},\boldsymbol{\mu}), \tag{15a}$$

$$\text{s.t.} : \boldsymbol{x}\in\mathcal{D},\ \boldsymbol{\mu}\in\mathcal{D}_\mu, \tag{15b}$$

where $\mathbf{0}\in\mathcal{D}$ is the empty cache placement, $\boldsymbol{\epsilon} = \epsilon\cdot\mathbf{1}\in\mathcal{D}_\mu$ is the vector of minimum service rates, and $C(\mathbf{0},\boldsymbol{\epsilon})$ is an upper bound on $C(\boldsymbol{x},\boldsymbol{\mu})$. MINCOST and MAXGAIN are equivalent, because (14a) and (15a) only differ by a constant $C(\mathbf{0},\boldsymbol{\epsilon})$.

B. DR-Submodularity

Let $\tilde{\mathcal{D}} = \text{conv}(\{\boldsymbol{x} : \boldsymbol{x}\in\mathcal{D}\}) \subseteq [0,1]^{|V||C|}$ be the convex hull of the constraint set $\mathcal{D}$. That is:

$$\tilde{\mathcal{D}} = \{\boldsymbol{y}\in[0,1]^{|V||C|} : \sum_{i\in C} y_{vi} \le c_v, \forall v\in V\}. \tag{16}$$

Given a $\boldsymbol{y}\in\tilde{\mathcal{D}}$, consider a random vector $\boldsymbol{x}$ generated as follows: every $x_{vi}\in\{0,1\}$ is an independent Bernoulli variable such that $\mathbf{P}(x_{vi}=1) = y_{vi}$. The multilinear extension [34] $G(\boldsymbol{y},\boldsymbol{\mu}) : \tilde{\mathcal{D}}\times\mathcal{D}_\mu\to\mathcal{R}_+$ of F is $\mathbb{E}_y[F(\boldsymbol{x},\boldsymbol{\mu})]$, i.e.:

$$G(\boldsymbol{y},\boldsymbol{\mu}) = \sum_{\boldsymbol{x}\in\{0,1\}^{|V||C|}} F(\boldsymbol{x},\boldsymbol{\mu})\times\prod_{(v,i)\in V\times C} y_{vi}^{x_{vi}}(1-y_{vi})^{1-x_{vi}}. \tag{17}$$

Given an $\mathcal{X}\subseteq\mathbb{R}^d$, we say that a function $f:\mathcal{X}\to\mathbb{R}$ is *DR-submodular* [18], if for all $\boldsymbol{a}\le\boldsymbol{b}\in\mathcal{X}$, and all $i\in\mathbb{N}$, $k\in\mathbb{R}_+$, s.t. $(k\boldsymbol{e}_i+\boldsymbol{a})$ and $(k\boldsymbol{e}_i+\boldsymbol{b})$ are in $\mathcal{X}$, we have $f(k\boldsymbol{e}_i+\boldsymbol{a})-f(\boldsymbol{a}) \ge f(k\boldsymbol{e}_i+\boldsymbol{b})-f(\boldsymbol{b})$. The following lemma establishes that G is DR-submodular over the extended domain $\tilde{\mathcal{D}}\times\mathcal{D}_\mu$:

Theorem 3. *If the expected cost functions* C_e^r *are non-decreasing and convex, the multilinear extension* G *is non-decreasing DR-submodular jointly on both* $\boldsymbol{\mu}$ *and* $\boldsymbol{y}$*.*

Proof. Let function $F(S,\boldsymbol{\mu}) \triangleq F(\boldsymbol{x}_S,\boldsymbol{\mu})$, $S = \{\text{supp}(\boldsymbol{x})\}$. We first introduce an auxiliary lemma and its proof is in App. E:

Lemma 4. *If the expected cost functions* C_e^r *are non-decreasing and convex, the set function* $F(S,\boldsymbol{\mu})$ *is: (a) non-decreasing concave on* $\boldsymbol{\mu}$ *and (b) non-decreasing submodular on set* S*.*

For convenience, we replace subscripts e and superscripts r by subscript $i\in E\times\sum_e\mathcal{R}_e$. By Lemma 4:

$$\frac{\partial G(\boldsymbol{y},\boldsymbol{\mu})}{\partial\mu_i} = \sum_{\boldsymbol{x}\in\{0,1\}^{|V||C|}}\frac{\partial F(\boldsymbol{x},\boldsymbol{\mu})}{\partial\mu_i}\times\prod_{(v,i)\in V\times C} y_{vi}^{x_{vi}}(1-y_{vi})^{1-x_{vi}} \ge 0,$$

and

$$\frac{\partial G(\boldsymbol{y},\boldsymbol{\mu})}{\partial y_i} = \mathbb{E}_y[F(\boldsymbol{x},\boldsymbol{\mu})|x_i=1] - \mathbb{E}_y[F(\boldsymbol{x},\boldsymbol{\mu})|x_i=0] \ge 0.$$

Thus G is non-decreasing in both $\boldsymbol{\mu}$ and $\boldsymbol{y}$. By Lemma 4, we get:

$$\frac{\partial^2 G(\boldsymbol{x},\boldsymbol{\mu})}{\partial\mu_i\partial\mu_j} = \sum_{\boldsymbol{x}\in\{0,1\}^{|V||C|}}\frac{\partial^2 F(\boldsymbol{x},\boldsymbol{\mu})}{\partial\mu_i\partial\mu_j}\times\prod_{(v,i)\in V\times C} y_{vi}^{x_{vi}}(1-y_{vi})^{1-x_{vi}} \le 0,$$

while, as shown in [34],

$$\begin{aligned}\frac{\partial^2 G(\boldsymbol{y},\boldsymbol{\mu})}{\partial y_i\partial y_j} =& \mathbb{E}_y[F(\boldsymbol{x},\boldsymbol{\mu})|x_i=1,x_j=1] - \mathbb{E}_y[F(\boldsymbol{x},\boldsymbol{\mu})|x_i=1,x_j=0]\\ &-\mathbb{E}_y[F(\boldsymbol{x},\boldsymbol{\mu})|x_i=0,x_j=1] + \mathbb{E}_y[F(\boldsymbol{x},\boldsymbol{\mu})|x_i=0,x_j=0]\\ &\le 0.\end{aligned} \tag{18}$$

We have, $\frac{\partial F(\boldsymbol{x},\boldsymbol{\mu})}{\partial\mu_i} = -\frac{\partial C_i(\rho_i)}{\partial\rho_i}\frac{\partial\rho_i}{\partial\mu_i} = \frac{\partial C_i(\rho_i)}{\partial\rho_i}\frac{\rho_i}{\mu_i} \ge 0$, and $\frac{\partial}{\partial\rho_i}\frac{\partial F(\boldsymbol{x},\boldsymbol{\mu})}{\partial\mu_i} = \frac{\partial^2 C_i(\rho_i)}{\partial\rho_i^2}\frac{\rho_i}{\mu_i} + \frac{\partial C_i(\rho_i)}{\partial\rho_i}\frac{1}{\mu_i} \ge 0$. So, $\frac{\partial F(\boldsymbol{x},\boldsymbol{\mu})}{\partial\mu_i}$ is non-decreasing w.r.t. ρ_i. Since ρ_i is non-increasing w.r.t. $\boldsymbol{x}$, $\frac{\partial F(\boldsymbol{x},\boldsymbol{\mu})}{\partial\mu_i}$ is non-increasing w.r.t. $\boldsymbol{x}$. Then,

$$\frac{\partial^2 G(\boldsymbol{y},\boldsymbol{\mu})}{\partial\mu_i\partial y_j} = \mathbb{E}_y[\frac{\partial F(\boldsymbol{x},\boldsymbol{\mu})}{\partial\mu_i}|x_j=1] - \mathbb{E}_y[\frac{\partial F(\boldsymbol{x},\boldsymbol{\mu})}{\partial\mu_i}|x_j=0] \le 0.$$

Hence, all of the entries of $G(\boldsymbol{x},\boldsymbol{\mu})$'s Hessian w.r.t. both $\boldsymbol{\mu}$ and $\boldsymbol{x}$ are non-positive, and G is DR-submodular [18]. □

This property is key; despite the fact that G is not concave, DR-submodularity implies we can maximize it within a constant factor. We stress here that the property holds jointly for $\boldsymbol{y}$ and $\boldsymbol{\mu}$, which is non-obvious.

C. Algorithm Overview

Leveraging Thm. 3, our algorithm consists of two steps:

Step 1: DR-submodular maximization. We first apply a variant of the Frank-Wolfe algorithm [54], summarized in Alg. 1, on the multilinear extension G. For brevity, we join $\boldsymbol{y}$ and $\boldsymbol{\mu}$ as one variable $\boldsymbol{z} = \{\boldsymbol{y},\boldsymbol{\mu}\}\in\mathcal{D}_z \equiv \{\boldsymbol{z}\in\tilde{\mathcal{D}}\times\mathcal{D}_\mu\}$. The algorithm first initializes the solution as $\boldsymbol{z} = \{\mathbf{0},\boldsymbol{\epsilon}\}$. Then, it iterates over the following steps:

$$\boldsymbol{m}_k \leftarrow \arg\max_{\boldsymbol{m}\in\mathcal{D}_z}\langle\boldsymbol{m},\widehat{\nabla G}(\boldsymbol{z}_k)\rangle, \tag{19a}$$

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \gamma_k\boldsymbol{m}_k, \tag{19b}$$

where $\widehat{\nabla G}(\boldsymbol{z}_k)$ is an estimate of the gradient of G, and γ_k is an appropriately chosen step size. An estimator of ∇G is needed because both ∇G and G, given by (17), contain an exponential (in $|V||C|$) number of terms. We describe this estimator in detail in Section IV-D. Given $\widehat{\nabla G}(\boldsymbol{z}_k)$, (19) is a linear program, which can be solved in polynomial time [54]. After K iterations, we get a fractional solution $\boldsymbol{z}_K = \{\boldsymbol{y}_K,\boldsymbol{\mu}_K\}$, i.e., the output of Alg. 1.

