Large-Scale Network Utility Maximization: Countering Exponential Growth with Exponentiated Gradients

Luigi Vigneri, Georgios Paschos, Panayotis Mertikopoulos

Large-Scale Network Utility Maximization: Countering Exponential Growth with Exponentiated Gradients

Luigi Vigneri, Georgios Paschos, and Panayotis Mertikopoulos

***Abstract*—Network utility maximization (NUM) is an iconic problem in network traffic management which is at the core of many current and emerging network design paradigms – and, in particular, software-defined networks (SDNs). Thus, given the exponential growth of modern-day networks (in both size and complexity), it is crucial to develop scalable algorithmic tools that are capable of providing efficient solutions in time which is *dimension-free*, i.e., independent – or nearly-independent – on the size of the system. To do so, we leverage a suite of modified gradient methods known as "mirror descent", and we derive a scalable and efficient algorithm for the NUM problem based on gradient exponentiation. We show that the convergence speed of the proposed algorithm only carries a logarithmic dependence on the size of the network, so it can be implemented reliably and efficiently in massively large networks where traditional gradient methods are prohibitively slow. These theoretical results are subsequently validated by extensive numerical simulations showing an improvement of several order of magnitudes over standard gradient methods in large-scale networks.**

I. Introduction

One of the most important design attributes of cloud-based networking is the ability to adjust the allocation of bandwidth and computing resources in real-time. This allows cloud-based networks to adapt to variable loads and demands "on the fly", leading in turn to dramatic performance gains in throughput, end-to-end connectivity, and reliability.

From a network control standpoint, a key obstacle to achieving these gains is that they require solving, rapidly and efficiently, complex optimization problems with several thousands – if not millions – of control variables. This is most readily seen in the reference problem of *network utility maximization* (NUM), a landmark framework pioneered by Kelly et al. [9] to ensure fairness and stability in communication networks through the maximization of a concave utility function subject to link capacity constraints. Owing to its flexibility, this framework has found widespread applications to wireless networks [16], sensor networks [7], caching systems [6], and many other fields (for a survey, see [18] and references therein). However, as such networks grow in size and complexity, it is crucial to develop *scalable* algorithms which could provide reasonable solutions to the NUM problem in real time.

This situation is exacerbated by the massive penetration of software-defined networks (SDNs) to the access and transport layers. Here, network controllers are required to centrally manage the allocation of bandwidth resources to an exponentially large number of sessions (often numbering in the millions for IP access network domains). As such, any NUM algorithm must be *dimension-free*, i.e., its execution runtime must be *independent* – or *nearly-independent* – of the size of the network.

L. Vigneri* is with IOTA Foundation, Berlin, Germany. G. Paschos is with Huawei Technologies, Boulogne Billancourt, France. P. Mertikopoulos is with Univ. Grenoble Alpes, CNRS, Inria, Grenoble, France.

*The work was done when the author was with Huawei Technologies.

This work was partially supported by the French National Research Agency (ANR) project ORACLESS, and the Huawei Flagship project ULTRON.

A. Our contributions

In this paper, we follow the classical formulation of the NUM problem as a concave program over a network with d concurrent sessions [18]. In contrast to application-driven formulations, this generalist approach provides us with a flexible theoretical springboard for the use of state-of-the-art optimization methods (which can then be ported to specific scenarios with minimal modifications). More concretely, motivated by recent breakthroughs in artificial intelligence and machine learning, we leverage an optimization technique known as "mirror descent" (or "ascent" in our case), and which was originally introduced by Nemirovski and Yudin [15] to solve image reconstruction problems with massively large data sets. In contrast to vanilla gradient methods, mirror ascent replaces standard (projected) gradient steps by a variable-geometry step which intrinsically accelerates the method in regions of the problem's feasible space where larger step-sizes might be more beneficial. In this way, depending on the geometry of the problem, mirror ascent methods often succeed in lifting the so-called "curse of dimensionality", and they provide convergence rates that vastly outperform unmodulated gradient methods.

By coupling mirror descent methodologies with a synthetic modification of the geometry of the problem at hand, we derive a fast and scalable interior-point method based on the exponentiation of network utility gradients, which we call exp-NUM (*exponentiated gradients algorithm for NUM*). In terms of performance gains, the proposed exp-NUM algorithm achieves an ε-optimal solution of the NUM problem in $\mathcal{O}(\log d/\varepsilon^2)$ iterations, i.e., in time which is $\mathcal{O}(\log d)$ relative to the size of the network – and hence, effectively *dimension-free*. As a result, thanks to this exponential acceleration factor, exp-NUM can be implemented reliably and efficiently in massively large networks where traditional first-order methods are prohibitively slow. This theoretical analysis is subsequently validated by an extensive suite of numerical simulations that exhibit exponential improvement over standard methods in large-scale random networks, and which clearly indicate the benefits of the proposed algorithm for large-scale network utility maximization.

Algorithm	Iteration Complexity	Function Class
Projected gradient ascent [5]	$\mathcal{O}(d/\varepsilon^2)$	Concave, Lipschitz continuous
Dual subgradient + primal averaging [10, 12]	$\mathcal{O}(d/\varepsilon^2)$	Concave, Lipschitz continuous
Backpressure [13]	$\mathcal{O}(d/\varepsilon^2)$	Concave, Lipschitz continuous
Virtual queues [21]	$\mathcal{O}(d/\varepsilon)$	Concave, Lipschitz continuous
ADMM [1, 8]	$\mathcal{O}(d/\varepsilon)$	Strongly concave, strongly smooth
Primal-dual [4]	$\mathcal{O}(\text{poly}(d)/\varepsilon)$	Strongly concave, Lipschitz continuous
Fast primal-dual [4]	$\mathcal{O}(\text{poly}(d)/\sqrt{\varepsilon})$	Concave, strongly smooth
exp-NUM [this paper]	$\boldsymbol{\mathcal{O}(\log d/\varepsilon^2)}$	**Concave, Lipschitz continuous**

TABLE I: Convergence speed of first-order methods for the NUM problem (second-order methods are omitted because of their much higher per-iteration complexity). Results pertaining to the exp-NUM algorithm are displayed in **bold**.

B. Relation to existing work

The performance of an iterative NUM algorithm can be characterized in terms of (*a*) the algorithm's *iteration complexity*, i.e., the number of iterations required to achieve a given error (typically measured in terms of the value of the problem's objective function); and (*b*) the algorithm's *per-iteration complexity*, i.e., the number of floating point operations required to perform an update. To the best of our knowledge, the fastest schemes available in terms of iteration complexity are based on Newton's method: if the NUM objective satisfies a self-concordance condition, the authors of [20] showed that an ε-optimal solution can be achieved in $\mathcal{O}(1/\sqrt{\varepsilon})$ iterations, improving in this way the $\mathcal{O}(1/\varepsilon)$ bound of [2]. On the other hand, Newton's method requires the computation of a Hessian matrix at each queried point and the solution of a $d \times d$ linear-system per update; as a result, the *per-iteration* complexity of Newton-based schemes is $\mathcal{O}(d^3)$, so it becomes prohibitive when d is of the order of a few thousands. The primal-dual algorithm of [4] also achieves an iteration complexity of $\mathcal{O}(\text{poly}(d)/\varepsilon)$ (or $\mathcal{O}(\text{poly}(d)/\sqrt{\varepsilon})$ when coupled with Nesterov acceleration), but assumes strong concavity of the NUM objective, which subsumes Hessian invertibility and does not hold in networks with several overlapping origin-destination (O/D) paths.

