End-to-End Distributed Flow Control for Networks with Nonconcave Utilities

Mahmoud Ashour, Jingyao Wang, Necdet Serhat Aybat, Constantino Lagoa, and Hao Che

Abstract—This paper proposes near-optimal decentralized allocation for traffic generated by real-time applications in communication networks. The quality of experience perceived by users in practical applications cannot be accurately modeled using concave functions. Therefore, we tackle the problem of optimizing general nonconcave network utilities. The approach for solving the resulting nonconvex network utility maximization problem relies on designing a sequence of convex relaxations whose solutions converge to a point that characterizes an optimal solution of the original problem. Three different algorithms are designed for solving the proposed convex relaxation, and their theoretical convergence guarantees are studied. All proposed algorithms are distributed in nature, where each user independently controls its traffic in a way that drives the overall network traffic allocation to an optimal operating point subject to resource constraints. All computations required by the algorithms are performed independently and locally at each user using local information available to that user. We highlight the tradeoff between the convergence speed and the network overhead required by each algorithm. Furthermore, we demonstrate the robustness and scalability of these algorithms by showing that traffic is automatically rerouted in case of a link failure or having new users joining the network. Numerical results are presented to validate our findings.

Index Terms—Distributed optimization, nonconcave utility maximization, traffic engineering

1 Introduction

The growing popularity of IP video services puts an ever increasing strain on the communication, computing, and storage resources at global scale. According to Cisco Inc. annual Visual Networking Index (VNI) forecast report [2], IP video traffic will be 82 percent of all consumer Internet traffic by 2021, up from 73 percent in 2016. Furthermore, [2] investigates user preference on the type of video delivery services and concludes that the Quality-of-Experience (QoE) plays a dominating role in the user's choice over other categories such as content, timing, quality, ease-of-use, portability, interactivity and sharing. Consequently, the success of the video service ecosystem largely depends on its ability to deal with the ever growing resource demands for video services, while being able to provide desired QoE to video users. However, the existing end-to-end flow rate adaptation solutions (e.g., the window-based Transport Control Protocol (TCP) congestion control mechanism [3]) and the in-network traffic engineering solutions (e.g., the flow-based Equal-Cost-Multi-Path (ECMP) load balancing mechanism [4]) are inadequate to provide such capacities for the following reasons. First, the QoE perceived by video users, as a function of flow rate, generally exhibits staircase behaviors [5]. In addition, the utility for voice applications is better described as a sigmoidal function with a convex part at low rate and a concave part at high rate and a single inflection point separating the two parts [6]. Such user utilities are nonconcave; hence, they are notoriously difficult to optimize. Most of the existing end-to-end mechanisms, including TCP and optimization-based solutions, can only deal with concave user utilities [7]. Second, the traditional ECMP and its variations, which are not optimization-based, are susceptible to unbalanced distribution of traffic due to "elephant flows" [8]. Due to lack of sufficient theoretical underpinnings, how such end-to-end and in-network mechanisms impact the overall health of the Internet, in terms of stability and optimality, is still unknown.

In this paper, we aim to develop the much needed theoretical underpinning upon which distributed, user-utility-aware, optimization-based, integrated end-to-end and in-network traffic allocation mechanisms can be designed. Thus, we propose near-optimal decentralized allocation algorithms for traffic generated by real-time applications in communication networks. The contributions of this paper are two-fold. First, we tackle the problem of optimizing a generalized class of nonconcave network utilities. The approach used to solve the resulting nonconvex network utility maximization (NUM) problem relies on designing a sequence of convex relaxations whose solutions converge to a point that characterizes an optimal solution of the original problem. Second, we design three different algorithms for solving the proposed convex relaxation and derive their theoretical convergence guarantees. The optimality of traffic allocations produced by these algorithms is with respect to the convex relaxation. All proposed

- *M. Ashour and C. Lagoa are with the Department of Electrical Engineering and Computer Science, Penn State University, State College, PA 16802. E-mail: {mma240, cml18}@psu.edu.*
- *J. Wang is with the Department of Automation, Xiamen University, Xiamen 36000, China. E-mail: wangjingyao1@xmu.edu.cn.*
- *N.S. Aybat is with the Department of Industrial Engineering, Penn State University, State College, PA 16802. E-mail: nsa10@psu.edu.*
- *H. Che is with the Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, TX 76019. E-mail: hche@cse.uta.edu.*

Manuscript received 30 Sept. 2017; revised 21 Feb. 2018; accepted 26 Apr. 2018. Date of publication 28 June 2018; date of current version 11 Sept. 2019. (Corresponding author: Mahmoud Ashour.)
Recommended for acceptance by J. Liu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TNSE.2018.2851844

algorithms are distributed, where each user independently controls its traffic in a way that drives the overall network to an optimal operating point subject to resource constraints. All computations required by the algorithms are performed independently and locally at each user using local information and some communication overhead. Numerical simulations demonstrate the robustness and scalability of these algorithms.

1.1 Related Work

There are numerous results on the optimization-based traffic control in the existing literature. Although some of them design dynamic load balancing algorithms, the proposed algorithms are no longer distributed by design and require the knowledge of some global information on the network. We review the literature with emphasis on references that provide distributed, optimization-based solutions, which are most relevant to the approach developed in this paper.