Step 2: Rounding. Finally, the fractional solution $\boldsymbol{y}_K$ is

Algorithm 1: Frank-Wolfe variant for $G(\boldsymbol{z})$

Input: $G(\boldsymbol{z})$, $\mathcal{D}_z$, step size $\gamma \in (0,1]$, initial point $\{\mathbf{0}, \boldsymbol{\epsilon}\}$.

$t \leftarrow 0$, $k \leftarrow 0$, $\boldsymbol{z}_0 \leftarrow \{\mathbf{0}, \boldsymbol{\epsilon}\}$

while $t < 1$ **do**

  $\boldsymbol{m}_k \leftarrow \arg\max_{\boldsymbol{m} \in \mathcal{D}_z} \left\langle \boldsymbol{m}, \widehat{\nabla G}(\boldsymbol{z}_k) \right\rangle$

  $\gamma_k \leftarrow \min\{\gamma, 1-t\}$

  $\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \gamma_k \boldsymbol{m}_k$, $t \leftarrow t + \gamma_k$, $k \leftarrow k+1$

end

return $\boldsymbol{z}_k$

rounded into an integer solution $\boldsymbol{x}_K$. We describe how to do this in Sec. IV-E. This produces an approximate solution $\boldsymbol{x}_K$ and $\boldsymbol{\mu}_K$ to MAXGAIN.

Intuitively, DR-sumbodular maximization step (19) solves:

$$\max_{\boldsymbol{y},\boldsymbol{\mu}} : G(\boldsymbol{y}, \boldsymbol{\mu}), \tag{20a}$$

$$s.t. : \boldsymbol{y} \in \tilde{\mathcal{D}},\ \boldsymbol{\mu} \in \mathcal{D}_\mu, \tag{20b}$$

where $\tilde{\mathcal{D}}$ is defined by (16). Alg. 1/Eq. (19) only solves (20) approximately because objective (20a) is not concave. Nevertheless, because (20a) is DR-submodular by Lemma 3, Alg. 1 produces a $1-1/e$ approximation to (20); see Lemma 8 for details.

Combined with the rounding step, the following theorem characterizes the approximation guarantee of the overall algorithm:

Theorem 4. *Let $\boldsymbol{x}^*$, $\boldsymbol{\mu}^*$ be an optimal solution to* (15)*, $\boldsymbol{\mu}_K$ be the output of Frank-Wolfe variant, and $\boldsymbol{x}_K$ the integer solution after rounding. Then, with high probability,*

$$\mathbb{E}[F(\boldsymbol{x}_K, \boldsymbol{\mu}_K)] \geq (1 - \frac{1}{e}) F(\boldsymbol{x}^*, \boldsymbol{\mu}^*). \tag{21}$$

The "with high probability" is w.r.t. the randomness of the estimator, while the expectation in (21) is w.r.t. the randomness in the rounding step. The proof is in Appendix F.

D. Estimator of Gradient

Eq. (19a) presumes access to the gradient ∇G. Nonetheless, both G and ∇G involve a summation over $2^{|V||C|}$ terms. To create a poly-time algorithm, the usual approach is to use a sampling-based estimator [34]. In short, the partial derivatives of G w.r.t. y_{vi} and μ_e^r are (see [34] for (22)):

$$\frac{\partial G(\boldsymbol{y}, \boldsymbol{\mu})}{\partial y_{vi}} = \mathbb{E}_y[C(\boldsymbol{x}, \boldsymbol{\mu}) | x_{vi} = 0] - \mathbb{E}_y[C(\boldsymbol{x}, \boldsymbol{\mu}) | x_{vi} = 1], \tag{22}$$

$$\frac{\partial G(\boldsymbol{y}, \boldsymbol{\mu})}{\partial \mu_e^r} = \frac{1}{\mu_e^r} \mathbb{E}_y[\frac{\partial C_e^r(\rho_e^r)}{\partial \rho_e^r} \cdot \rho_e^r]. \tag{23}$$

One can thus estimate the gradient by (a) producing T random samples $\boldsymbol{x}^{(l)}$, $l = 1, ..., T$ of the random vector $\boldsymbol{x}$, consisting of independent Bernoulli coordinates with $\mathbf{P}(x_{vi} = 1) = y_{vi}$, and (b) computing the empirical mean w.r.t. y_{vi}:

$$\widehat{\frac{\partial G(\boldsymbol{y}, \boldsymbol{\mu})}{\partial y_{vi}}} = \frac{1}{T} \sum_{l=1}^{T} (C([\boldsymbol{x}^l]_{-(v,i)}, \boldsymbol{\mu}) - C([\boldsymbol{x}^l]_{+(v,i)}, \boldsymbol{\mu})), \tag{24}$$

where $[\boldsymbol{x}^l]_{-(v,i)}$, $[\boldsymbol{x}^l]_{+(v,i)}$ are equal to vector $\boldsymbol{x}$ with the (v,i)-th coordinate set to 0 and 1, respectively, and w.r.t μ_e^r:

$$\widehat{\frac{\partial G(\boldsymbol{y}, \boldsymbol{\mu})}{\partial \mu_e^r}} = \frac{1}{T \mu_e^r} \sum_{l=1}^{T} \frac{\partial C_e^r(\rho_e^r([\boldsymbol{x}^l], \mu_e^r))}{\partial \rho_e^r([\boldsymbol{x}^l], \mu_e^r)} \cdot \rho_e^r([\boldsymbol{x}^l], \mu_e^r). \tag{25}$$

According to Calinescu et al. [34], for the (with high probability) $1 - 1/e$ approximation ratio, $O((|V||C|)^2 \ln(|V||C|))$ samples suffice. There are other ways to estimate the gradient, e.g., via a Taylor expansion [15]. This is more efficient, so we also use it in Sec. VI. We refer readers to [15] for more details.

E. Swap Rounding

We review swap rounding [62], which is a probabilistic rounding step. Given a fractional $\boldsymbol{y}_K$, it can be written as a convex combination of some integer vectors $\boldsymbol{B}_l$, i.e., $\boldsymbol{y}_K = \sum_{l=1}^{L} \beta_l \boldsymbol{B}_l$, where $\sum_{l=1}^{L} \beta_l = 1$, $\beta_l \geq 0$, and $\boldsymbol{B}_l \in \mathcal{D}$. By construction, each $\boldsymbol{B}_l$ is maximal. This algorithm iteratively merges these $\boldsymbol{B}_l$, each iteration one $\boldsymbol{B}_l$, to produce a new integer solution $\boldsymbol{x}_l$, until $\boldsymbol{x}_l$ equal to $\boldsymbol{B}_{l+1}$. If $\boldsymbol{x}_{l'}$ differs $\boldsymbol{B}_{l'+1}$ by an item i in v, the item i replaces another different item j in $\boldsymbol{B}_{l'+1}$ with probability proportional to $\sum_{l=1}^{l'} \beta_l$, or another different item j in $\boldsymbol{B}_{l'+1}$ replaces item i in $\boldsymbol{x}_{l'}$ with probability proportional to $\beta_{l'+1}$. Swap rounding ensures that the objective does not decrease in expectation during rounding, i.e.,

$$\mathbb{E}[G(\boldsymbol{x}_K, \boldsymbol{\mu}_K)] \geq G(\boldsymbol{y}_K, \boldsymbol{\mu}_K). \tag{26}$$

The algorithm terminates in at most $O(|V||C|)$ steps.

F. Time Complexity

To ensure Thm. 4 holds, the number of samples T used for sample-based estimator of gradient is $O((|V||C|)^2 \ln(|V||C|))$ [15]. Each sample requires at most $O(|E||\mathcal{R}|)$ operations. Given a gradient, (19) requires polynomial time in the number of constraints and variables, which are $O(|V||C| + |E||\mathcal{R}|)$. We iterate (19) at most $O(|V||C|)$ times [15]. The rounding schemes presented in Sec. IV-E are also poly-time, i.e., at most $O(|V||C|)$ steps. In summary, the overall time complexity of our algorithm is $O(|E||\mathcal{R}|(|V||C|)^3 \ln(|V||C|))$.

V. EXTENSIONS

A. Multi-Item Requests

We can easily extend our model to requests with multiple items (see also [63]). Such multi-item requests, where several items are requested simultaneously, can be used to model requests for files of different size that are partitioned to equally-sized chunks. In this case, an item can be thought of as modeling a chunk, a file is a set of chunks, and the catalog comprises the union of all chunks across all files.