In terms of *per-iteration* complexity, the fastest known methods rely on first-order information, and, for this reason, they are the benchmark reference solutions for large-scale NUM problems. Within this class, (projected) gradient ascent [5, 18] and dual (Lagrangian) ascent [10] require $\mathcal{O}(d/\varepsilon^2)$ iterations to achieve an ε-optimal solution. An appealing alternative is the distributed backpressure algorithm, where nodes route packets according to queue backlog differences in order to balance the backlogs in the network. Backpressure can be understood as a stochastic dual subgradient algorithm [22] with the queue backlogs playing the role of stochastic Lagrangian multipliers, thus leading to an $\mathcal{O}(d/\varepsilon^2)$ iteration complexity.

This convergence speed can be improved to $\mathcal{O}(d/\varepsilon)$ by using an alternating direction method of multipliers (ADMM) approach [1, 8]. ADMM updates are relatively cheap, but the method requires additional assumptions on the NUM objective (viz. strong concavity and strong smoothness) which may fail to hold in general. These assumptions were recently lifted by the authors of [21] who provide a virtual queues algorithm with an $\mathcal{O}(d/\varepsilon)$ convergence speed; however, the proposed algorithm employs a complicated prox-mapping which makes its per-iteration complexity prohibitive for large networks.

By comparison, the iteration complexity of the exp-NUM algorithm is $\mathcal{O}(\log d/\varepsilon^2)$, i.e., it converges to an ε-optimal solution in logarithmic time relative to the problem's dimension (in our case, the number of sessions). At the same time, the algorithm's per-iteration complexity is the cheapest among all methods, so this acceleration does not come at the expense of costlier updates. As such, of the algorithms discussed above, exp-NUM is the only one whose complexity provably scales with the size of the network. In practice, this means a runtime reduction of $\tilde{\Theta}(d)$, amounting to *an improvement of* 4 *orders of magnitude* in large-scale networks with $\mathcal{O}(10^5)$ sessions (and the gains only scale upwards as d becomes larger); We find this property of exp-NUM particularly appealing as it provides a very promising platform for the systematic design of scalable resource allocation protocols in massively large networks.

This dramatic runtime reduction by a factor of $\tilde{\mathcal{O}}(d)$ should not be interpreted as a blanket result that applies to *all* convex programs: it is a by-product of the simplicial structure of the convex program at hand which allows the use of a carefully crafted, geometry-aware regularization process. Thus, in addition to the novel application of mirror ascent methodologies, an important part of our contribution is the principled modification of the problem's geometry so as to be able to take full advantage of this tailor-made regularization process. We find this approach particularly appealing as it provides a springboard for the application of similar techniques to a wide range of resource allocation problems in the area of networking.

For convenience, we summarize the complexity results for first-order methods in Table I. For the purposes of validation, we also perform in Section V an extensive suite of simulations that shows the scalability gains of exp-NUM over traditional first-order methods in large-scale random networks.

II. System model and problem formulation

In this section, we describe the basic setup of the NUM problem; for our notational conventions, see Table II.

A. Network utility maximization

The core ingredients of the network utility maximization (NUM) problem are as follows:

- *Network model:* The underlying network is modeled as a directed multi-graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ where $\mathcal{N}$ is the set of the network's *nodes* and $\mathcal{L}$ is its set of *links*.

- *Sessions:* Each O/D pair in the network may carry a communication *session*. We will write $\mathcal{S} = \{1, \dots, d\}$ for the set of such sessions; we will also assume for now that each session $s \in \mathcal{S}$ is associated with a single O/D path, and transmits over said path at rate $x_s \geq 0$ (see Section IV-C for the multi-path case).
- *Utility function:* We posit that, if the s-th session transmits at rate x_s, its utility is given by $U_s(x_s)$, where $U_s \colon \mathbb{R}_+ \to \mathbb{R}$ is assumed differentiable and concave.[1]
- *Capacity constraints:* Each link $\ell \in \mathcal{L}$ is assumed to have a finite capacity c_ℓ.

In view of the above, the load on link $\ell \in \mathcal{L}$ induced by a transmission rate profile $\mathbf{x} = (x_s)_{s \in \mathcal{S}}$ is given by

$$x_\ell = \sum_{s \ni \ell} x_s, \tag{1}$$

where the notation "$s \ni \ell$" signifies that link ℓ belongs to the path carrying the traffic of the s-th session. The *network utility maximization* (NUM) problem may then be formulated as

$$\begin{aligned} \text{maximize} \quad & V(\mathbf{x}) \equiv \sum_{s \in \mathcal{S}} U_s(x_s) \\ \text{subject to} \quad & x_s \geq 0 \quad \text{for all } s \in \mathcal{S} \\ & x_\ell \leq c_\ell \quad \text{for all } \ell \in \mathcal{L} \end{aligned} \tag{NUM$_0$}$$

i.e., as the problem of maximizing the network's aggregate utility while respecting each link's individual capacity.

Concerning the session utility functions U_s, the standard choice in the literature is the α-fair model [9, 17, 18]

$$U^\alpha(z) = \begin{cases} \frac{z^{1-\alpha}}{1-\alpha} & \text{if } \alpha \geq 0, \alpha \neq 1, \\ \log z & \text{if } \alpha = 1. \end{cases} \tag{2}$$

This utility model is sufficiently flexible to account for several different optimality and fairness criteria (thus ensuring objective diversification in a network setting). In particular, for $\alpha = 0$, (NUM$_0$) boils down to sum-rate maximization; for $\alpha = 1$, solving (NUM$_0$) leads to proportionally fair rate allocations; for $\alpha = 2$, (NUM$_0$) amounts to potential delay fairness (also linked to TCP congestion control in the Internet [11]); finally, as $\alpha \to \infty$, we get the standard max-min fairness formulation.

To increase the flexibility of our model, we further posit that each session is assigned a weight $w_s > 0$ measuring its importance for the network at large (for instance, a session carrying emergency medical information should carry more weight than a routine music streaming session). With all this in mind, our core utility model will be of the form $U_s(x_s) = w_s U^\alpha(x_s)$ for some fixed α (determined by the network administrator depending on the precise criterion to be optimized).

B. Penalty-based formulation

In large-scale networks, the number of sessions quickly grows massively large, rendering the rigid link capacity constraints in (NUM$_0$) effectively impossible to handle as stated. For this reason, it is very common in the literature [9, 18] to consider instead a penalty-based formulation where the objective of (NUM$_0$) is written as

$$V(\mathbf{x}) = \sum_{s \in \mathcal{S}} U_s(x_s) - \mu \sum_{\ell \in \mathcal{L}} g_\ell(x_\ell), \tag{3}$$

where $\mu > 0$ is a *rigidity* parameter controlling the impact of the *capacity penalty functions* $g_\ell \colon \mathbb{R}_+ \to \mathbb{R}$, $\ell \in \mathcal{L}$. Specifically, instead of maximizing the sum of utilities constrained by link capacities, the network administrator now maximizes the aggregate network utility minus a (potentially very steep) cost for overshooting the capacity of a given link. The benefit of this formulation is that the problem's constraints are now uncoupled across its control variables (x_s, $s \in \mathcal{S}$); on the other hand, this coupling now appears in the problem's objective (which was uncoupled in the original formulation), so the problem's interaction structure remains the same.