Generally, the optimization-based, distributed traffic control problem is formulated as a network utility maximization problem by adopting a fluid-flow model and considering the link capacity constraints. One approach (e.g., [9], [10]) to address this problem is to incorporate link congestion costs into the overall utility function so that the constrained problem can be approximated by an unconstrained problem. The resulting optimization problem can then be solved by using a gradient-based first-order algorithm. Another approach (e.g., [11], [12], [13]) considers a Lagrangian dual based problem. Instead of directly solving the dual problem, the second approach solves a relaxation of the dual problem after closely approximating it through incorporating a price function into the overall utility function. Using this approach, distributed control laws proven to be locally stable in the presence of variable feedback delays are proposed.

A third approach (e.g., [14], [15], [16], [17], [18], [19]) is to solve the original problem directly. In [15], [16], [18], [19], this problem is addressed by using a technique based on the theory of Sliding Mode Control. Both end-to-end [15], [16], [19] and hop-by-hop [18] optimal control laws are proposed that allow multipath forwarding and enable multiple classes of services, and require minimum feedback for control.

However, all of the above algorithms are designed to maximize concave utility functions. There have been only few publications on centralized algorithms [20] and distributed algorithms [21], [22] for non-concave utility maximization. Reference [20] proposes a centralized algorithm based on sum-of-squares (SoS) relaxations and positivstellensatz theorem in real algebraic geometry to calculate approximations of the optimal solution along with some performance bounds to evaluate the approximation error. This efficient but centralized numerical method is suitable for optimizing utilities that can be transformed to polynomial utilities. In [21], the authors propose distributed but suboptimal heuristics for sigmoidal utilities. Reference [22] determines the optimality conditions for the canonical price-based distributed algorithm to converge globally for nonlinear utilities. These two approaches illustrate the choice between admission control and capacity planning to deal with non-convexity. However, neither approach provides a theoretically polynomial-time and practically efficient algorithm (centralized or distributed) for non-concave utility maximization.

Finally, our previous work [23] designs a distributed traffic allocation algorithm that performs hop-by-hop data rate adaptation and load balancing. While the work in [23] is useful for providing sophisticated service quality features at the inter-networking layer in a connectionless network, the solution in this paper is particularly powerful as it allows similar sophisticated service quality features to be adopted at the transport or higher layers. A preliminary version of this work has appeared in [1]. Indeed, the class of utility functions considered in this manuscript contains that considered in [1] as a special case. Moreover, in addition to the algorithm proposed in [1], two related new distributed algorithms with theoretical convergence guarantees are derived in this work.

2 Preliminaries

This section outlines the notation used throughout the paper and presents some key background material.

2.1 Notation

We denote the n-dimensional real vector space by $\mathbb{R}^n$, and $\mathbb{R}^n_+$ is the nonnegative orthant. The set of integers is $\mathbb{Z}$, and $\mathbb{Z}_+$ is the set of nonnegative integers. We represent scalars by lowercase letters, e.g., x, vectors by boldface lowercase letters, e.g., $\mathbf{x}$, matrices with boldface uppercase letters, e.g., $\mathbf{X}$, and sets with calligraphic letters, e.g., $\mathcal{X}$. The cardinality of the set $\mathcal{X}$ is denoted by $|\mathcal{X}|$. Vectors are viewed as column vectors and the transpose of $\mathbf{x}$ is $\mathbf{x}^\top$. Given $n \in \mathbb{Z}_+$, define $\mathcal{I}_n = \{1, \ldots, n\}$. Suppose for each $i \in \mathcal{I}_n$, $\mathbf{x}_i \in \mathbb{R}^{m_i}$ for some $m_i \in \mathbb{Z}_+$, then $\mathbf{x} = [\mathbf{x}_i]_{i \in \mathcal{I}_n}$ is constructed through vertical concatenation, i.e., $\mathbf{x} = [\mathbf{x}_1^\top, \ldots, \mathbf{x}_n^\top]^\top$. Moreover, for each $i \in \mathcal{I}_n$, the jth entry of $\mathbf{x}_i = [x_{i,j}]_{j \in \mathcal{I}_{m_i}}$ is denoted by $x_{i,j}$. For the case, $m_i = 1$ for all $i \in \mathcal{I}_n$, the i-th entry of $\mathbf{x} = [x_i]_{i \in \mathcal{I}_n} \in \mathbb{R}^n$ is similarly denoted by x_i for each $i \in \mathcal{I}_n$. We use $\|\mathbf{x}\|_p$ to denote the p-th norm of $\mathbf{x}$, and omit the subscript p for the Euclidean norm. A vector of all ones is denoted by $\mathbf{1}$. The binary function $1_{\mathcal{X}}(\mathbf{x})$ equals 1 if $\mathbf{x} \in \mathcal{X}$ and is 0 otherwise. We use $\Pi_{\mathcal{Y}}(\mathbf{x})$ to denote the Euclidean projection of $\mathbf{x}$ onto $\mathcal{Y}$, i.e., $\Pi_{\mathcal{Y}}(\mathbf{x}) = \operatorname{argmin}\{\|\mathbf{y} - \mathbf{x}\| : \mathbf{y} \in \mathcal{Y}\}$. For $\mathbf{x} \in \mathbb{R}^n$, $\Pi_{\mathbb{R}^n_+}(\mathbf{x})$ is denoted by $\mathbf{x}^+$, i.e., $\mathbf{x}^+ = \max(\mathbf{x}, \mathbf{0})$, where the max operation is interpreted component-wise.

2.2 Background

We briefly introduce some results on the theory of polynomial optimization, and provide a concise description of convex optimization algorithms relevant to this work.

2.2.1 Theory of Moments

The following theorem can be used to convert polynomial optimization problems into an equivalent infinite dimensional convex semidefinite program (SDP) over the moments of probability measures [24]; hence, by appropriately truncating the moment sequences, one can obtain a sequence of convex SDPs with increasing size.