Formally, a multi-item request $r = (\mathcal{I}^r, p^r) \in \mathcal{R}$ is determined by (a) a set of requested items $\mathcal{I}^r = \{i_l^r\}_{l=1}^{L} \subseteq C$ (the file), and (b) the followed path p^r. Again, we assume that requests of type r are generated according to a Poisson process with rate λ^r. This can be directly analyzed using our model by observing that a single multi-item request is equivalent to multiple simultaneous single-item requests, namely, (i_1^r, p^r), (i_2^r, p^r), ..., (i_L^r, p^r), where $i_l^r \in \mathcal{I}^r$; the latter

are generated (simultaneously) at Poisson epochs with rate λ^r. Although these request arrival processes are not independent, the expected cost C under $\mathtt{M/M/\infty}$ and $\mathtt{M/M/1c}$ queues still have the form as objectives (11a) and (13a), where:

$$\mathbb{E}_{\mathtt{M/M/\infty}}[c_e^r(n_e^r)] = \sum_{r'=(i,p^r),i\in\mathcal{I}^r} \mathbb{E}_{\mathtt{M/M/\infty}}[c_e^{r'}(n_e^{r'})],$$

$$\mathbb{E}_{\mathtt{M/M/1c}}[c_e^r(n_e^r)] = \sum_{r'=(i,p^r),i\in\mathcal{I}^r} \mathbb{E}_{\mathtt{M/M/1c}}[c_e^{r'}(n_e^{r'})].$$

This is because, (a) the dependence between requests does affect their steady state marginal distributions, and (b) our objective is the sum of expectations, which is the same irrespective of whether the constituent r.v.s are dependent or not. Hence, we can analyze multi-item requests via *independent* single-item arrivals, and our analysis, algorithms, and guarantees directly extend to the multi-item requests setting.

B. Queuing on Nodes

Our model also readily extends to introducing queuing on nodes. Given all the paths $\{p^r, r \in \mathcal{R}\}$, for every node $v \in V$, we denote by $\mathcal{R}_v$ the set of response types passing through node v, i.e.,

$$\mathcal{R}_v = \{r \in \mathcal{R} : v \in p^r\}.$$

In this extension, each node $v \in V$ is be associated with a service rate $\mu_v \in \mathbb{R}_+$. The service rate in a node is again split across response types, items, etc. For example, in the case of response-based merging, arrivals of responses of type r on node v are Poisson with rate:

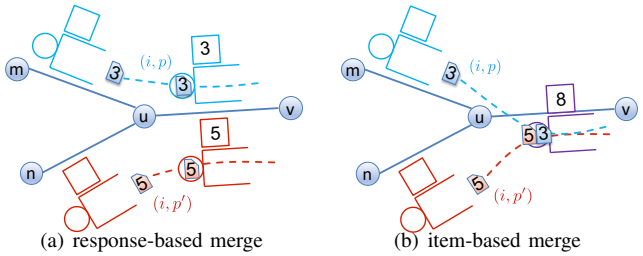
$$\lambda_v^r = \lambda_v^r(\boldsymbol{x}) = \lambda^r \prod_{k'=1}^{k_{p^r}(v)-1} (1 - x_{p^r_{k'},i^r}).$$

Intuitively, this states that responses of type r pass through node $v \in V$ iff all path predecessors of node v do not store item i^r, i.e., $x_{v'i^r} = 0$ for all $v' : k_{p^r}(v') < k_{p^r}(v)$. The remaining model can be changed correspondingly, mutatis mutandis. The same approximation guarantee $(1-1/e)$ can be obtained in this setting, again exploiting DR-submodularity.

C. Item-based Counting Queues

So far, we have considered merging responses of the same type. Our model and analysis can however be extended to counting queues merging responses *based on the item they carry*. In particular, if a response of type $r = (i, p)$ arrives in a queue serving a response of type $r' = (i, p')$, they both carry the same item and, therefore, can be simultaneously served. This creates additional opportunities for preserving bandwidth; however, it comes at the expense of additional complexity to keep track of how items associate to responses.

In more detail, instead of having counting queues associated with every response $r \in \mathcal{R}$ traversing an edge, each edge e is associated with counting queues *per item* $i \in C$, each with service rate μ_e^i. Again, these can be restricted to items i whose responses traverse e. When a response of type r carrying item i arrives at an empty queue on edge e, it receives service with rate μ_e^i immediately. If before the service terminates, another response carrying item i arrives, even if of a different type r', the two are merged. Instead of queuing up, the new response merges with the response already in the server and they are served simultaneously with rate μ_e^i.



(a) response-based merge (b) item-based merge

Fig. 4: An example illustrating differences between response-based counting queues and item-based counting queues. In both queues, 3 responses of type $r = (i, p)$ and 5 responses of type $r' = (i, p')$ arrive at edge (v, u) simultaneously. (a) In a response-based merge, response r and response r' are served in separate queues. (b) In item-based merge, response r and r' merge together as a new response, in a common queue over edge (v, u), while keeping track of provenance at the packet header. After the end of service, the merged response is split by node u and routed to the appropriate paths for each type, as paths for type r and r' diverge.

This item-based merge behavior further decreases traffic in the network, as more responses merge compared to the response-based counting queues. Nevertheless, merged responses need to keep track of provenance; the merged response needs to keep track not only of the count of responses containing i, but also how many are associated with each type $r \in \mathcal{R}$. The paths associated with each type $r \in \mathcal{R}$ are also documented. This information is stored in the header of responses. In particular, the merged response stores sub-counters for every type of response, along with the corresponding paths in its header. The merged response is served together in downstream edges as long as paths are common. In addition, when the two paths diverge, nodes need to split merged responses, and route them accordingly; this is a more complex operation than the simple routing performed in per-request merging. This is illustrated in Fig. 4(b). In this example, 3 responses of type $r = (i, p)$ and 5 responses of type $r' = (i, p')$ have accumulated at $e = (v, u)$; the response header keeps track of this break down. As paths p, p' diverge at u, when the service completes, the receiving node u splits the merged response into two responses: one w.r.t r, with counter 3, and one w.r.t r', with counter 5, and routes them to edge (u, m) and (u, n), respectively, according to the information stored in the response header.

Our model and analysis can be extended to incorporate this behavior as follows. Given the requests $r = (i^r, p^r) \in \mathcal{R}$, for every edge $e \in E$, we denote by $\mathcal{I}_e$ the set of items that may potentially pass through edge e, i.e., for $e = (v, u)$,

$$\mathcal{I}_{(v,u)} = \{i^r \in C : (u, v) \in p^r, (i^r, p^r) \in \mathcal{R}\}.$$

The service rate in an edge is split across items, rather than response types. For every item $i \in \mathcal{I}_e$, there exists a queue

with service rate $\mu_e^i \in \mathbb{R}_+$. Assume that the minimum service rate for all queues is some small $\epsilon \in \mathbb{R}_+$. Let vector $\boldsymbol{\mu} = [\mu_e^i]_{e\in E, i\in\mathcal{I}_e} \in \mathbb{R}_+^{\sum_e |\mathcal{I}_e|}$ be the global service rate assignment vector. We denote the set of feasible assignments by:

$$\mathcal{D}_{\mu'} = \{\boldsymbol{\mu} \in \mathbb{R}_+^{\sum_e |\mathcal{I}_e|} : \mu_e^i \geq \epsilon, \sum_{i\in\mathcal{I}_e} \mu_e^i \leq \mu_e, \forall e \in E, i \in \mathcal{I}_e\}. \quad (27)$$

Let $n_e^i \in \mathbb{N}$ be the queue size. We assume that queuing cost is a function of n_e^i and denote by $c_e^i(n_e^i) : \mathbb{N} \to \mathbb{R}_+$ the cost of responses carrying item i on edge e.

Just as in the response-based counting queues network, our item-based counting queues network does not have a product form steady state distribution; we cannot characterize the latter easily in a closed form. However, when approximated through M/M/∞ queues, the network becomes again amenable to analysis. In contrast to response-based routing, where the approximation resulted in multiple separate Jackson networks (one per type $r \in \mathcal{R}$), the resulting network under M/M/∞ is now *a Kelly* network [16], i.e., a multi-class Jackson network, where classes are determined by types $r \in \mathcal{R}$. Again, in steady state, the joint distribution has a product form and departures are Poisson [57]: as a result, arrivals of responses carrying item i on edge $e = (v, u)$ where $(u, v) \in p^r$ are Poisson with rate:

$$\lambda_e^i = \sum_{r=(i,p)\in\mathcal{R}_e} \lambda_e^r(\boldsymbol{x}) = \sum_{r=(i,p)\in\mathcal{R}_e} \lambda^r \prod_{k'=1}^{k_p(u)} (1 - x_{p_{k'}i}).$$

The load of responses carrying item $i \in \mathcal{I}_e$ on $e = (v, u) \in E$ is:

$$\rho_e^i = \rho_e^i(\boldsymbol{x}, \mu_e^i) = \frac{1}{\mu_e^i} \sum_{r=(i,p)\in\mathcal{R}_e} \lambda^r \prod_{k'=1}^{k_p(u)} (1 - x_{p_{k'}i}).$$

Given a cache network by graph $G(V, E)$, service rate capacities μ_e, $e \in E$, storage capacities c_v, $v \in V$, a requests set $\mathcal{R}$ and arrival rates λ_r, $r \in \mathcal{R}$, we formulate the cache cost minimization problem under item-based M/M/∞ queues as follows:

ItemBasedMinCost$_{\text{M/M/}\infty}$

$$\min_{\boldsymbol{x},\boldsymbol{\mu}} : C_{\text{M/M/}\infty}(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{e\in E} \sum_{i\in\mathcal{I}_e} \mathbb{E}_{\text{M/M/}\infty}[c_e^i(n_e^i)], \quad (28a)$$

$$\text{s.t.} : \boldsymbol{x} \in \mathcal{D},\ \boldsymbol{\mu} \in \mathcal{D}_{\mu'}, \quad (28b)$$

where $\mathcal{D}$ is defined by Eq. (1) and $\mathcal{D}_{\mu'}$ is defined by Eq. (27).