Regarding the choice of penalty functions g_ℓ, the indicator function $g_\ell(z) = \infty$ if $z > c_\ell$ and $g_\ell(z) = 0$ if $z \leq c_\ell$ exactly recovers (NUM$_0$) as stated above – but also exhibits the same drawbacks in terms of tractability. To reap the benefits of rigidity relaxation, it is instead more convenient to assume that each g_ℓ is an increasing convex function, a typical choice being

$$g_\ell(x_\ell) = [x_\ell - c_\ell]_+^q, \tag{4}$$

i.e., incurring a cost that grows as a q-th power (typically $q = 1$ or $q = 2$) when the load $x_\ell = \sum_{s \ni \ell} x_s$ on link ℓ exceeds the link's capacity.

Needless to say, this framework offers considerably more leeway and flexibility to the network adminstrator relative to the rigid formulation (NUM$_0$). First, from standard epiconvergence arguments [3], the solutions of (NUM$_0$) can be recovered *exactly* from this relaxed formulation if $g'_\ell(c_\ell) > 0$ and μ is large enough.

TABLE II: Notation used in the paper.

Network model	
$\mathcal{N}$	Set of nodes
$\mathcal{L}$	Set of links
$\mathcal{S}$	Set of sessions
d	Number of sessions (*dimensionality*)
c_ℓ	Capacity of link $\ell \in \mathcal{L}$
C	Global cap
Network utility maximization	
$\mathbf{x}$	Session rates (*control variable*)
U_s	Utility function of session $s \in \mathcal{S}$
g_ℓ	Penalty function for link $\ell \in \mathcal{L}$
μ	Rigidity coefficient
V	Objective function
$\mathbf{v}$	Gradient of V
L	derivative bound
Mirror ascent	
h	Distance-generating function
D	Bregman divergence
γ	Step size
k	Iteration counter
n	Total number of iterations
$\mathcal{P}$	Prox-mapping

[1] The differentiability assumption could be lifted at the cost of using supergradients instead of gradients, but we do not do so for simplicity.

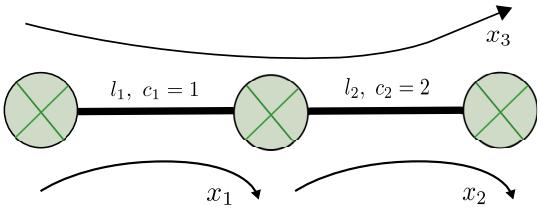

Fig. 1: Network with three sessions sharing two links.

In addition, by controlling g_ℓ, the administrator can over- or under-charge links depending on the situation of the network (for instance, by replacing c_ℓ by $c_\ell(1 \pm \varepsilon_\ell)$ for some $\varepsilon_\ell > 0$). For a detailed discussion, we refer the reader to [18] and references therein.

For reasons that will become clear in the sequel, we take a slight modification of the above flexible framework and focus on the following network utility maximization problem

$$\begin{aligned} \text{maximize} \quad & V(\mathbf{x}) \equiv \sum_{s\in\mathcal{S}} U_s(x_s) - \mu \sum_{\ell\in\mathcal{L}} g_\ell(x_\ell) \\ \text{subject to} \quad & x_s \geq 0 \text{ and } \sum_{s\in\mathcal{S}} x_s \leq C \end{aligned} \tag{NUM}$$

where C is a global capacity bound. This upper bound is an extrinsic parameter which is chosen by the network administrator: for instance, it can be chosen as the sum of link capacities $\sum_\ell c_\ell$, as the maximum link capacity $\max_\ell c_\ell$, or, more simply, as a means to implement a congestion control mechanism as in the classical paper of Kelly et al. [9]. Importantly, we should also note here that it is possible to choose C in a way such that any solution of (NUM_0) is also a solution of (NUM).

All in all, the extra constraint in (NUM) should be treated as a synthetic geometry modification that does not alter the problem's solution set but instead guarantees that interim candidate solutions are well-behaved (in case a solution algorithm needs to be cut short of its execution runtime). Example 1 below provides some additional clarifications:

Example 1 (A toy example). Fig. 1 shows a network consisting of two links ℓ_1 and ℓ_2 with respective capacities $c_1 = 1$ and $c_2 = 2$. Let x_1 be the rate of the session that uses only ℓ_1, x_2 that of the session using only ℓ_2, and x_3 the rate of the session using both links, with corresponding utility functions $U_1(x_1) = x_1$, $U_2(x_2) = x_2$, and $U_3(x_3) = 3x_3$. The feasibility region of (NUM_0) is depicted by the green shaded area in Fig. 2; the larger, overlapping orange area represents the cap constraint of (NUM). In this case, it is easy to see that the solution of (NUM_0) is $x_1^* = 0$, $x_2^* = 1$, and $x_3^* = 1$. Instead of exploring the entire positive orthant as in the standard relaxation of the NUM problem [18], the capped formulation introduces a cut-off which narrows the problem's feasible region while still remaining tractable and scalable from an algorithmic standpoint. Note also that, when one picks the global capacity constraint as above, the solution of (NUM_0) lies within the feasible region of (NUM) and can be recovered by choosing μ appropriately.

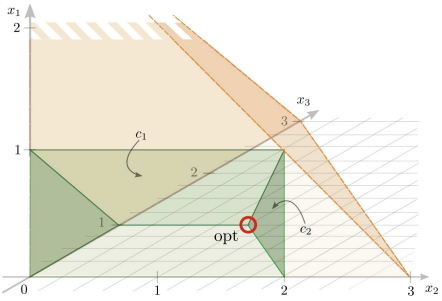

Fig. 2: Feasible region (green), global capacity constraint (orange) and solution profile of Example 1.

III. Gradient Ascent and Mirror Ascent

In this section, we start laying down the algorithmic groundwork for the exponentiated gradient algorithm for NUM (described in detail in Section IV). To that end, we begin with some standing notational conventions and assumptions:

1) We write $\Omega \equiv \{\mathbf{x} \in \mathbb{R}^d_+ : x_s \geq 0 \text{ and } \sum_s x_s \leq C\}$ for the feasible region of (NUM) and $V_* \equiv \max_{\mathbf{x}\in\Omega} V(\mathbf{x})$ for the maximum value of V over Ω.
2) We write $\|\mathbf{x}\|_p = \left(\sum_{s=1}^d |x_s|^p\right)^{1/p}$ for the p-norm of $\mathbf{x}$. When $p = 2$, this is the standard Euclidean norm; we will also make use of the Manhattan norm $\|\mathbf{x}\|_1 = \sum_{s=1}^d |x_s|$ (corresponding to $p = 1$), and the sup norm $\|\mathbf{x}\|_\infty = \max_{s=1,\dots,d}|x_s|$ (corresponding to $p = \infty$).
3) Finally, we make the blanket regularity assumption that

$$\left|\frac{\partial V}{\partial x_s}\right| \leq L \tag{5}$$

for some $L > 0$ and all $\mathbf{x} \in \Omega$, $s \in \mathcal{S}$. In words, this means that the derivatives of V are bounded over the problem's feasible region (i.e., V is Lipschitz continuous).