Theorem 1. *Let f be a real-valued polynomial, $\mathcal{F} \subset \mathbb{R}^n$ be a semialgebraic set, and $\mathcal{M}(\mathcal{F})$ be the set of finite positive Borel measures supported on $\mathcal{F}$. Then, the problems*

$$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathcal{F}\}, \tag{1a}$$

$$\min\left\{\int_{\mathcal{F}} f d\mu : \mu \in \mathcal{M}(\mathcal{F}),\ \mu(\mathcal{F}) = 1\right\}, \tag{1b}$$

are equivalent in the following sense

- *a) the optimal values of (1a) and (1b) are the same;*
- *b) if $\mathbf{x}^*$ is an optimal solution to (1a), then the Dirac measure centered at $\mathbf{x}^*$ is optimal to (1b);*
- *c) if μ^* is an optimal solution to (1b), then any point in the support of μ^* is optimal to (1a).*

Proof. a) follows from [25, Theorem 5.1]. b) and c) easily follow from a). □

The problem of moments bridges the gap between the optimization over a space of probability measures whose support is contained in a certain set and the optimization over the moments of such measures. More precisely, given a sequence of scalars $\{t_j\}_{j=0}^{s}$, the problem of moments is to determine whether there exists a representing Borel measure that has $\{t_j\}_{j=0}^{s}$ as its first $(s+1)$ moments. The following theorem provides necessary and sufficient conditions for the existence of Borel measures whose support is included in bounded symmetric intervals of the real line [25].

Theorem 2. *Given $\mathbf{t} = [t_j]_{j\in\{0,\ldots,s\}}$, there exists a Borel measure $\mu(.)$ with support contained in $\mathcal{I} = [a, b]$ such that $t_j = \int_{\mathcal{I}} y^j d\mu$ for $j \in \{0, \ldots, s\}$ if and only if*

i) the following holds when $s = 2n$ for some $n \in \mathbb{Z}_+$:

$$\mathbf{H}_n(\mathbf{t}) \succeq 0, \tag{2}$$

$$(a+b)\mathbf{B}_{n-1}(\mathbf{t}) \succeq ab\mathbf{H}_{n-1}(\mathbf{t}) + \mathbf{C}_{n-1}(\mathbf{t}), \tag{3}$$

ii) the following holds when $s = 2n+1$ for some $n \in \mathbb{Z}_+$:

$$b\mathbf{H}_n(\mathbf{t}) \succeq \mathbf{B}_n(\mathbf{t}), \tag{4}$$

$$\mathbf{B}_n(\mathbf{t}) \succeq a\mathbf{H}_n(\mathbf{t}), \tag{5}$$

where $\mathbf{H}_n(\mathbf{t})$, $\mathbf{B}_n(\mathbf{t})$, and $\mathbf{C}_n(\mathbf{t})$ denote $(n+1)\times(n+1)$ Hankel matrices defined as $\mathbf{H}_n(\mathbf{t})(i,j) = t_{i+j-2}$, $\mathbf{B}_n(\mathbf{t})(i,j) = t_{i+j-1}$, $\mathbf{C}_n(\mathbf{t})(i,j) = t_{i+j}$, for all $i, j \in \mathbb{Z}_+$ such that $1 \le i, j \le n+1$.

Proof. Theorems 3.3 and 3.4 in [25]. □

2.2.2 Alternating Direction Method of Multipliers (ADMM)

Consider the problem

$$\begin{aligned} p^* \ &= \min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \\ &\text{s.t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}, \end{aligned} \tag{6}$$

for some proper, closed, convex functions f and g, for some matrices $\mathbf{A}$ and $\mathbf{B}$, and a vector $\mathbf{c}$ with appropriate dimensions. ADMM iterations [26], [27] are given as follows:

$$\mathbf{x}^{k+1} \in \operatorname*{argmin}_{\mathbf{x}}\ f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}^k - \mathbf{c} + \mathbf{u}^k\|^2 \tag{7}$$

$$\mathbf{z}^{k+1} \in \operatorname*{argmin}_{\mathbf{z}}\ g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}^k\|^2 \tag{8}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}, \tag{9}$$

where $\rho > 0$ is a fixed algorithm parameter. Suppose $p^* > -\infty$, it is shown in [27] that under fairly general assumptions, the ADMM algorithm is well-defined and generates $\{\mathbf{x}^k, \mathbf{z}^k\}$ such that $f(\mathbf{x}^k) + g(\mathbf{z}^k) \to p^*$ and $\|\mathbf{A}\mathbf{x}^k + \mathbf{B}\mathbf{z}^k - \mathbf{c}\| \to 0$ as $k \to \infty$. Furthermore, the algorithm is fairly robust to minimization errors in (7) and (8), e.g., see [27], [32]. It is also worth noting that ADMM based algorithms are well suited for decentralized optimization, e.g., [30].