Similarly, if M/M/1c queues with Poisson arrivals and counters being 1, we also consider a cache cost minimization problem under item-based M/M/1c queues, defined as:

ItemBasedMinCost$_{\text{M/M/1c}}$

$$\min_{\boldsymbol{x},\boldsymbol{\mu}} : C_{\text{M/M/1c}}(\boldsymbol{x}, \boldsymbol{\mu}) = \sum_{e\in E} \sum_{i\in\mathcal{I}_e} \mathbb{E}_{\text{M/M/1c}}[c_e^i(n_e^i)], \quad (29a)$$

$$\text{s.t.} : \boldsymbol{x} \in \mathcal{D},\ \boldsymbol{\mu} \in \mathcal{D}_{\mu'}. \quad (29b)$$

Both problems (28) and (29) can be solved by techniques we proposed before, since all of statements in Sec. IV hold by just replacing superscript r with i. This is because we have simply changed how arrival rates are aggregated (on a per item rather than per request basis). In particular, constraints on the loads per edge remain affine in these arrival rates. Under Assumption 1, the expected cost function are non-decreasing and convex w.r.t. load ρ_e^i. Similarly, we can reformulate those two problems in a more general class of involving a non-decreasing, convex function of the load. The multilinear extension of this objective is again non-decreasing DR-submodluar jointly on both service assignment and item placement, and our Frank-Wolfe variant algorithm (Alg. 1), and rounding algorithm yield the same approximation and complexity guarantees.

topologies	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert C\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert Q\rvert$	c_v
ER	100	1042	1000	5000	4	50
ER-20Q	100	1042	100	1000	20	2
star	100	198	1000	5000	4	50
HC	128	896	100	1000	4	2
HC-20Q	128	896	100	1000	20	2
dtelekom	68	546	1000	5000	4	50
abilene	9	26	1000	5000	4	50
geant	22	66	1000	5000	4	50

TABLE II: Graph Topologies and Experiment Parameters.

VI. Experiments

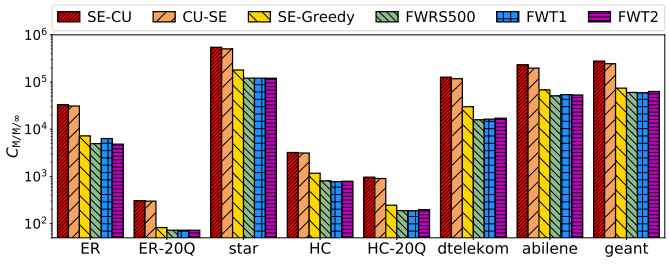
A. Experiment Setting

We execute our algorithms on Erdős-Rényi (ER), star, hypercube (HC), Deutsche Telekom (dtelekom), GEANT, and Abilene backbone networks [64]. The graph parameters of different topologies are shown in Tab. II. Each node $v \in V$ has c_v storage to cache item from a catalog of size $|C|$. Each item $i \in C$ is stored permanently in one designated server $\mathcal{S}_i$ which is picked uniformly at random (u.a.r.) from V. Also, we u.a.r. select $|Q|$ nodes from V as query nodes. Each of them generates around $\lfloor|\mathcal{R}|/|Q|\rfloor$ requests. For each response type $r \in \mathcal{R}$, rate λ^r is uniformly distributed over $[1.0, 2.0]$. The item i^r requested by r is chosen from catalog C via a power law distribution with exponent 1.2. The path p^r is the shortest path between the query node $p_1^r \in Q$ and designated server $p_K^r \in \mathcal{S}_{i^r}$. We set $\mu_e = 200.0$ at each edge e, and $\epsilon = 0.1$. Cost functions $c_e(n_e^r)$ are moments of the queue size $\mathbb{E}[(n_e^r)^k]$, where $k = 1, 2, 3, 4$.

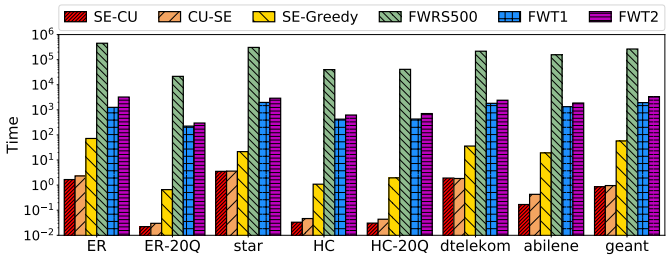
We conduct two types of experiments. (i) In the *offline* setting, we compute the expected costs $C_{\text{M/M/}\infty}$ and $C_{\text{M/M/1c}}$ according to (11a) and (13a), respectively. (ii) In the online setting: we simulate packets in M/M/1c and M/M/∞ queues network, and compute the time-average cost. More specifically, we monitor queues status at epochs t_s of a Poisson process with rate 1.0, leveraging PASTA [65], for 5000 time slots. For N measurements, the time average cost is:

$$\bar{C}_\cdot = \frac{1}{N} \sum_{s=0}^{N} \sum_{e\in E} \sum_{r\in\mathcal{R}_e} c_e^r(n_e^r(t_s)),$$

where $\cdot \in \{\text{M/M/}\infty, \text{M/M/1c}\}$ indicates on the type of queues simulated, and $n_e^r(t_s)$ is queue size at epoch t_s.

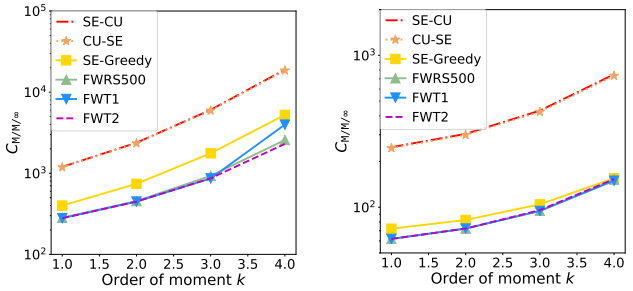


(a) Cost



(b) Execution Time

Fig. 5: Cost and execution time for different topologies and algorithms for quadratic costs.



(a) `dtelekom` (b) `ER20Q`

Fig. 6: Cost at different order of moments. Our algorithms achieve the lowest cost in different cost functions.

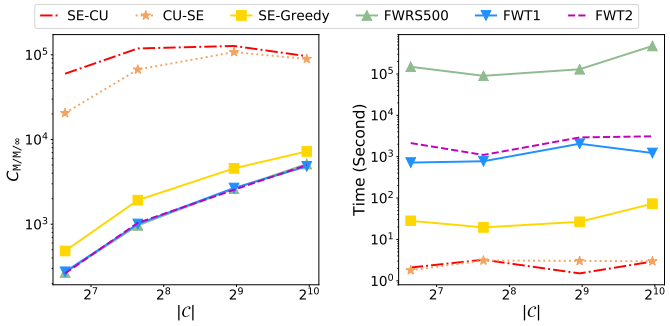


Fig. 7: Cost and execution time at different catalog sizes $|C|$ for quadratic costs over the `star` topology. Our algorithms achieve the lowest cost across different catalog sizes. Execution time of `FW` is higher than other (sub-optimal) algorithms, with random sampling being more cost-intensive than Taylor approximations. Nevertheless, the execution time scales polynomially with $|C|$.

B. Cache and Service Rate Algorithms

We compare to both online and offline algorithms. Offline algorithms are: (a) *Service Equally-Cache Uniformly* (SE-CU): first equally assign service rates for $\mathcal{R}_e$ over all $e \in E$ and then uniformly place items in each node. (b) *Cache Uniformly-Service Equally* (CU-SE): first uniformly place items in each node and then equally assign service rates for responses passing through edge e for all $e \in E$. Note that service rates depend on item placements in CU-SE. (c) *Service Equally-Greedy* (SE-Greedy): first equally assign service rates for $\mathcal{R}_e$ over all $e \in E$ and then use the classic greedy algorithm [50] for item placements. (d) *Frank-Wolfe with 500 Random Samples* (FWRS500): Alg. 1 with gradient estimated by 500 random samples. (e) *Frank-Wolfe with 1st/2nd order Taylor expansion* (FWT1/FWT2): Alg. 1 with gradient estimated by the first and the second order Taylor expansion respectively (c.f. [15]).

We also consider online algorithms, in which service rates are determined in advance and item placements change dynamically. As in the offline algorithms, we consider two service strategies: (a) *Service Equally* (SE): equally assign service rates for $\mathcal{R}_e$ over all $e \in E$, (b) μ_{FW}: service rates $\boldsymbol{\mu}$ calculated by Alg. 1. We combine these with item placements based on path replication [10]: when responses are back-propagated over the reverse path, nodes they encounter store requested items, evicting items via LRU, LFU, or FIFO eviction policies.

C. Results

Different Topologies. The cache cost $C_{\mathrm{M/M/\infty}}$ and the running time generated by different algorithms for quadratic costs is shown in Fig. 5(a) and 5(b). SE-Greedy improves over SE-CU and CU-SE, nevertheless, FW algorithms yield further improvements. As in [15], Taylor approximations are faster than sampling, without a loss in performance.

Different Cost Functions. The impact of the cost function on different algorithms is shown over `dtelekom` and `ER-20Q` in Fig. 6(a) and 6(b). Consistently with Fig. 5, FW outperforms competitors, with SE-Greedy being a close second. FWT1 performance degrades at the 4th moment in Fig. 6(a) due to the poor quality of the 1st order Taylor expansion.