A. Projected gradient ascent

We begin by revisiting a standard benchmark for solving network utility maximization problems, that of *projected gradient ascent* (PGA). Following [5], this is captured by the recursion

$$\mathbf{x}^{k+1} = \Pi_\Omega(\mathbf{x}^k + \gamma \mathbf{v}^k) \tag{PGA}$$

where:

1) $k = 1, 2, \dots$, is the algorithm's iteration counter.
2) $\mathbf{x}^k \in \Omega$ is the state of the algorithm at iteration k.
3) $\mathbf{v}^k = \nabla V(\mathbf{x}^k)$ is the gradient of the NUM objective at $\mathbf{x}^k$.
4) $\Pi_\Omega \colon \mathbb{R}^d \to \Omega$ is the *Euclidean projector*

$$\Pi_\Omega(\mathbf{w}) = \arg\min_{\mathbf{w}\in\Omega}\{\|\mathbf{x} - \mathbf{w}\|_2^2\}. \tag{6}$$

In words, the PGA algorithm takes a step along the gradient of the objective (modulated by the step-size parameter γ) and, if the result lies outside the problem's feasible region, it projects back to Ω and repeats the process.

To streamline our presentation, we defer the analysis of the algorithm until later in this section (cf. the discussion right after Theorem 1). For now, we only state that, under the blanket

assumptions above, (PGA) achieves the following performance guarantee:

Proposition 1 (Convergence rate of projected gradient ascent)**.** *Suppose that* (PGA) *is run for n iterations with step-size γ. Then, the solution candidate $\bar{\mathbf{x}} \equiv \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}^k$ enjoys the guarantee*

$$V_* - V(\bar{\mathbf{x}}) \leq \frac{C^2}{n\gamma} + \frac{\gamma d L^2}{2}. \tag{7}$$

In particular, if (PGA) *is run for n iterations with step-size $\gamma = C/L \cdot \sqrt{2/(nd)}$, we get*

$$V_* - V(\bar{\mathbf{x}}) \leq LC\sqrt{2d/n}, \tag{8}$$

i.e., $\bar{\mathbf{x}}$ achieves an accuracy certificate $V_* - V(\bar{\mathbf{x}}) \leq \varepsilon$ within at most $2dL^2C^2/\varepsilon^2 = \mathcal{O}(d/\varepsilon^2)$ iterations.

This is the main convergence guarantee for (PGA), so several remarks are in order. First, the step-size choice $\gamma = C/L\sqrt{2/(nd)}$ has been calibrated so as to minimize the RHS of (7): as a result, this is the *optimum* parameter choice given a fixed running horizon. Hence, if we wish to guarantee an accuracy of ε in as few iterations as possible, Proposition 1 indicates that (PGA) should be run for $n = 2dL^2C^2/\varepsilon^2$ iterations with step-size $\gamma = \varepsilon/(dL^2)$.

A second issue concerns the use of averaging, i.e., taking $\bar{\mathbf{x}}$ as a solution candidate instead of the algorithm's "last iterate", $\mathbf{x}^n$. The reason for this is that, without stronger assumptions, a large step-size could lead to overshooting the solution of (NUM), thus causing delays in convergence time; on the other hand, if the step-size is too small, the method will converge even slower. Averaging accelerates this convergence because the barycenter of iterates that oscillate around the solution set of (NUM) tends to be closer to a solution than any individual iterate.

B. Mirror ascent

Even though (PGA) is straightforward to implement and enjoys a clean performance guarantee, the number of iterations required to reach an accuracy certificate of ε scales with the number of sessions d (which could be of the order of millions in IP access domain networks with a few hundred nodes). Scale-up difficulties of this kind are encountered fairly often in large-scale optimization problems, and a promising way to circumvent them is to modify the algorithm's update mechanism so that it takes larger steps when the geometry of the feasible region is more constrained (meaning that there is less risk of overshooting).

This is precisely the intuition behind the so-called *mirror ascent* (MA) method of Nemirovski and Yudin [15] which works by replacing the Euclidean projector of (PGA) with a more geometry-aware update structure. Heuristically, instead of performing the update

$$\begin{aligned}\mathbf{x}^{k+1} &= \Pi_\Omega(\mathbf{x}^k + \gamma\mathbf{v}^k) = \arg\min_{\mathbf{x}\in\Omega}\{\|\mathbf{x}^k + \gamma\mathbf{v}^k - \mathbf{x}\|_2^2\}\\ &= \arg\min_{\mathbf{x}\in\Omega}\{\langle\gamma\mathbf{v}^k, \mathbf{x}^k - \mathbf{x}\rangle + \tfrac{1}{2}\|\mathbf{x}^k - \mathbf{x}\|_2^2\},\end{aligned} \tag{9}$$

the mirror ascent algorithm employs the modified update

$$\begin{aligned}\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}\in\Omega}\{\langle\gamma\mathbf{v}^k, \mathbf{x}^k - \mathbf{x}\rangle + D(\mathbf{x}, \mathbf{x}^k)\},\\ &=: \mathcal{P}_\Omega(\mathbf{x}^k, \gamma\mathbf{v}^k)\end{aligned} \tag{MA}$$

where the so-called "*prox-mapping*" $\mathcal{P}_\Omega$ is induced by the "*Bregman divergence*" term D, itself a generalization of the squared Euclidean norm $\|\mathbf{x}^k - \mathbf{x}\|_2^2$.

The starting point for the definition of the Bregman divergence D (and the associated prox-mapping $\mathcal{P}_\Omega$) is the notion of a *distance-generating function* (DGF), i.e., a function $h\colon \Omega \to \mathbb{R}$ which is:

1) *Continuous* on Ω and *continuously differentiable* on the relative interior $\Omega^\circ = \{\mathbf{x} \in \mathbb{R}_+^d : x_s > 0, \sum_s x_s < C\}$ of Ω.[2]
2) *Strongly convex*, i.e., there exists some $\sigma > 0$ such that

$$h(\mathbf{x}') \geq h(\mathbf{x}) + \langle\nabla h(\mathbf{x}), \mathbf{x}' - \mathbf{x}\rangle + \frac{\sigma}{2}\|\mathbf{x}' - \mathbf{x}\|_p^2 \tag{10}$$

for all $\mathbf{x}, \mathbf{x}' \in \Omega$.[3]

The *Bregman divergence* associated to h is then defined as

$$D(\mathbf{p}, \mathbf{x}) = h(\mathbf{p}) - h(\mathbf{x}) - \langle\nabla h(\mathbf{x}), \mathbf{p} - \mathbf{x}\rangle, \tag{11}$$

i.e., as the difference between $h(\mathbf{p})$ and the best linear approximation of $h(\mathbf{p})$ starting from $\mathbf{x}$.

Remark. For clarity, we mention here that the above properties must be satisfied by the distance-generating function h, *not* the network utility maximization objective function V.