2.2.3 Primal-Dual Subgradient Method

Reference [28] proposes a subgradient method to compute approximate saddle points for convex-concave functions. Given closed convex sets $\mathcal{X}$ and $\mathcal{Y}$, consider

$$L^* \ = \ \min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{y}\in\mathcal{Y}} L(\mathbf{x}, \mathbf{y}), \tag{10}$$

for a convex-concave function $L : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. The method consists of the following iterations [28]:

$$\mathbf{x}^{k+1} \leftarrow \Pi_{\mathcal{X}}(\mathbf{x}^k - \alpha L_{\mathbf{x}}(\mathbf{x}^k, \mathbf{y}^k)) \tag{11}$$

$$\mathbf{y}^{k+1} \leftarrow \Pi_{\mathcal{Y}}(\mathbf{y}^k + \alpha L_{\mathbf{y}}(\mathbf{x}^k, \mathbf{y}^k)), \tag{12}$$

where $\alpha > 0$ is a given stepsize parameter, and $L_{\mathbf{x}}(\mathbf{x}^k, \mathbf{y}^k)$, $L_{\mathbf{y}}(\mathbf{x}^k, \mathbf{y}^k)$ denote some subgradients of L and $-L$ with respect to $\mathbf{x}$ and $\mathbf{y}$, respectively, at $(\mathbf{x}^k, \mathbf{y}^k)$. Suppose $L^* \in \mathbb{R}$, under some assumptions, reference [28] provides error bounds on the averaged function values, i.e., $\frac{1}{k}\sum_{i=0}^{k-1} L(\mathbf{x}^i, \mathbf{y}^i) - L^*$, and error bounds for the ergodic iterate sequence, i.e., $L(\hat{\mathbf{x}}^k, \hat{\mathbf{y}}^k) - L^*$, where $\hat{\mathbf{x}}^k = \frac{1}{k}\sum_{i=0}^{k-1} \mathbf{x}^i$ and $\hat{\mathbf{y}}^k$ is defined similarly.

3 Problem Formulation

This section introduces the network model for end-to-end traffic allocation. Furthermore, it presents the NUM problem formulation, and highlights the challenges associated with computing a solution to this problem.

3.1 Network Model

We consider a network with a set of sources sending data to their corresponding destinations. The network is represented by a set $\mathcal{L} = \{1, \ldots, L\}$ of directed links with finite positive capacities $\mathbf{c} = [c_l]_{l\in\mathcal{L}}$, which are shared by a set of sources $\mathcal{S} = \{1, \ldots, N\}$. Each source $i \in \mathcal{S}$ transmits data at rate $x_{i,p}$ along a predetermined path $p \in \mathcal{P}_i$, where $p \subset \mathcal{L}$ is a directed path consisting of a set of links that connect the source to its destination, and $\mathcal{P}_i$ is the set of all paths that can be used simultaneously by source i. Each source $i \in \mathcal{S}$ has a utility function $U_i : \mathbb{R}_+ \to \mathbb{R}_+$. The utility of source i, $U_i(r_i)$, is a function of its aggregate data rate transmitted over all possible paths, where $r_i = \sum_{p\in\mathcal{P}_i} x_{i,p}$. For each source i, let $\mathbf{x}_i = [x_{i,p}]_{p\in\mathcal{P}_i} \in \mathbb{R}_+^{|\mathcal{P}_i|}$ denote the vector of data rates over all paths in $\mathcal{P}_i$, and $\mathbf{r} = [r_i]_{i\in\mathcal{S}} \in \mathbb{R}_+^N$. Define the matrix $\mathbf{A}_i \in \mathbb{R}^{L\times|\mathcal{P}_i|}$ such that its (l,p)th entry equals 1 if path $p \in \mathcal{P}_i$ uses link $l \in \mathcal{L}$, and is 0 otherwise. The l-th row of $\mathbf{A}_i$ is denoted by $\mathbf{a}_i^l$ and is viewed as $|\mathcal{P}_i|$-dimensional column vector, i.e., $\mathbf{A}_i = [(\mathbf{a}_i^l)^\top]_{l\in\mathcal{L}}$, and $\mathbf{A} = [\mathbf{A}_1 \ldots \mathbf{A}_N]$. Let $\mathcal{P}_i^l = \{p \in \mathcal{P}_i : l \in p\}$ be the set of paths for the data of source i that use link $l \in \mathcal{L}$, $\mathcal{L}_i = \{l \in \mathcal{L} :\ \exists p \in \mathcal{P}_i \text{ s.t. } l \in p\}$ be the set of links used by at least one path of source i, and

$\mathcal{S}_l = \{i \in \mathcal{S} : \exists p \in \mathcal{P}_i \text{ s.t. } l \in p\}$ be the set of sources using link $l \in \mathcal{L}$. Note for each $i \in \mathcal{S}$, $\mathcal{P}_i^l$ is empty for all $l \notin \mathcal{L}_i$.

3.2 Problem Statement

We consider communication networks in which possibly *multiple* paths are available for the data of each source. The objective is to perform decentralized distribution of the traffic over all data paths available to sources in order to maximize the network utility function such that the traffic allocation satisfies the network resource constraints. In this work, the only network resources considered are the data links with given finite link capacities.

This work is applicable to network utility functions expressed as a sum of local user utilities. More precisely, we consider maximizing utility functions of the form

$$U(\mathbf{r}) = \sum_{i \in \mathcal{S}} U_i(r_i) \tag{13}$$

subject to network resource constraints and QoS guarantees. The traffic is allocated so that no single link in the network is congested. A link $l \in \mathcal{L}$ is said to be congested if the sum rate requested to be transmitted on that link exceeds its capacity. The network capacity constraints are

$$\sum_{i \in \mathcal{S}} \sum_{p \in \mathcal{P}_i^l} x_{i,p} \leq c_l, \quad l \in \mathcal{L}. \tag{14}$$