Different Catalog Sizes. The impact of the catalog size on different algorithms for quadratic costs over the `star` topology is shown in Fig. 7. Consistently with Fig. 5, FW outperforms competitors in terms of expected cost, with SE-Greedy being a close second. Recall that catalog sizes do not affect the performance guarantee, as stated by Thm. 4. The execution time of `FW` increases polynomially with the growth of catalog sizes $|C|$, with an estimated exponent close to 2 (see the right figure in Fig. 7. This is under the worst-case time complexity of $O(|C|^3 \log |C|)$ anticipated by the analysis in Sec. IV-F.

Different Query Node Set Sizes. The impact of the query node size on different algorithms for quadratic costs over the `ER` topology is shown in Fig. 8. We scale $|Q|$ while keeping the number of requests per querying node fixed (to 400 requests

TABLE III: Expected costs and time-average costs under different topologies for quadratic cost functions.

	ER	ER-20Q	star	HC	HC-20Q	dtelekom	abilene	geant
$C_{\mathtt{M/M/1/C}}(\boldsymbol{z}_{\mathtt{M/M/\infty}})$	331.56	79.92	3827.04	1111.59	226.51	615.56	1540.13	1696.72
$\bar{C}_{\mathtt{M/M/1/C}}(\boldsymbol{z}_{\mathtt{M/M/\infty}})$	339.57	81.23	3879.81	1202.26	244.71	622.34	1713.98	1859.67
$C_{\mathtt{M/M/1/C}}(\boldsymbol{z}_{\mathtt{M/M/1/C}})$	331.77	79.77	3842.19	1115.24	212.26	619.85	1569.91	1707.47
$\bar{C}_{\mathtt{M/M/1/C}}(\boldsymbol{z}_{\mathtt{M/M/1/C}})$	341.22	80.88	3871.82	1198.67	228.25	623.15	1758.96	1884.49

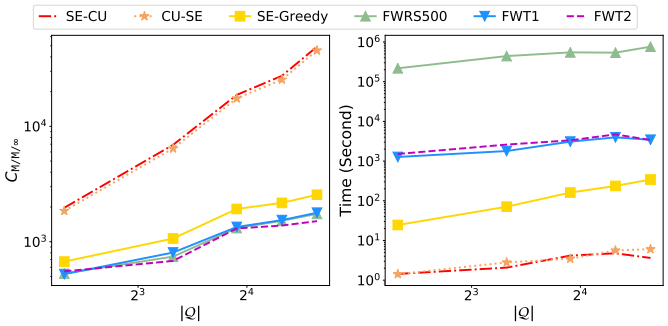


Fig. 8: Cost and execution time at different query node set sizes $|\mathcal{Q}|$ for quadratic costs over the ER topology. Our algorithms achieve the lowest cost across different query node set sizes. We again observe that execution time is higher than suboptimal algorithms, but scales polynomially with $|\mathcal{Q}|$.

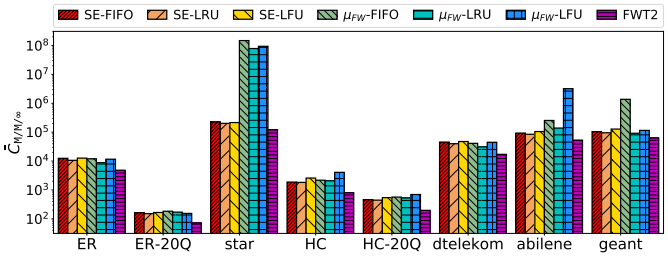
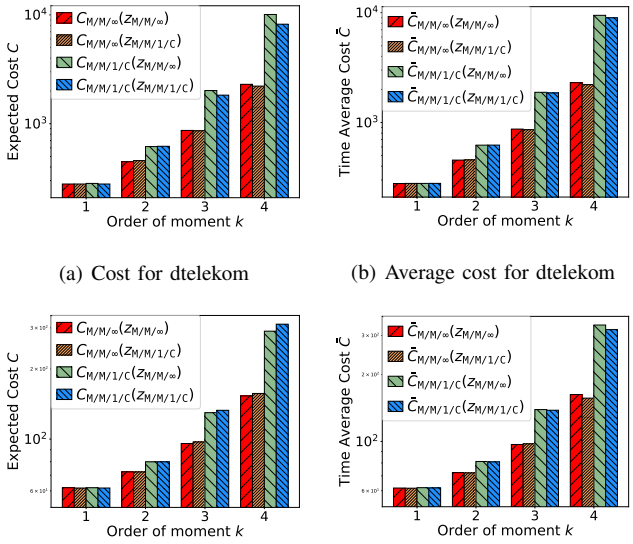


Fig. 9: Time average cost for different typologies and algorithms under quadratic costs.



(a) Cost for dtelekom (b) Average cost for dtelekom

(c) Cost for ER-20Q (d) Average cost for ER-20Q

Fig. 10: Expected costs and time-average costs in $\mathtt{M/M/\infty}$ v.s. $\mathtt{M/M/1C}$ queue networks. The two approximate models obtain similar, and good, solutions.

per node). As a result, when growing the number of sources we also grow the demand proportionally (i.e., $|\mathcal{R}| = 400|\mathcal{Q}|$). Again, FW variants outperform all competitors w.r.t. expected cost, with SE-Greedy being a close second. We also observe that execution time increases polynomially w.r.t. the number of query nodes $|\mathcal{Q}|$, because we increase $|\mathcal{R}|$ at the same time. This is slightly below the worst-case time complexity of $O(|\mathcal{R}|)$ anticipated by the analysis in Sec. IV-F.

Online Algorithms. Fig. 9 compares FWT2 to online algorithms under quadratic costs over $\mathtt{M/M/\infty}$ queues. FWT2 achieves the lowest time average costs $\bar{C}_{\mathtt{M/M/\infty}}$. Eviction algorithms with μ_{FW} are worse than SE most of the time. This means good performance of FWT2 comes from joint optimization. Note that time average costs of FWT2 in Fig. 9 and expected costs in Fig. 5(a) are almost identical, which verifies the reliability of our experiments from another perspective.

Comparing $\mathtt{M/M/1C}$ and $\mathtt{M/M/\infty}$ Queues. Finally, we confirm the quality of our two approximations of $\mathtt{M/M/1C}$ queues experimentally. Our goal is to (i) understand how well expected cost objectives (11a) and (13a) capture the time average costs $\bar{C}_{\mathtt{M/M/1C}}$, and (ii) assess the quality of solutions $\boldsymbol{z}_{\mathtt{M/M/1C}}$ and $\boldsymbol{z}_{\mathtt{M/M/\infty}}$ to Problems (13) and (11), respectively.

To that end, we plot both the expected cost objectives $C_{\mathtt{M/M/1C}}$, $C_{\mathtt{M/M/\infty}}$, as well as the time averages $\bar{C}_{\mathtt{M/M/1C}}$, $\bar{C}_{\mathtt{M/M/\infty}}$ for the two inputs $\boldsymbol{z}_{\mathtt{M/M/1C}}$ and $\boldsymbol{z}_{\mathtt{M/M/\infty}}$ in Fig. 10. We make the following broad observations. First, expected costs (Fig. 10(a) and 10(c)) are almost identical to time average costs (Fig. 10(b) and 10(d)). This is anticipated for $\mathtt{M/M/\infty}$ queues, that form a Jackson network, but is not obvious for $\mathtt{M/M/1C}$ queues: we assume Poisson arrivals with counters equal to 1 to formulate our approximate problems. To be more specific, in Fig. 10(b) and Fig. 10(d), the time average cost $\bar{C}_{\mathtt{M/M/1C}}$ resulting from simulation of (original) $\mathtt{M/M/1C}$ queue networks is almost identical to the expected cost $C_{\mathtt{M/M/1C}}$, computed via the objective for Problem (14), shown in Fig. 10(a) and Fig. 10(c). This indicates that, in practice, they are good approximations of the time-average cost of the original system. Second, cost functions $C_{\mathtt{M/M/\infty}}$ and $C_{\mathtt{M/M/1C}}$ differ, and this difference becomes more pronounced as k increases; this is again anticipated by Thm. 1, as the stochastic domination becomes looser for larger k. Nevertheless, the *solutions* $\boldsymbol{z}_{\mathtt{M/M/1C}}$ *and* $\boldsymbol{z}_{\mathtt{M/M/\infty}}$ *exhibit almost identical behavior w.r.t all four objectives*. For example, $C_{\mathtt{M/M/1C}}(\boldsymbol{z}_{\mathtt{M/M/\infty}}) \approx C_{\mathtt{M/M/1C}}(\boldsymbol{z}_{\mathtt{M/M/1C}}) \approx$

$\bar{C}_{\mathtt{M/M/1c}}(\boldsymbol{z}_{\mathtt{M/M/1c}}) \approx \bar{C}_{\mathtt{M/M/1c}}(\boldsymbol{z}_{\mathtt{M/M/\infty}})$. This means that, even though the two objectives are not the same, the quality of the solutions that they produce is quite similar, indicating a "robustness" in the choice of approximation (via Problem (11) or (13)). We also observe this in Table III, where these numbers are shown for quadratic objectives across topologies.

VII. Conclusion

We model a cache network as a system of counting queues in which identical packets merge when collocated. We propose an offline algorithm; even though item placements and service rate assignments should not be presumed as static, as they depend on the demand as solutions to Problem (14), fully adaptive algorithms would be interesting to study. Though such a setting harder to analyze, intuition gained in our setting may be applicable in this contexts too, via, e.g., the distributed, adaptive algorithm employed by Ioannidis and Yeh [12]. Exploring this is an interesting future direction.