Before continuing, it is instructive to show how (PGA) can be recovered as a special case of (MA). Indeed, going back to Eq. (9), we see that the prox-mapping of (PGA) is simply

$$\mathcal{P}_\Omega(\mathbf{x}, \mathbf{w}) = \Pi_\Omega(\mathbf{x} + \mathbf{w}), \tag{12}$$

and is generated by the Bregman divergence

$$D(\mathbf{x}, \mathbf{x}') = \frac{1}{2}\|\mathbf{x}' - \mathbf{x}\|_2^2, \tag{13}$$

which is itself induced by the Euclidean distance-generating function

$$h(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|_2^2. \tag{14}$$

The calculations needed to verify the above are straightforward, so we omit them; instead, we only note for posterity that the strong convexity modulus of the Euclidean regularizer is 1 relative to the Euclidean norm $\|\cdot\|_2$.

With all this at hand, we are finally in a position to state and prove the main performance guarantee of the MA algorithm applied to (NUM):

Theorem 1 (Convergence rate of mirror ascent)**.** *Let h be a distance-generating function which is σ-strongly convex with respect to the p-norm on $\mathbb{R}^d$. Then, if* (MA) *is run for n iterations with step-size γ, the solution candidate $\bar{\mathbf{x}} \equiv \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}^k$ enjoys the guarantee*

$$V_* - V(\bar{\mathbf{x}}) \leq \frac{Z}{n\gamma} + \frac{\gamma d^{2-2/p}L^2}{2\sigma}, \tag{15}$$

[2] Strictly speaking, we will tacitly assume that h is continuously differentiable on the relative interior of every face of Ω, but this technicality can be safely ignored in what follows.

[3] In the above, the value of p in the p-norm $\|\cdot\|_p$ is a tunable parameter whose role will be discussed below.

with $Z = D(\mathbf{x}^*, \mathbf{x}^1)$ *for some solution* $\mathbf{x}^*$ *of* (NUM). *In particular, if* (MA) *is run for* n *iterations with step-size*

$$\gamma = \frac{1}{L}\sqrt{\frac{2\sigma}{d^{2-2/p}}\frac{Z}{n}}, \tag{16}$$

we get

$$V_* - V(\bar{\mathbf{x}}) \leq L\sqrt{\frac{2d^{2-2/p}}{\sigma}\frac{Z}{n}}, \tag{17}$$

i.e., $\bar{\mathbf{x}}$ *achieves an accuracy certificate* $V_* - V(\bar{\mathbf{x}}) \leq \varepsilon$ *within at most* $2d^{2-2/p}Z/\sigma \cdot L^2/\varepsilon^2$ *iterations.*

This will be our main result for (MA) so, before discussing its proof, we provide some useful remarks:

Remark 1. First, given that the MA algorithm subsumes the PGA algorithm, Theorem 1 subsumes in turn Proposition 1. Indeed, to obtain the bound (7) of Proposition 1, it suffices to recall that the Euclidean regularizer is 1-strongly convex with respect to the Euclidean norm (cf. the discussion above). It is also easy to see that $Z = C^2$ in this case, so the bound (7) follows from (15) by setting $p = 2$, $\sigma = 1$ and $Z = C^2$.

Remark 2. Importantly, the dependence of the convergence rate of (MA) on the dimensionality of the problem is controlled by the coefficient $d^{2-2/p}Z/\sigma$. This observation is crucial for scalability: for the MA algorithm, the exponent of d in $d^{2-2/p}$ must be minimized, i.e., we must focus on the case $p = 1$; concretely, this means that the most suitable distance-generating function for the network utility maximization problem should be examined relative to the Manhattan norm ($p = 1$). Therefore, the challenge is to construct a distance-generating function for which the ratio Z/σ grows as slowly as possible with d; this will be our main objective in the next section.

Proof of Theorem 1: By the definition of the prox-mapping and that of the Bregman divergence, (MA) satisfies the recursion

$$\nabla h(\mathbf{x}^{k+1}) - \nabla h(\mathbf{x}^k) - \gamma \mathbf{v}^k = 0. \tag{18}$$

Hence, by standard properties of the Bregman divergence [see e.g., 14, Lemma 2.1], we get

$$D^{k+1} - D(\mathbf{x}^*, \mathbf{x}^k) \leq \gamma\langle \mathbf{x}^k - \mathbf{x}^*, \mathbf{v}^k\rangle + \frac{\gamma^2}{2\sigma}\|\mathbf{v}^k\|_q^2, \tag{19}$$

where $\mathbf{x}^* \in \arg\max V$ and q is such that $1/q + 1/p = 1$. Then, letting $D^k = D(\mathbf{x}^*, \mathbf{x}^k)$ and telescoping the above yields

$$\sum_{k=1}^n \left(D^{k+1} - D^k + \gamma\langle \mathbf{x}^* - \mathbf{x}^k, \mathbf{v}^k\rangle\right) \leq \gamma^2 \sum_{k=1}^n \frac{\|\mathbf{v}^k\|_q^2}{2\sigma} \tag{20}$$

implying in turn that

$$D^{n+1} - D^1 + \gamma \sum_{k=1}^n \langle \mathbf{x}^* - \mathbf{x}^k, \mathbf{v}^k\rangle \leq \frac{\gamma^2}{2\sigma}\sum_{k=1}^n \|\mathbf{v}^k\|_q^2. \tag{21}$$

Then, since the Bregman divergence is nonnegative, we get

$$\sum_{k=1}^n \langle \mathbf{x}^* - \mathbf{x}^k, \nabla V(\mathbf{x}^k)\rangle \leq \frac{D(\mathbf{x}^*, \mathbf{x}^1)}{\gamma} + \frac{\gamma\sum_{k=1}^n \|\nabla V(\mathbf{x}^k)\|_q^2}{2\sigma}. \tag{22}$$

By concavity, we also have $\langle \mathbf{x}^* - \mathbf{x}^k, \nabla V(\mathbf{x}^k)\rangle \geq V_* - V(\mathbf{x}^k)$; hence, by Jensen's inequality applied to $\bar{\mathbf{x}} = n^{-1}\sum_{k=1}^n \mathbf{x}^k$, Eq. (22) gives

$$V_* - V(\bar{\mathbf{x}}) \leq \frac{D(\mathbf{x}^*, \mathbf{x}^1)}{\gamma} + \frac{\gamma d^{2/q} L^2}{2\sigma}, \tag{23}$$

where we used the definition of the q-norm and the standing assumption that $|\partial V/\partial x_s| \leq L$. The bound (15) then follows by noting that $2/q = 2 - 2/p$ (by definition).

Finally, the value of the optimum step-size (16) follows by minimizing the RHS of (15) with respect to γ; the rest of our claims then follow from elementary algebraic manipulations. ■

IV. Gradient Exponentiation

A. The exp-NUM algorithm

The analysis of the previous section shows that the most suitable distance-generating function for the NUM problem should be constructed relative to the Manhattan norm (since the value $p = 1$ minimizes the dependence of the coefficient $d^{2-2/p}$ in (15) with respect to d). In problems where the feasible region is the probability simplex $\Delta = \{\mathbf{z} \in \mathbb{R}_+^{d+1} : \sum_{s=0}^d z_s = 1\}$, a popular choice is the (negative) entropy function

$$\phi(\mathbf{z}) = \sum\nolimits_{s=0}^d z_s \log z_s, \tag{24}$$

which is known to be 1-strongly convex with respect to the Manhattan norm [19, p. 136].