Furthermore, minimum QoS guarantees are considered at each source $i \in \mathcal{S}$ in the form of a lower bound on its aggregate data rate. Nevertheless, we also assume an upper bound on the data rate of each source for practical reasons. These lower and upper bounds on rates are enforced through the constraints $b_i \leq r_i \leq B_i$, for some $b_i, B_i \geq 0$ and all $i \in \mathcal{S}$. Thus, the optimal traffic allocation is obtained by solving the following problem:

$$\begin{aligned} \max_{\mathbf{x},\mathbf{r}} \quad & \sum_{i \in \mathcal{S}} U_i(r_i) \\ \text{s.t.} \quad & \sum_{i \in \mathcal{S}} \mathbf{A}_i \mathbf{x}_i \preceq \mathbf{c} \\ & (\mathbf{x}_i, r_i) \in \mathcal{X}_i, \quad i \in \mathcal{S}, \end{aligned} \tag{15}$$

where the set $\mathcal{X}_i$ is defined as

$$\mathcal{X}_i = \left\{(\mathbf{x}_i, r_i) \in \mathbb{R}_+^{|\mathcal{P}_i|} \times \mathbb{R}_+ : r_i = \mathbf{1}^\top \mathbf{x}_i, b_i \leq r_i \leq B_i\right\}. \tag{16}$$

Next, we list our assumptions on the NUM problem in (15).

Assumption 1. *The NUM problem (15) is feasible. Moreover, there exists at least one point $(\bar{\mathbf{x}}, \bar{\mathbf{r}})$ such that $\sum_{i \in \mathcal{S}} \mathbf{A}_i \bar{\mathbf{x}}_i \prec \mathbf{c}$ and $(\bar{\mathbf{x}}_i, \bar{r}_i) \in \mathcal{X}_i$ for all $i \in \mathcal{S}$.*

Assumption 2. *Each source $i \in \mathcal{S}$ has a possibly nonconcave utility function of its aggregate transmission rate, $U_i(r_i)$, and it is monotonically nondecreasing.*

Assumption 3. *For each source $i \in \mathcal{S}$, a finite number of paths to its destination is determined by a routing algorithm. However, source $i \in \mathcal{S}$ is oblivious to the capacities of the links forming those paths and does not know whether or not any other source is sharing the same links with itself.*

Assumption 4. *Each source $i \in \mathcal{S}$ has moderate computational capabilities to solve SDPs. However, intermediate nodes need not do computations beyond multiplication and addition.*

Most of the techniques developed in the literature for optimal decentralized traffic allocation allow only for concave diminishing reward utility functions. However, this work develops low-complexity distributed algorithms to compute an approximate solution to (15) while allowing for a generalized class of nonconcave user utility functions that can model user satisfaction in various real-time applications, e.g., video streaming [29]. In such applications, the solution of the resulting traffic allocation problem is an intricate task. The challenge associated with solving the optimization problem (15) is two-fold. First, (15) is nonconvex since we aim at maximizing a nonconcave objective function. Second, global network information is not assumed to be available to a centralized entity; a fact that stimulates the necessity of developing a decentralized algorithm.

4 Convex Relaxation

In this section, we outline the approach used to overcome the nonconvexity of the NUM problem (15). In this regard, we design a sequence of convex relaxations whose solutions converge to a point that characterizes an optimal solution of the original problem.

4.1 Nonconcave Utility Functions

A primary purpose of this work is to provide a traffic engineering method that works well under a wide variety of user utility functions. Instead of designing an algorithm that is finely tailored to a specific type of utility functions, e.g., step functions, we consider a general class of functions that can capture user utilities arising in diverse practical applications. The proposed approach relies on the ability to approximate general utility functions via a class of functions that can be tractably optimized without losing structural information encoded by those original utility functions. In particular, we propose to optimize a general class of monotonically nondecreasing, piecewise polynomial-like utility functions of the form:

$$\tilde{U}_i(r_i) = \sum_{w=1}^{\bar{w}_i} \left(\sum_{j=0}^{\ell} p_{i,j}^w r_i^{j/\ell} \right) 1_{\mathcal{I}_i^w}(r_i), \tag{17}$$

which can possibly be nonconcave. Recall that the range of interest for the variable r_i is the interval $[b_i, B_i]$. This interval is partitioned into $\bar{w}_i \in \mathbb{Z}_+$ nonoverlapping subintervals, where $_{\bar{w}_i}\mathcal{I}_i^w$ is a subinterval in that partition such that $\cup_{w=1}^{\bar{w}_i} \mathcal{I}_i^w = [b_i, B_i]$. For each w, a polynomial-like function is defined over $\mathcal{I}_i^w$ with coefficients $p_{i,j}^w$, where $0 \leq j \leq \ell$, for some $\ell \in \mathbb{Z}_+$. One can think of ℓ as resembling the degree of a polynomial. In the following lines, we explain the motivation behind proposing to optimize this particular form of utility functions. More precisely, we provide answers to the following questions: i) why a piecewise function ? and ii) why polynomial-like ?

Using piecewise functions opens room for a crafted approximation of many utility functions. It is well-known that any function can be approximated by a piecewise linear function, e.g., [31]. Evidently, U_i can be approximated by a

piecewise linear function that is monotonically nondecreasing and this class of functions is a special case of (17). For instance, for each i, if we set $p_{i,j}^w = 0$ for all $j \in \{0,\dots,\ell\}\backslash\{0,\ell\}$ and $w \in \{1,\dots,\bar{w}_i\}$, then the resulting function is piecewise linear over $[b_i, B_i]$. Thus, removing the restriction on $p_{i,j}^w$ to be zero for $0 < j < \ell$ and all w, i.e., considering a piecewise polynomial-like function, provides an approximation that is at least as good as that obtained by piecewise linear functions. Obviously, the higher the value of $\bar{w}_i$, the better the approximation. Nevertheless, as $\bar{w}_i$ increases, the complexity of the proposed traffic allocation algorithms increases. For a given utility function $U_i(r_i)$, efficient approximation techniques, e.g., SoS, can be used to calculate the coefficients $p_{i,j}^w$ that render (17) a close approximation to the true utility function according to some defined metric.