In our system, responses merge; a more natural approach would be to merge requests. This occurs in several real-life networks, such as ICN/CCN [2]–[5]. When requests merge, they form $\mathtt{M/G/1c}$ queues: service time in such queues is not Markovian, and depends on caching and service rate assignments. Though related to the system we consider here, such a network is harder to model and analyze, and its study is still an open problem.

Acknowledgment

The authors gratefully acknowledge support from National Science Foundation grants NeTS-1718355 and CCF-1750539.

References

[1] Y. Li and S. Ioannidis, "Universally stable cache networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.

[3] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 181–192, 2007.

[4] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of information (netinf)–an information-centric networking architecture," *Computer Communications*, vol. 36, no. 7, pp. 721–735, 2013.

[5] E. Yeh, T. Ho, Y. Cui, M. Burd, R. Liu, and D. Leong, "Vip: A framework for joint dynamic forwarding and caching in named data networks," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, 2014, pp. 117–126.

[6] D. A. Farber, R. E. Greer, A. D. Swart, and J. A. Balter, "Internet content delivery network," Nov. 25 2003, uS Patent 6,654,807.

[7] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 374–385.

[8] A. Anand, V. Sekar, and A. Akella, "Smartre: an architecture for coordinated network-wide redundancy elimination," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4. ACM, 2009, pp. 87–98.

[9] X. Zhang, J. Liu, B. Li, and Y.-S. Yum, "Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3. IEEE, 2005, pp. 2102–2111.

[10] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4. ACM, 2002, pp. 177–190.

[11] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[12] S. Ioannidis and E. Yeh, "Adaptive caching networks with optimality guarantees," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 44, no. 1. ACM, 2016, pp. 113–124.

[13] S. E. Hajri and M. Assaad, "Energy efficiency in cache-enabled small cell networks with adaptive user clustering," *IEEE Transactions on Wireless Communications*, vol. 17, no. 2, pp. 955–968, 2017.

[14] S. Ioannidis and E. Yeh, "Jointly optimal routing and caching for arbitrary network topologies," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1258–1275, 2018.

[15] M. Mahdian, A. Moharrer, S. Ioannidis, and E. Yeh, "Kelly cache networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 217–225.

[16] F. P. Kelly, *Reversibility and stochastic networks*. Cambridge University Press, 2011.

[17] B. Abolhassani, J. Tadrous, and A. Eryilmaz, "Wireless multicasting for content distribution: Stability and delay gain analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1–9.

[18] A. Bian, K. Levy, A. Krause, and J. M. Buhmann, "Continuous DR-submodular maximization: Structure and algorithms," in *Advances in Neural Information Processing Systems*, 2017, pp. 486–496.

[19] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," in *Proceedings of the 6th International Conference*. ACM, 2010, p. 4.

[20] I. Baev, R. Rajaraman, and C. Swamy, "Approximation algorithms for data placement problems," *SIAM Journal on Computing*, vol. 38, no. 4, pp. 1411–1429, 2008.

[21] Y. Bartal, A. Fiat, and Y. Rabani, "Competitive algorithms for distributed data management," *Journal of Computer and System Sciences*, vol. 51, no. 3, pp. 341–358, 1995.

[22] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko, "Tight approximation algorithms for maximum general assignment problems," in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. Society for Industrial and Applied Mathematics, 2006, pp. 611–620.

[23] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. Citeseer, 2010, pp. 1–9.

[24] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal routing and content caching in heterogeneous networks," in *IEEE INFOCOM 2015-IEEE Conference on Computer Communications*. IEEE, 2015, pp. 936–944.

[25] S. Shukla and A. A. Abouzeid, "Proactive retention aware caching," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[26] K. Poularakis and L. Tassiulas, "Code, cache and deliver on the move: A novel caching paradigm in hyper-dense small-cell networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 3, pp. 675–687, 2016.

[27] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas, "Distributed caching algorithms in the realm of layered video streaming," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 757–770, 2018.

[28] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in mec networks with storage, computation, and communication constraints," *IEEE/ACM Transactions on Networking*, 2020.

[29] J. Li, T. K. Phan, W. K. Chai, D. Tuncer, G. Pavlou, D. Griffin, and M. Rio, "DR-cache: Distributed resilient caching with latency guarantees," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 441–449.

[30] B. Liu, K. Poularakis, L. Tassiulas, and T. Jiang, "Joint caching and routing in congestible networks of arbitrary topology," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 105–10 118, 2019.

[31] Z. Yang, D. Jia, S. Ioannidis, N. Mi, and B. Sheng, "Intermediate data caching optimization for multi-stage and parallel big data frameworks," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 2018, pp. 277–284.

[32] K. Poularakis and L. Tassiulas, "On the complexity of optimal content placement in hierarchical caching networks," *IEEE Transactions on Communications*, vol. 64, no. 5, pp. 2092–2103, 2016.
[33] G. Domingues, E. d. S. e Silva, R. M. Leao, D. S. Menasche, and D. Towsley, "Enabling opportunistic search and placement in cache networks," *Computer Networks*, vol. 119, pp. 17–34, 2017.
[34] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.
[35] F. Zafari, J. Li, K. K. Leung, D. Towsley, and A. Swami, "Optimal energy tradeoff among communication, computation and caching with qoi-guarantee," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.
[36] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1305–1314, 2002.
[37] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *2012 24th International Teletraffic Congress (ITC 24)*. IEEE, 2012, pp. 1–8.
[38] V. Martina, M. Garetto, and E. Leonardi, "A unified approach to the performance analysis of caching systems," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 2040–2048.
[39] G. Bianchi, A. Detti, A. Caponi, and N. Blefari Melazzi, "Check before storing: What is the performance price of content integrity verification in LRU caching?" *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 59–67, 2013.
[40] N. C. Fofack, P. Nain, G. Neglia, and D. Towsley, "Analysis of TTL-based cache networks," in *6th International ICST Conference on Performance Evaluation Methodologies and Tools*. IEEE, 2012, pp. 1–10.
[41] D. S. Berger, P. Gland, S. Singla, and F. Ciucu, "Exact analysis of TTL cache networks," *Performance Evaluation*, vol. 79, pp. 2–23, 2014.
[42] G. Domingues, R. M. Leao, D. S. Menasche *et al.*, "Flexible content placement in cache networks using reinforced counters," *arXiv preprint arXiv:1501.03446*, 2015.
[43] J. R. Jackson, "Networks of waiting lines," *Operations research*, vol. 5, no. 4, pp. 518–521, 1957.
[44] C.-S. Chang, *Performance guarantees in communication networks*. Springer Science & Business Media, 2012.
[45] F. P. Kelly, "Loss networks," *The annals of applied probability*, pp. 319–378, 1991.
[46] Z. Zhang and A. S. Acampora, "A heuristic wavelength assignment algorithm for multihop wdm networks with wavelength routing and wavelength re-use," *IEEE/ACM Transactions on networking*, vol. 3, no. 3, pp. 281–288, 1995.
[47] S. Keshav, *An engineering approach to computer networking: ATM networks, the Internet, and the telephone network*. Addison-Wesley Reading, 1997, vol. 116.
[48] Y. Fang and I. Chlamtac, "Teletraffic analysis and mobility modeling of pcs networks," *IEEE Transactions on Communications*, vol. 47, no. 7, pp. 1062–1072, 1999.
[49] C. Courcoubetis, *Pricing Communication Networks Economics, Technology and Modelling*. Wiley Online Library, 2003.
[50] A. Krause and D. Golovin, "Submodular function maximization." 2014.
[51] D. P. Bertsekas, *Nonlinear programming*. Athena scientific Belmont, 1999.
[52] Y. Filmus and J. Ward, "Monotone submodular maximization over a matroid via non-oblivious local search," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 514–542, 2014.
[53] M. Sviridenko, J. Vondrák, and J. Ward, "Optimal approximation for submodular and supermodular optimization with bounded curvature," *Mathematics of Operations Research*, vol. 42, no. 4, pp. 1197–1218, 2017.
[54] A. Bian, B. Mirzasoleiman, J. M. Buhmann, and A. Krause, "Guaranteed non-convex optimization: Submodular maximization over continuous domains," *Proceedings of Machine Learning Research*, vol. 54, pp. 111–120, 2017.
[55] A. A. Ageev and M. I. Sviridenko, "Pipage rounding: A new method of constructing algorithms with proven performance guarantee," *Journal of Combinatorial Optimization*, vol. 8, no. 3, pp. 307–328, 2004.
[56] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data networks*. Prentice-Hall International New Jersey, 1992, vol. 2.
[57] R. G. Gallager, *Stochastic processes: theory for applications*. Cambridge University Press, 2013.
[58] P. G. Harrison and N. M. Patel, *Performance modelling of communication networks and computer architectures (International Computer S*. Addison-Wesley Longman Publishing Co., Inc., 1992.
[59] G. Bolch, S. Greiner, H. De Meer, and K. S. Trivedi, *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.
[60] M. R. Hestenes, "Optimization theory: the finite dimensional case," *New York*, 1975.
[61] M. H. Stone, "Applications of the theory of boolean rings to general topology," *Transactions of the American Mathematical Society*, vol. 41, no. 3, pp. 375–481, 1937.
[62] C. Chekuri, J. Vondrak, and R. Zenklusen, "Dependent randomized rounding via exchange properties of combinatorial structures," in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 2010, pp. 575–584.
[63] S. Ioannidis and E. Yeh, "Adaptive caching networks with optimality guarantees," *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 737–750, 2018.
[64] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," *Relatório técnico, Telecom ParisTech*, pp. 1–6, 2011.
[65] R. W. Wolff, "Poisson arrivals see time averages," *Operations Research*, vol. 30, no. 2, pp. 223–231, 1982.
[66] J. Riordan, "Moment recurrence relations for binomial, poisson and hypergeometric frequency distributions," *The Annals of Mathematical Statistics*, vol. 8, no. 2, pp. 103–111, 1937.
[67] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

Yuanyuan Li received the B.E. degree from School of Electronic and Information Engineering, South China University of Technology, China and M.S. degree from Department of Computer Science and Engineering, Shanghai Jiao Tong University. She is now a Ph.D. student in Computer Engineering, Northeastern University, Boston, USA., under the supervision of Prof. Stratis Ioannidis. Her research interests include networking, optimization and machine learning.