Motivated by the above, we will focus in what follows on the *modified* entropy function

$$h(\mathbf{x}) = \sum\nolimits_{s\in\mathcal{S}} x_s \log x_s + x_0 \log x_0, \tag{25}$$

where $x_0 = C - \sum_{s=1}^d x_s$ is a "slack variable" measuring the (Manhattan) distance between a state $\mathbf{x} \in \Omega$ and the boundary face $\sum_{s=1}^d x_s$ of Ω. This distance-generating function will be the cornerstone of our analysis, so we begin by deriving the associated mirror ascent algorithm.

To that end, the first step is to obtain an explicit expression for the prox-mapping $\mathcal{P}_\Omega$ induced by the DGF (25). This is provided by the following lemma:

Lemma 1 (Entropic prox-mapping)**.** *With notation as above, the prox-mapping* $\mathcal{P}_\Omega$ *associated to* (25) *is*

$$\mathcal{P}_\Omega(\mathbf{x}, \mathbf{w}) = C\frac{(x_s \exp(w_s))_{s\in\mathcal{S}}}{C + \sum_{s\in\mathcal{S}} x_s[\exp(w_s) - 1]} \tag{26}$$

Proof: By definition, $\mathcal{P}_\Omega(\mathbf{x}, \mathbf{w})$ solves the problem

$$\mathcal{P}_\Omega(\mathbf{x}, \mathbf{w}) = \arg\min_{\mathbf{x}'\in\Omega}\{\langle \mathbf{w}, \mathbf{x} - \mathbf{x}'\rangle + D(\mathbf{x}', \mathbf{x})\} \tag{27}$$

with $D(\mathbf{x}', \mathbf{x})$ denoting the Bregman divergence

$$D(\mathbf{x}', \mathbf{x}) = \sum\nolimits_{s\in\mathcal{S}} x'_s \log\frac{x'_s}{x_s} + x'_0 \log\frac{x'_0}{x_0}, \tag{28}$$

as derived from definition (11). To solve this problem, we will start with the ansatz that $\mathcal{P}_\Omega(\mathbf{x}, \mathbf{w})$ is an interior point of Ω and treat (27) as an unconstrained problem; if the solution of the unconstrained problem ends up being an interior point of Ω, it will be a true solution of (27).

To proceed, let

$$\Lambda(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} - \mathbf{x}'\rangle + D(\mathbf{x}', \mathbf{x}) \tag{29}$$

denote the objective function of (27). Then, by differentiating with respect to $\mathbf{x}'$, we obtain

$$\frac{\partial \Lambda}{\partial x'_s} = -w_s + \log\left(\frac{x'_s}{x_s}\right) + 1 - \log\left(\frac{x'_0}{x_0}\right) - 1, \tag{30}$$

where we used the fact that $\partial x'_0/\partial x'_s = -1$. Accordingly, setting these derivatives equal to zero, we get

$$\frac{x'_s}{x'_0} = \frac{x_s}{x_0}\exp(w_s), \tag{31}$$

and hence, summing over all sessions $s \in \mathcal{S}$ yields

$$\frac{C - x'_0}{x'_0} = \frac{\sum_s x_s \exp(w_s)}{x_0} \tag{32}$$

i.e., $x'_0 = C \cdot x_0/(x_0 + \sum_s x_s \exp(w_s))$. Our result then follows by substituting in (31) and backsolving. ∎

With this lemma at hand, we obtain the *exponentiated gradient ascent* (EGA) algorithm

$$x_s^{k+1} = C\frac{x_s^k \exp(\gamma \partial_s V(\mathbf{x}^k))}{C + \sum_{s\in\mathcal{S}} x_s^k[\exp(\gamma\partial_s V(\mathbf{x}^k)) - 1]}, \tag{EGA}$$

which, in the present context, will be referred to as the *exponentiated gradients algorithm for NUM* (exp-NUM). For concreteness, a pseudocode implementation of exp-NUM can be found in Algorithm 1; for a schematic illustration of the differences between exp-NUM and the PGA algorithm of the previous section, see also Fig. 3.

Algorithm 1: exponentiated gradients algorithm for NUM

Require: step-size γ, number of iterations n

function MAIN
  $x^1 \leftarrow \{C/(d+1), C/(d+1), \dots, C/(d+1)\}$
  for k← 1 to n **do**
    $\mathbf{v}^k \leftarrow \nabla V(\mathbf{x}^k)$
    $\mathbf{x}^{k+1} \leftarrow C \cdot \mathbf{x}^k \odot \exp(\gamma \mathbf{v}^k)/(C - \sum_s x_s^k \cdot (1 - \exp(\gamma v_s^k)))$
  end for
  return $\bar{\mathbf{x}} = \frac{1}{n}\sum_{k=1}^n \mathbf{x}^k$
end function

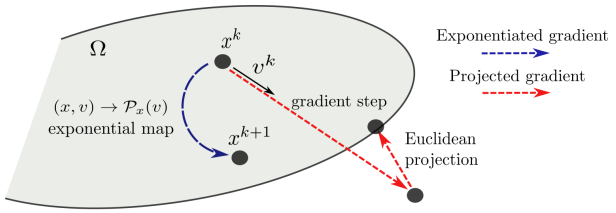

Fig. 3: Projected vs. exponentiated gradient ascent.

B. Convergence rate analysis of exp-NUM

We now turn to the convergence speed of the exp-NUM algorithm. Since exp-NUM is, first and foremost, a mirror ascent method, the analysis of the previous section shows that it suffices to calculate the following quantities: *a*) a lower bound for the strong convexity modulus of h relative to the ℓ_1 norm on Ω; *b*) an upper bound for $Z = D(\mathbf{x}^*, \mathbf{x}^1)$ relative to a convenient initialization of the algorithm. We begin below with the former:

Lemma 2 (Strong convexity modulus)**.** *The entropic DGF* (25) *is* $(1/C)$*-strongly convex with respect to the* ℓ_1 *norm over* Ω*.*

Proof: As stated above, the negative entropy ϕ of (24) is 1-strongly convex over the probability simplex $\Delta = \{\mathbf{x} \in \mathbb{R}_+^{d+1} : \sum_{s=0}^d x_s = 1\}$ [19, p. 136]. To leverage this property, set $z_s = x_s/C$ for $s = 1, \dots, d$, and $z_0 = x_0/C = 1 - \sum_{s=1}^d x_s$. Then, up to an additive constant that does not depend on $\mathbf{x}$, we get

$$\begin{aligned} C\phi\left(\frac{\mathbf{x}}{C}\right) &= C\sum\nolimits_{s\in\mathcal{S}} \frac{x_s}{C} \cdot \log\left(\frac{x_s}{C}\right) \\ &= \sum\nolimits_{s\in\mathcal{S}} x_s \log x_s - \sum\nolimits_{s\in\mathcal{S}} x_s \log C \\ &= \sum\nolimits_{s\in\mathcal{S}} x_s \log x_s + \left(C - \sum\nolimits_{s\in\mathcal{S}} x_s\right)\log C - C\log C \\ &= h(\mathbf{x}) - C\cdot\log C. \end{aligned} \tag{33}$$