4.2 Relaxed Problem Formulation

We propose a sequence of convex relaxations whose solutions converge to a point that characterizes an optimal solution of (15) by leveraging results on the moment approach to polynomial optimization. Given parameters $s, \bar{s} \in \mathbb{Z}_+$ such that $\bar{s} \le s$, consider the following SDP:

$$\begin{aligned}
\max_{\mathbf{m},\mathbf{x},\mathbf{r}} \quad & \sum_{i\in\mathcal{S}} \sum_{w=1}^{\bar{w}_i} (\mathbf{p}_i^w)^\top \mathbf{m}_i^w \\
\text{s.t.} \quad & \sum_{w=1}^{\bar{w}_i} m_{i,0}^w = 1, \;\; i \in \mathcal{S} \\
& \mathbf{M}(\mathbf{m}_i^w) \succeq 0, \; \bar{\mathbf{M}}(\mathbf{m}_i^w) \succeq 0, \;\; w = 1,\dots,\bar{w}_i, \; i \in \mathcal{S} \\
& \sum_{w=1}^{\bar{w}_i} m_{i,j}^w \le r_i^{j/\ell}, \;\; j = 1,\dots,\bar{s}, \; i \in \mathcal{S} \\
& \sum_{i\in\mathcal{S}} \mathbf{A}_i \mathbf{x}_i \preceq \mathbf{c} \\
& (\mathbf{x}_i, r_i) \in \mathcal{X}_i, \;\; i \in \mathcal{S},
\end{aligned} \tag{18}$$

where the decision variables are $\mathbf{r}$, $\mathbf{x} = [\mathbf{x}_i]_{i\in\mathcal{S}}$, and $\mathbf{m} = [\mathbf{m}_i]_{i\in\mathcal{S}}$ such that $\mathbf{m}_i = [\mathbf{m}_i^w]_{w\in\{1,\dots,\bar{w}_i\}}$ for $\mathbf{m}_i^w = [m_{i,j}^w]_{j\in\{0,\dots,s\}}$, and the vector of coefficients of $\tilde{U}_i$, $\mathbf{p}_i^w = [p_{i,j}^w]_{j\in\{0,\dots,s\}}$, is defined such that if $s > \ell$, then $p_{i,j}^w = 0$ for $\ell < j \le s$. Moreover, let $\bar{\mathcal{I}}_i^w$ be a scaled version of $\mathcal{I}_i^w$ defined as follows: if $\mathcal{I}_i^w = [a, b]$ for some $a, b \in \mathbb{R}_+$, then $\bar{\mathcal{I}}_i^w = [a^{1/\ell}, b^{1/\ell}]$. The matrices $\mathbf{M}(\mathbf{m}_i^w)$ and $\bar{\mathbf{M}}(\mathbf{m}_i^w)$ are formed as in (2) and (3), respectively, when s is even; otherwise, as in (4) and (5), when s is odd; e.g., if $\bar{\mathcal{I}}_i^w = [a^{1/\ell}, b^{1/\ell}]$ and s is odd, i.e., $s = 2n+1$ for some $n \in \mathbb{Z}_+$, then

$$\begin{aligned}
\mathbf{M}(\mathbf{m}_i^w) &= b^{1/\ell}\mathbf{H}_n(\mathbf{m}_i^w) - \mathbf{B}_n(\mathbf{m}_i^w) \\
\bar{\mathbf{M}}(\mathbf{m}_i^w) &= \mathbf{B}_n(\mathbf{m}_i^w) - a^{1/\ell}\mathbf{H}_n(\mathbf{m}_i^w),
\end{aligned}$$

where $\mathbf{H}_n$ and $\mathbf{B}_n$ are constructed as shown in Theorem 2.

The following proposition states that near-optimal traffic allocations which approximately maximize the sum of local nonconcave user utility functions of the form (17), subject to network capacity constraints and QoS guarantees, can be obtained by solving (18) for sufficiently large $\bar{s} \in \mathbb{Z}_+$.

Proposition 1. *Given $\ell, s, \bar{s} \in \mathbb{Z}_+$, let $(\mathbf{m}^*(s), \mathbf{x}^*(s), \mathbf{r}^*(s))$ be an optimal solution to (18). Then, $(\mathbf{x}^*(s), \mathbf{r}^*(s)) \to (\mathbf{x}^*, \mathbf{r}^*)$ as $s, \bar{s} \to \infty$, where $(\mathbf{x}^*, \mathbf{r}^*)$ is an optimal solution to the nonconvex NUM problem in (15) with user utility functions as in (17), i.e., $U_i = \tilde{U}_i$ for $i \in \mathcal{S}$. Moreover, problem (18) is convex for $\bar{s} \le \ell$.*

Proof. See Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TNSE.2018.2851844. □

Roughly speaking, for more general $U = \sum_{i\in\mathcal{S}} U_i$ satisfying Assumption 2, by choosing $\ell = \bar{s} = s$ for sufficiently large $s \in \mathbb{Z}_+$, we can approximate (15) with a convex problem in (18). Indeed, $\tilde{U} = \sum_{i\in\mathcal{S}} \tilde{U}_i$ approximates U well, and (18) approximates $\max\{\tilde{U}(\mathbf{r}) : \sum_{i\in\mathcal{S}} \mathbf{A}_i\mathbf{x}_i \preceq \mathbf{c}, \; (\mathbf{x}_i, r_i) \in \mathcal{X}_i, \; i \in \mathcal{S}\}$. Therefore, Proposition 1 resolves the first challenge associated with solving the optimal traffic allocation problem which is dealing with nonconcave utility functions. In this setting, a sum of linear functions is maximized subject to convex constraints including linear matrix inequalities, i.e., (18) is a convex optimization problem. Therefore, (18) can be readily solved by a standard convex optimization algorithm if global network information is available. However, one of the main objectives of this work is to develop a decentralized traffic allocation algorithm that leverages local information available at each user and requires minimal information exchange through the network.