Stratis Ioannidis is an Associate Professor in the Electrical and Computer Engineering department at Northeastern University, in Boston, MA, where he also holds a courtesy appointment with the Khoury College of Computer Sciences. He received his B.Sc. (2002) in Electrical and Computer Engineering from the National Technical University of Athens, Greece, and his M.Sc. (2004) and Ph.D. (2009) in Computer Science from the University of Toronto, Canada. Prior to joining Northeastern, he was a research scientist at the Technicolor research centers in Paris, France, and Palo Alto, CA, as well as at Yahoo Labs in Sunnyvale, CA. He is the recipient of an NSF CAREER award, a Google Faculty Research Award, a Facebook Research Award, and Best Paper Awards at the 2017 ACM Conference on Information-centric Networking (ICN) and the 2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN).

APPENDIX

A. Proof of Lemma 1

By the balance/equilibrium equations of this Markov process (see Eq. (1.3), p. 3 in [16]), the steady state distribution $\boldsymbol{\pi} = [\pi_n]_{n=0}^{\infty}$ satisfies:

$$\begin{cases} \pi_0 \lambda_e^r = \sum_{n>0} \pi_n \mu_e^r, \\ \pi_{n-1} \lambda_e^r = \pi_n (\lambda_e^r + \mu_e^r), \quad n \geq 2, \\ \sum_n \pi_n = 1. \end{cases}$$

We can show Eq. (3) is the solution by induction on n. □

B. Proof of Theorem 1

We first state two auxiliary lemmas. For convenience, we use ρ representing ρ_e^r in this proof.

Lemma 5. *The k-th moment $m^k_{\mathtt{M/M/1C}}(\rho)$ and $m^k_{\mathtt{M/M/\infty}}(\rho)$ can be obtained from recurrence relations:*

$$m^{k+1}_{\mathtt{M/M/1C}}(\rho) = \rho m^k_{\mathtt{M/M/1C}}(\rho) + \rho(\rho+1)\frac{\mathrm{d}m^k_{\mathtt{M/M/1C}}(\rho)}{\mathrm{d}\rho},$$

$$m^{k+1}_{\mathtt{M/M/\infty}}(\rho) = \rho m^k_{\mathtt{M/M/\infty}}(\rho) + \rho\frac{\mathrm{d}m^k_{\mathtt{M/M/\infty}}(\rho)}{\mathrm{d}\rho}.$$

Proof. The steady state distribution of $\mathtt{M/M/\infty}$ queue is a Poisson distribution with parameter ρ, and by eq. (3.8), pp. 106 in [66], we get the recurrence relation for $m^k_{\mathtt{M/M/\infty}}$. Similarly, the steady state distribution of $\mathtt{M/M/1C}$ queue is a geometric distribution with parameter $p = \frac{\rho}{\rho+1}$. Taking the derivative of $m^k_{\mathtt{M/M/1C}}$ w.r.t. p, $\frac{\mathrm{d}m^k_{\mathtt{M/M/1C}}}{\mathrm{d}p} = -\frac{1}{1-p}m^{k+1}_{\mathtt{M/M/1C}} + \frac{1}{p}m^k_{\mathtt{M/M/1C}}$. Using the chain rule (see Eq. (A.6), p. 642 in [67]) and rearranging the terms, we get the recurrence relations for $m^k_{\mathtt{M/M/1C}}$ w.r.t. ρ. □

Lemma 6. *Both $m^k_{\mathtt{M/M/1C}}(\rho)$ and $m^k_{\mathtt{M/M/\infty}}(\rho)$ are polynomials, i.e., $m^k_{\mathtt{M/M/1C}}(\rho) = \sum_{i=1}^k \beta_i^k \rho^i$, $m^k_{\mathtt{M/M/\infty}}(\rho) = \sum_{i=1}^k \alpha_i^k \rho^i$, where $\beta_i^k, \alpha_i^k > 0$, and $\frac{\beta_i^k}{\alpha_i^k} = i!$.*

Proof. We prove this by induction. For $k = 1$, this follows from (5) (also Lem. 5 for $k = 0$). Suppose it holds for $k = \ell$, i.e., $m^\ell_{\mathtt{M/M/1C}}(\rho) = \sum_{i=1}^\ell i!\alpha_i^\ell \rho^i$, and $m^\ell_{\mathtt{M/M/\infty}}(\rho) = \sum_{i=1}^\ell \alpha_i^\ell \rho^i$. Then, when $k = \ell + 1$, by Lemma 5:

$$m^{\ell+1}_{\mathtt{M/M/1C}}(\rho) = \alpha_1^\ell \rho + \sum_{i=2}^{\ell} i!(\alpha_{i-1}^\ell + i\alpha_i^\ell)\rho^i + (\ell+1)!\alpha_\ell^\ell \rho^{\ell+1}$$

$$m^{\ell+1}_{\mathtt{M/M/\infty}}(\rho) = \alpha_1^\ell \rho + \sum_{i=2}^{\ell} (\alpha_{i-1}^\ell + i\alpha_i^\ell)\rho^i + \alpha_\ell^\ell \rho^{\ell+1}$$

Comparing terms, we have $\frac{\beta_i^{\ell+1}}{\alpha_i^{\ell+1}} = i!$. □

To prove Thm. 1, observe that by Lemma 6, $\frac{m^k_{\mathtt{M/M/\infty}}(\rho)}{m^k_{\mathtt{M/M/1C}}(\rho)} = \frac{\sum_{i=1}^k \alpha_i^k \rho^i}{\sum_{i=1}^k i!\alpha_i^k \rho^i}$. Hence,

$$\frac{\sum_{i=1}^k \alpha_i^k \rho^i}{\sum_{i=1}^k k!\alpha_i^k \rho^i} \leq \frac{m^k_{\mathtt{M/M/\infty}}(\rho)}{m^k_{\mathtt{M/M/1C}}(\rho)} \leq \frac{\sum_{i=1}^k \alpha_i^k \rho^i}{\sum_{i=1}^k \alpha_i^k \rho^i},$$

which implies $\frac{1}{k!} \leq \frac{m^k_{\mathtt{M/M/\infty}}(\rho)}{m^k_{\mathtt{M/M/1C}}(\rho)} \leq 1$. □

C. Proof of Theorem 2

(Sketch) Observe that $\textsc{MinCost}_{\mathtt{M/M/\infty}}$ and $\textsc{MinCost}_{\mathtt{M/M/1C}}$ are identical when $c_e^r(n_e^r) = n_e^r$, i.e., when the cost is the queue size: by (5), both expected costs are equal to ρ_e^r. We reduce the (NP-hard) fixed routing cost problem by Ioannidis and Yeh [12] to this problem. To do so, if an edge has cost w_{uv}, we set $\mu_{uv} = |\mathcal{R}_{uv}|/w_{uv}$ and $\epsilon = 1/w_{uv}$. Then the service rate μ_{uv}^r at each queue on edge (u, v) is exactly $1/w_{uv}$ (i.e., $\mathcal{D}_\mu$ is a singleton), and both $\textsc{MinCost}_{\mathtt{M/M/\infty}}$ and $\textsc{MinCost}_{\mathtt{M/M/1C}}$ coincide with the fixed cost routing problem. □

D. Proof of Lemma 3

By Eq. (12), the expected costs of $\mathtt{M/M/1C}$ queues are:

$$\mathbb{E}_{\mathtt{M/M/1C}}[c_e^r(n_e^r)] = c_e^r(0) + \sum_{n=0}^{\infty}(c_e^r(n+1) - c_e^r(n))(\frac{\rho_e^r}{\rho_e^r+1})^{n+1}.$$

The corresponding first derivative w.r.t. ρ_e^r is,

$$\frac{\mathrm{d}\mathbb{E}_{\mathtt{M/M/1C}}[c_e^r(n_e^r)]}{\mathrm{d}\rho_e^r} = \sum_{n=0}^{\infty}(c_e^r(n+1) - c_e^r(n)) \cdot \frac{(n+1)(\rho_e^r)^n}{(\rho_e^r+1)^{n+2}} \geq 0.$$

Moreover, $\frac{\mathrm{d}^2\mathbb{E}_{\mathtt{M/M/1C}}[c_e^r(n_e^r)]}{\mathrm{d}(\rho_e^r)^2} = \sum_{n=0}^{\infty}\Delta_n$, where $\Delta_n = (c_e^r(n+1) - c_e^r(n))(n+1)\frac{n(\rho_e^r)^{n-1} - 2(\rho_e^r)^n}{(\rho_e^r+1)^{n+3}}$. By Assumption 1, let $n_0 \equiv \lfloor 2\rho_e^r \rfloor$ we have that $\Delta_n \geq (c_e^r(n_0+1) - c_e^r(n_0))(n+1)\frac{n(\rho_e^r)^{n-1} - 2(\rho_e^r)^n}{(\rho_e^r+1)^{n+3}}$, for all $n \in \mathbb{N}$. Hence,

$$\frac{\mathrm{d}^2\mathbb{E}_{\mathtt{M/M/1C}}[c_e^r(n_e^r)]}{\mathrm{d}(\rho_e^r)^2} \geq (c_e^r(n_0+1) - c_e^r(n_0)) \cdot \sum_{n=0}^{\infty}\Big[(n+1)n\frac{(\rho_e^r)^{n-1}}{(\rho_e^r+1)^{n+3}} - 2(n+1)\frac{(\rho_e^r)^n}{(\rho_e^r+1)^{n+3}}\Big] = 0.$$