Since ϕ has strong convexity modulus 1, the function $\phi(\mathbf{x}/C)$ will have modulus $1/C^2$. Thus, $h(\mathbf{x}) = C \cdot \phi(\mathbf{x}/C) + C\log C$ is $(1/C)$-strongly convex over Ω. ∎

As a last step, we need an initialization point which is both easy to implement and is as close as possible (in terms of the Bregman divergence) to all other points in Ω. In view of the above, the most natural initialization candidate is the barycenter of Ω, viz.

$$\mathbf{x}^c = (C/(d+1), \dots, C/(d+1)). \tag{34}$$

With this initialization, we obtain the following convergence speed estimate:

Theorem 2 (Convergence rate of exp-NUM)**.** *The output* $\bar{\mathbf{x}}$ *of Algorithm 1 enjoys the guarantee*

$$V_* - V(\bar{\mathbf{x}}) \leq \frac{C\log(d+1)}{n\gamma} + \frac{\gamma C L^2}{2}. \tag{35}$$

In particular, if Algorithm 1 is run for n *iterations with step-size* $\gamma = L^{-1}\sqrt{2\log(d+1)/k}$*, we get*

$$V_* - V(\bar{\mathbf{x}}) \leq LC\sqrt{2\log(d+1)/n}, \tag{36}$$

i.e., $\bar{\mathbf{x}}$ *achieves an accuracy certificate* $V_* - V(\bar{\mathbf{x}}) \leq \varepsilon$ *within at most* $2\log(d+1)\cdot L^2C^2/\varepsilon^2 = \mathcal{O}(\log d/\varepsilon^2)$ *iterations.*

Proof. As a first step, we will establish an upper bound for the quantity $Z = D(\mathbf{x}^*, \mathbf{x}^1) = D(\mathbf{x}^*, \mathbf{x}^c)$. To that end, substituting in (28), we obtain for all $\mathbf{x} \in \Omega$:

$$D(\mathbf{x}, \mathbf{x}^c) = \sum\nolimits_{s=0}^d x_s \log x_s - C\log C + C\log(d+1). \tag{37}$$

Therefore, since the maximum of the expression $\sum_{s=0}^d x_s \log x_s$ over Ω is obtained when $x_s = C$ for some $s \in \{0, \dots, d\}$ and $x_{s'} = 0$ for all $s' \neq s$, we readily get

$$D(\mathbf{x}, \mathbf{x}^c) \leq C\log C - C\log C + C\log(d+1) = C\log(d+1). \tag{38}$$

This shows that $D(\mathbf{x}^*, \mathbf{x}^c) \leq \log(d+1)$. The bound (35) then follows by applying Theorem 1 and substituting the parameter values $Z = C\log(d+1)$, $p = 1$ and $\sigma = 1/C$ in the estimated bound (15). □

TABLE III: Default parameters used in the simulations.

d	1000	$\lvert\mathcal{L}\rvert$	100
c_l	$\mathcal{N}(0.5, 0.1)$	C	$\max_l c_l$
w_s	$\mathcal{N}(0.5, 0.1)$	μ	0.1

C. Multipath scenario

We close this section by discussing the scenario where the traffic of each session can split into multiple paths. In this case, the network controller can route traffic over different paths in order to reduce the load on specific links. This multipath scenario can be formulated through the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{s\in\mathcal{S}} U_s\left(\sum_{p\in\mathcal{P}_s} x_{sp}\right) - \mu\cdot\sum_{\ell\in\mathcal{L}} g_\ell\left(\sum_{s:\ell\in\mathcal{P}_s} x_{sp}\right) \\ \text{subject to} \quad & \textstyle\sum_{s\in S}\sum_{p\in\mathcal{P}_s} x_{sp} \le C \end{aligned} \tag{NUM-mult}$$

The simplicial structure of (NUM-mult) pushes for an algorithm based on mirror ascent, for the same reasons as before. Concretely, to extend (25) to the multipath setting, consider the distance-generating function

$$h(\mathbf{x}) = \sum\nolimits_{s,p} x_{sp}\cdot\log x_{sp} + \left(C - \sum\nolimits_{s,p} x_{sp}\right)\cdot\log\left(C - \sum\nolimits_{s,p} x_{sp}\right). \tag{39}$$

Similarly to Lemma 1, we derive the *exp-multiNUM* proximal map $\mathcal{P}_{\mathbf{x}}(\mathbf{v})$ for the multipath scenario using the DGF above:

$$\mathcal{P}_\Omega(\mathbf{x},\mathbf{w}) = C\cdot\frac{\mathbf{x}\odot\exp(\mathbf{w})}{C - \sum_{s,p} x_{sp}\cdot(1-\exp(w_{sp}))}, \tag{40}$$

According to this algorithm, we are still able to obtain logarithmic dependence on the problem's dimensionality (considering the product of paths and sessions). Specifically, following the same reasoning as in the previous section, we get the convergence rate estimate

$$V(\mathbf{x}^*) - V(\overline{\mathbf{x}}) \le C\cdot L\cdot\sqrt{\frac{2\cdot\log(\sum_s|\mathcal{P}_s|+1)}{k}}. \tag{41}$$

The proof is similar to that of Theorem 2 and is omitted due to space limitations.

V. Numerical Experiments

In this section, we evaluate the performance of the proposed exp-NUM algorithm via extensive numerical simulations. Here, the main goals are to show the rate of convergence compared to other gradient-based methods, and to discuss how the problem dimensionality affects convergence and algorithm complexity per iteration. Our simulations consider a wide number of scenarios to cover complementary aspects of the NUM problem. We use two different α-fair models, namely $\alpha = 0$ and $\alpha = 1$, corresponding respectively to sum-rate maximization and proportional fairness. The underneath network topology used is based either on Erdős-Rényi or Barabási-Albert models. An exhaustive list of the parameters used in the simulations is provided in Table III.

TABLE IV: Execution time per iteration of different iterative algorithms.

Algorithm	$d = 100$	$d = 1000$	$d = 10.000$
exp-NUM	$5{,}0\times10^{-4}$ s	$6{,}9\times10^{-3}$ s	$7{,}3\times10^{-2}$ s
GD	$7{,}0\times10^{-4}$ s	$9{,}5\times10^{-3}$ s	$1{,}1\times10^{-1}$ s
Newton	$1{,}1\times10^{-2}$ s	$2{,}1\times10^{0}$ s	∞

A. Convergence rate

In Fig. 4 we show the main outcomes of our simulations. Specifically, we evaluate four scenarios by combining the two α-fair models ($\alpha = 0$ and $\alpha = 1$) with the aforementioned network models. For each scenario, we plot the objective value $V(\overline{\mathbf{x}})$ for the first 100.000 iterations (top), and the corresponding error (i.e., the difference with the optimal solution) in log-log scale (bottom). The plots compare exp-NUM with the PGA algorithm, with step-sizes computed according to Proposition 1 and Theorem 2 in order to reach a fixed accuracy threshold, common for both algorithms (to ensure fair testing).

The first immediate result that we observe from the plots is that exp-NUM approaches a solution of (NUM) in a relatively small number of iterations. While both algorithms eventually get close to the optimal objective value, the error for exp-NUM after 100.000 iterations is between one and two orders of magnitude smaller than gradient ascent. This observation is inline with our theoretical findings: by Theorem 2, the expected theoretical bound for exp-NUM is about 20 times tighter than that of PGA when $d = 1000$, and this difference only grows larger as the number of variables increases.