5 Centralized Traffic Allocation

In this section, we develop an iterative algorithm that converges to a solution of (18) with $\bar{s} \le \ell$. ADMM is applied to (18) resulting in a *centralized* traffic allocation algorithm, where global network information is assumed known at a central network entity. This algorithm represents a parent to the distributed algorithms proposed later, and acts as a benchmark against which we compare the performance of decentralized counterparts.

We now introduce some notation. Let

$$\begin{aligned}
\mathcal{K}_i = \Big\{ & (\mathbf{m}_i, \mathbf{x}_i, r_i) : \sum_{w=1}^{\bar{w}_i} m_{i,0}^w = 1, \sum_{w=1}^{\bar{w}_i} m_{i,j}^w \le r_i^{j/\ell}, j = 1,\dots,\bar{s}, \\
& \mathbf{M}(\mathbf{m}_i^w) \succeq 0, \bar{\mathbf{M}}(\mathbf{m}_i^w) \succeq 0, w = 1,\dots,\bar{w}_i, (\mathbf{x}_i, r_i) \in \mathcal{X}_i \Big\}
\end{aligned} \tag{19}$$

$$\mathcal{C} = \left\{ \mathbf{x} \in \mathbb{R}^{\sum_{i\in\mathcal{S}} |\mathcal{P}_i|} : \sum_{i\in\mathcal{S}} \mathbf{A}_i\mathbf{x}_i \preceq \mathbf{c} \right\}. \tag{20}$$

Then, (18) can be stated as

$$\begin{aligned}
\max_{\mathbf{m},\mathbf{x},\mathbf{r}} \quad & \sum_{i\in\mathcal{S}} \mathbf{p}_i^\top \mathbf{m}_i \\
\text{s.t.} \quad & (\mathbf{m}_i, \mathbf{x}_i, r_i) \in \mathcal{K}_i, \; i \in \mathcal{S} \\
& \mathbf{x} \in \mathcal{C},
\end{aligned} \tag{21}$$

where $\mathbf{p}_i = [\mathbf{p}_i^w]_{w\in\{1,\dots,\bar{w}_i\}}$. Among the advantages of the proposed convex relaxation (21) for the NUM problem is that it is amenable to decentralized computation due to its separable structure. By examining (21), we notice that:

- The variables $\mathbf{m}_i$ and r_i are local to the ith source and need not be broadcast to any other node.
- The objective function is a sum of linear functions of the local variables.

- The structure of $\mathcal{K}_i$ implies that the constraint $(\mathbf{m}_i, \mathbf{x}_i, r_i) \in \mathcal{K}_i$ is a local constraint to the ith source that can be handled locally.
- The only constraint that forces interaction among the sources is the network capacity constraints.

Next, we introduce a new variable $\mathbf{z} = [\mathbf{z}_i]_{i\in\mathcal{S}}$, where $\mathbf{z}_i \in \mathbb{R}^{|\mathcal{P}_i|}$, to obtain an equivalent formulation to (21):

$$\begin{aligned} \max_{\mathbf{m},\mathbf{x},\mathbf{r},\mathbf{z}} \quad & \sum_{i\in\mathcal{S}} \mathbf{p}_i^\top \mathbf{m}_i \\ \text{s.t.} \quad & (\mathbf{m}_i, \mathbf{x}_i, r_i) \in \mathcal{K}_i,\ i \in \mathcal{S} \\ & \mathbf{z}_i = \mathbf{x}_i,\ i \in \mathcal{S} \\ & \mathbf{z} \in \mathcal{C}, \end{aligned} \tag{22}$$

which is suitable for applying ADMM. ADMM is a primal-dual algorithm; hence, it requires updating both primal and dual variables. Note that primal update can be distributed among the source nodes in $\mathcal{S}$. In particular, we consider two ADMM blocks, namely, $(\mathbf{m}, \mathbf{x}, \mathbf{r})$ and $\mathbf{z}$. Then, customizing the ADMM update rules in (7), (8), and (9) leads to the centralized traffic allocation algorithm (CTAA) outlined in Algorithm 1, where as shown in step 3 of Algorithm 1, updating the primal variables requires solving N independent optimization problems in parallel, with each source node solving one locally. A detailed mathematical derivation of CTAA is presented in our earlier paper [1].

Algorithm 1. CTAA
($\rho > 0$, $\mathbf{z}^0$, $\mathbf{u}^0 = [\mathbf{u}_i^0]_{i\in\mathcal{S}}$)

1 Initialize $\mathbf{z}^0$, $\mathbf{u}^0$.
2 **for** $k = 0, 1, \ldots$ **do**
3 For each source $i \in \mathcal{S}$:
$(\mathbf{m}_i, \mathbf{x}_i, r_i)^{k+1} \leftarrow \operatorname{argmax}\ \mathbf{p}_i^\top \mathbf{m}_i - \frac{\rho}{2}\left\|\mathbf{x}_i - \mathbf{z}_i^k + \mathbf{u}_i^k\right\|^2$
s.t. $(\mathbf{m}_i, \mathbf{x}_i, r_i) \in \mathcal{K}_i$.
4 Central node computes actual transmission rates:
$\mathbf{z}^{k+1} \leftarrow \Pi_{\mathcal{C}}(\mathbf{x}^{k+1} + \mathbf{u}^k)$
5 For each source $i \in \mathcal{S}$:
$\mathbf{u}_i^{k+1} \leftarrow \mathbf{u}_i^k + \mathbf{x}_i^{k+1} - \mathbf{z}_i^{k+1}$.