Thus, $\mathbb{E}_{\mathtt{M/M/1C}}[c_e^r(n_e^r)]$ is non-decreasing and convex w.r.t. ρ_e^r. Similarly, by Eq. (10):

$$\mathbb{E}_{\mathtt{M/M/\infty}}[c_e^r(n_e^r)] = c_e^r(0) + \sum_{n=0}^{\infty}(c_e^r(n+1) - c_e^r(n))e^{-\rho_e^r}\sum_{l=n+1}^{\infty}\frac{(\rho_e^r)^l}{l!}.$$

The corresponding first derivative w.r.t. ρ_e^r is,

$$\frac{\mathrm{d}\mathbb{E}_{\mathtt{M/M/\infty}}[c_e^r(n_e^r)]}{\mathrm{d}\rho_e^r} = \sum_{n=0}^{\infty}(c_e^r(n+1) - c_e^r(n)) \cdot e^{-\rho_e^r}\frac{(\rho_e^r)^n}{n!} \geq 0.$$

Moreover, $\frac{\mathrm{d}^2\mathbb{E}_{\mathtt{M/M/\infty}}[c_e^r(n_e^r)]}{\mathrm{d}(\rho_e^r)^2} = \sum_{n=0}^{\infty}\Delta_n$, where $\Delta_n = (c_e^r(n+1) - c_e^r(n))(-e^{-\rho_e^r}\frac{(\rho_e^r)^n}{n!} + e^{-\rho_e^r}\frac{n(\rho_e^r)^{n-1}}{n!})$. By Assumption 1, let $n_0 \equiv \lfloor \rho_e^r \rfloor$ we have that $\Delta_n \geq (c_e^r(n_0+1) - c_e^r(n_0))(-e^{-\rho_e^r}\frac{(\rho_e^r)^n}{n!} + e^{-\rho_e^r}\frac{n(\rho_e^r)^{n-1}}{n!})$, for all $n \in \mathbb{N}$. Hence,

$$\frac{d^2\mathbb{E}_{\mathtt{M/M/\infty}}[c_e^r(n_e^r)]}{d(\rho_e^r)^2} \geq (c_e^r(n_0+1) - c_e^r(n_0)) \cdot (\sum_{n=0}^{\infty} -e^{-\rho_e^r}\frac{(\rho_e^r)^n}{n!} + \sum_{n=1}^{\infty} e^{-\rho_e^r}\frac{(\rho_e^r)^{n-1}}{(n-1)!}) = 0.$$

Thus, $\mathbb{E}_{\mathtt{M/M/\infty}}[c_e^r(n_e^r)]$ is non-decreasing and convex w.r.t. ρ_e^r. □

E. Proof of Lemma 4

For convenience, we replace subscripts e and superscripts r by subscript $i \in E \times \sum_e \mathcal{R}_e$. (a) By Lemma 3, $C_i(\rho_i)$ is non-decreasing convex w.r.t. ρ_i. Also, $\rho_i = \frac{\lambda_i}{\mu_i}$ is decreasing convex w.r.t. μ_i. Hence, by Eq. (3.10), p.84 in [67], we have that $\frac{\partial C_i(\boldsymbol{x},\mu_i)}{\partial \mu_i} \leq 0$, $\frac{\partial^2 C_i(\boldsymbol{x},\mu_i)}{\partial \mu_i^2} \geq 0$. Hence, the first derivative of $F(S,\boldsymbol{\mu})$ w.r.t. μ_i is:

$$\frac{\partial F(S,\boldsymbol{\mu})}{\partial \mu_i} = -\frac{\partial C(\boldsymbol{x},\boldsymbol{\mu})}{\partial \mu_i} \geq 0,$$

and the second derivative w.r.t. μ_i, μ_j is:

$$\frac{\partial^2 F(S,\boldsymbol{\mu})}{\partial \mu_i \partial \mu_j} = -\frac{\partial^2 C(\boldsymbol{x},\boldsymbol{\mu})}{\partial \mu_i \partial \mu_j} = \begin{cases} 0 & i \neq j \\ -\frac{\partial^2 C_i(\boldsymbol{x},\mu_i)}{\partial \mu_i^2} & i = j \end{cases} \leq 0,$$

so $\nabla_{\boldsymbol{\mu}} F(S,\boldsymbol{\mu}) \geq \mathbf{0}$ and $\nabla^2_{\boldsymbol{\mu}} F(S,\boldsymbol{\mu}) \leq 0$. Thus, $F(S,\boldsymbol{\mu})$ is non-decreasing and concave on $\boldsymbol{\mu}$. (b) We know that C_i is non-decreasing and convex w.r.t. ρ_i. By Theorem 1 and Corollary 1 in [15], $F(S,\boldsymbol{\mu})$ is non-decreasing and submodular on set S. □

F. Proof of Theorem 4

Proof. We begin by stating a lemma on the feasibility of the the output of Alg. 1.

Lemma 7. *The output of Alg. 1, $\boldsymbol{z}_K$, and the intermediate result $\boldsymbol{z}_k$ always belong to feasible region $\mathcal{D}_z$, i.e., $\boldsymbol{z}_k \in \mathcal{D}_z$, for all $k \in \{1,\ldots,K\}$.*

Proof. By construction (see Alg. 1), we have that $\boldsymbol{z}_K = \sum_{k=1}^K \gamma_k \boldsymbol{m}_k$, where $\gamma_k > 0$ and $\sum_{k=1}^K \gamma_k = 1$. Moreover, by Eq. (19a), $\boldsymbol{m}_k \in \mathcal{D}_z$ for all $k \in \{1,\ldots,K\}$. Hence, $\boldsymbol{z}_K$ is a convex combination of points in $\mathcal{D}_z$; since $\mathcal{D}_z$ is a convex set; $z_K \in \mathcal{D}_z$. By Eq. (19), $\mathbf{0} \leq \boldsymbol{z}_k \leq \boldsymbol{z}_K$, and $\mathcal{D}_z$ is a down-closed convex set, thus z_k always belongs to $\mathcal{D}_z$ for all $k \in \{1,\ldots,K\}$. □

Then, we state a lemma on the quality of the output of Alg. 1, assuming that the estimate $\widehat{\nabla G}$ is produced via the sampling method outlined in Sec. IV-D.

Lemma 8. *For a fixed number of iterations K which is large enough, and constant stepsize $\gamma_k = \gamma = K^{-1}$, Alg. 1 provides the following approximation guarantee with high probability:*

$$G(\boldsymbol{z}_K) \geq (1 - \frac{1}{e}) G(\boldsymbol{z}^*), \tag{30}$$

where $\boldsymbol{z}_K$ is output of Alg. 1, $\boldsymbol{z}^*$ is an optimal solution to Problem (20).

Proof. Function G is DR-submodular, as shown in Lemma 3, domain $\tilde{\mathcal{D}} \times \mathcal{D}_\mu$ is a down-closed convex set, and Lipschitz parameter of ∇G is $2C(\mathbf{0},\boldsymbol{\epsilon})$ because $||\nabla^2 G(\boldsymbol{z})||$ is bounded by $2C(\mathbf{0},\boldsymbol{\epsilon})$ according to (18). With above conditions, Corollary 1 by Bian et al. [54] states: $G(\boldsymbol{z}_K) \geq (1-\frac{1}{e})G(\boldsymbol{z}^*) - \frac{L}{2K}$, where $L = 2C(\mathbf{0},\boldsymbol{\epsilon})$ is a finite constant. Calinescu et al. [34] show that for large enough K, the offset $\frac{L}{2K}$ can be omitted and (30) still holds with high probability. The term "with high probability", due to sample-based estimation of ∇G, and means probability at least $1 - 1/|V||C|$. □

To conclude the proof of Theorem 4, we have:

$$\mathbb{E}[F(\boldsymbol{x}_K,\boldsymbol{\mu}_K)] \overset{\text{Eq. (17)}}{=} \mathbb{E}[G(\boldsymbol{x}_K,\boldsymbol{\mu}_K)] \overset{\text{Eq. (26)}}{\geq} G(\boldsymbol{y}_K,\boldsymbol{\mu}_K)$$
$$\overset{\text{Lem. 8}}{\geq} (1-\frac{1}{e})G(\boldsymbol{y}^*,\boldsymbol{\mu}^*) \geq (1-\frac{1}{e})F(\boldsymbol{x}^*,\boldsymbol{\mu}^*),$$

where $\boldsymbol{x}^*$ and $\boldsymbol{\mu}^*$ is an optimal solution to (15), $\boldsymbol{y}^*$ is an optimal solution to (20), $\boldsymbol{y}_K$ and $\boldsymbol{\mu}_K$ is the output of Frank-Wolfe variant algorithm, and $\boldsymbol{x}_K$ is the integer solution after rounding. The first equation holds because F and G are equal under integer arguments $\boldsymbol{x}_K$. The last inequality holds because (20) has a larger feasible region. □