Another interesting observation is that the outcomes of the simulations do not show substantial differences based on the topology of the network. This is also confirmed by further analysis performed using the Watts-Strogatz model (but which we do not report due to space limitations). This is an important achievement for our algorithm which guarantees that its convergence rate is basically independent of the underlying network topology.

B. Problem dimensionality

The main reason to use first-order, gradient-based methods is their low *per-iteration* complexity, i.e., the fact that their updates remain relatively cheap in massively large systems. To make this precise, in Table IV, we compare the execution time per iteration of exp-NUM and PGA along with the Newton-based method of [2]. In the test machine used for our experiments, Netwon's method can be used efficiently only when d is fairly small: as the dimension of the problem increases, the need for the inversion of the Hessian matrix, which requires large processing and memory resources, makes the computation infeasible. In fact, even within the class of first-order methods, each iteration of exp-NUM is around 40% faster than a PGA update because exp-NUM does not require an projection step (which, in the case of PGA involves a sorting subroutine, and thus increases the per-iteration complexity by an $\mathcal{O}(\log d)$ factor). We summarize all this in Table IV.

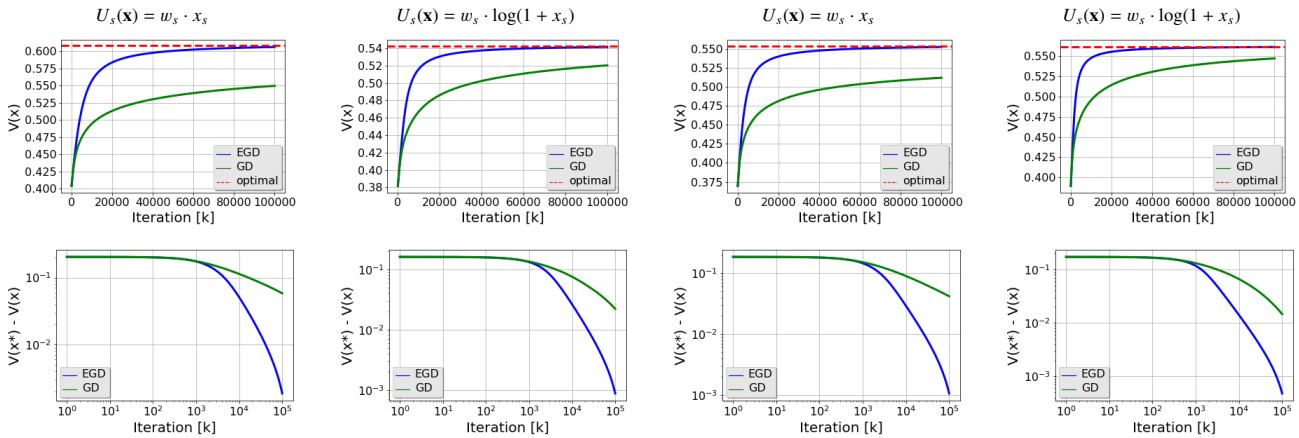

Fig. 4: Value convergence (in log-log scale) for exp-NUM and PGA in Erdős-Rényi and Barabási-Albert networks.

VI. Conclusions

In this paper, we examined the challenging scenario of resource allocation in large-scale networks with massive numbers of active sessions. In this setting, the dimensionality d of the system (number of sessions) represents a significant impediment to solving the network utility maximization rapidly and efficiently. To overcome this, we proposed the exp-NUM algorithm, a modified gradient algorithm that exploits the geometry of the problem to take bold (but safe) step-size decisions. We have shown that exp-NUM is effectively *dimension-free* and attains an ε-optimal solution within $\mathcal{O}(\log d/\varepsilon^2)$ iterations – an exponential improvement over the poly(d) dimensionality dependence of first-order benchmark methods. In addition to these theoretical guarantees, we also provided a battery of simulation experiments which verify the scheme's superior scalability and fast convergence. We find this property particularly appealing as it provides a springboard for the application of similar techniques to a wide range of resource allocation problems in the area of networking.

An important question that remains is whether the algorithm can also be accelerated relative to the desired accuracy ε – e.g., to obtain an ε-optimal solution in no more than $\mathcal{O}(\log d/\sqrt{\varepsilon})$. We believe this should be possible by leveraging Nesterov's fast gradient method; we defer this analysis to the future.

References

[1] Z. Allybokus, K. Avrachenkov, J. Leguay, and L. Maggi. Real-time fair resource allocation in distributed software defined networks. In *2017 29th International Teletraffic Congress (ITC 29)*, volume 1, pages 19–27, Sept 2017.

[2] S. Athuraliya and S. H. Low. Optimization flow control with newton-like algorithm. *Telecommun. Syst.*, 15(3-4):345–358, Dec. 2000.

[3] H. Attouch. *Variational Convergence for Functions and Operators*. Pitman, Boston, MA, 1984.

[4] A. Beck, A. Nedić, A. Ozdaglar, and M. Teboulle. An $o(1/k)$ gradient method for network resource allocation problems. *IEEE Transactions on Control of Network Systems*, 1(1):64–73, March 2014.

[5] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1995.

[6] M. Dehghan, L. Massoulie, D. Towsley, D. Menasche, and Y. C. Tay. A utility optimization approach to network cache design. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016.

[7] R. Deng, Y. Zhang, S. He, J. Chen, and X. Shen. Maximizing network utility of rechargeable sensor networks with spatiotemporally coupled constraints. *IEEE Journal on Selected Areas in Communications*, 34(5):1307–1319, May 2016.

[8] R. Gupta, L. Vandenberghe, and M. Gerla. Centralized network utility maximization over aggregate flows. In *2016 14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 1–8, May 2016.

[9] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, Mar 1998.

[10] S. H. Low and D. E. Lapsley. Optimization flow control. i. basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, Dec 1999.

[11] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Trans. Netw.*, 8(5):556–567, Oct. 2000.

[12] A. Nedić and A. Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization*, 19(4):1757–1780, 2009.

[13] M. J. Neely. Energy optimal control for time-varying wireless networks. *IEEE Transactions on Information Theory*, 52(7):2915–2934, July 2006.

[14] A. S. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[15] A. Nemirovsky and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. A Wiley-Interscience publication. Wiley, 1983.

[16] D. P. Palomar and M. Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8):1439–1451, Aug 2006.

[17] B. Radunović and J.-Y. L. Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Trans. Netw.*, 15(5):1073–1083, Oct. 2007.

[18] S. Shakkottai and R. Srikant. Network optimization and control. *Foundations and Trends® in Networking*, 2(3):271–379, 2008.

[19] S. Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

[20] E. Wei, A. Ozdaglar, and A. Jadbabaie. A distributed newton method for network utility maximization; part ii: Convergence. *IEEE Transactions on Automatic Control*, 58(9):2176–2188, Sept 2013.

[21] H. Yu and M. J. Neely. A simple parallel algorithm with an $o(1/t)$ convergence rate for general convex programs. *SIAM Journal on Optimization*, 27(2):759–783, jan 2017.

[22] M. Zargham, A. Ribeiro, and A. Jadbabaie. Accelerated backpressure algorithm. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 2269–2275, Dec 2013.