We now proceed with the description of the algorithm's operation. The algorithm parameter ρ can be any positive scalar. The vectors $\mathbf{z}$ and $\mathbf{u} = [\mathbf{u}_i]_{i\in\mathcal{S}}$ can be arbitrarily initialized, where $\mathbf{u}_i \in \mathbb{R}^{|\mathcal{P}_i|}$. The vector $\mathbf{x}_i$ stores the *desired* transmission rates of source i over $\mathcal{P}_i$; on the other hand, $\mathbf{z}_i$ denotes the *actual* rate vector that source i transmits throughout Algorithm 1's iterations. The constraint $\mathbf{z}_i = \mathbf{x}_i$ is not satisfied for every iteration. Nevertheless, both $\mathbf{z}_i$ and $\mathbf{x}_i$ eventually converge to a consensus as the algorithm keeps running. Each source $i \in \mathcal{S}$ keeps the vectors $\mathbf{p}_i$, $\mathbf{m}_i$, $\mathbf{x}_i$, $\mathbf{z}_i$, and $\mathbf{u}_i$ as local information. Moreover, the structure of the set $\mathcal{K}_i$ is known only at source i for every $i \in \mathcal{S}$, i.e., (19) indicates that $\mathcal{K}_i$ is fully characterized by local information available at source i such as the lower and upper bounds imposed on its own data rate. The kth iteration of CTAA consists of the following operations:

i) In step 3, each source $i \in \mathcal{S}$ updates its desired rates $\mathbf{x}_i$ by solving an SDP using local information. This step is carried out in parallel across all sources.
ii) Each source $i \in \mathcal{S}$ transmits its local variables $\mathbf{x}_i$ and $\mathbf{u}_i$ to a central network node.
iii) The actual transmission rates of all sources, $\mathbf{z} = [\mathbf{z}_i]_{i\in\mathcal{S}}$, are updated in step 4 through a projection operation onto the set $\mathcal{C}$ performed by the central network node.
iv) The central node sends the updated $\mathbf{z}$ vector to the source nodes, i.e., source i receives its updated $\mathbf{z}_i$.
v) Each source $i \in \mathcal{S}$ updates the vector $\mathbf{u}_i$ locally as in step 5. This step is carried out in parallel and independently across all sources.

Even though Algorithm 1 decomposes the large dimensional SDP in (21) into separate moderate size SDPs that can be locally handled by each source in parallel, its implementation requires the presence of a central network entity to perform the $\mathbf{z}$-update step. step 4 ensures that the actual transmission rates of all sources conform to network capacity constraints via solving the Euclidean projection problem $\Pi_{\mathcal{C}}(\mathbf{x}^{k+1} + \mathbf{u}^k)$. Let $\bar{\mathbf{z}}_i^k = \mathbf{x}_i^{k+1} + \mathbf{u}_i^k$ and $\bar{\mathbf{z}}^k = [\bar{\mathbf{z}}_i^k]_{i\in\mathcal{S}}$, then $\Pi_{\mathcal{C}}(\bar{\mathbf{z}}^k)$ entails solving the quadratic program (QP):

$$\mathbf{z}_*^k = \operatorname*{argmin}_{\mathbf{z}} \left\{ \frac{1}{2} \sum_{i\in\mathcal{S}} \|\mathbf{z}_i - \bar{\mathbf{z}}_i^k\|^2 :\ \sum_{i\in\mathcal{S}} (\mathbf{a}_i^l)^\top \mathbf{z}_i \le c_l,\ l \in \mathcal{L} \right\}. \tag{23}$$

Solving (23) requires the central node to know all the routes source i uses, i.e., matrix $\mathbf{A}_i$, for each $i \in \mathcal{S}$ and the capacity of every link $l \in \mathcal{L}$, i.e., $\mathbf{c}$. Given this information, (23) is a simple convex quadratic problem, which can be solved by a standard convex optimization algorithm for convex QPs. We conclude that a direct implementation of ADMM exemplifies a centralized solution method to the NUM problem with considerable communication overhead. Next, we propose decentralized traffic allocation algorithms based on *inexact* versions of ADMM.

6 Distributed Traffic Allocation

We propose three different distributed traffic allocation algorithms (DTAA) based on ADMM, namely DTAA-RFA, DTAA-RFC, and DTAA-BFC. Eckstein and Yao show in [32] that it is possible to obtain a variant of ADMM in which at least one of the subproblems in (7) or (8) requires an iterative solution method. In other words, under some conditions, computing *inexact* solutions to ADMM subproblems suffices to retain the overall convergence of the algorithm. It is evident that step 4 of CTAA is precisely the reason why a central network entity is needed. Inspired by the insight in [32], we propose iterative solutions for (23) that opens room for a distributed implementation of a nearly-optimal traffic allocation algorithm. Here, we highlight the implementation characteristics of the proposed algorithms without dense mathematical details that might affect the clarity of exposition. Nevertheless, the derivation of DTAA-RFA, DTAA-RFC, and DTAA-BFC are presented in Appendix C, D, and E, respectively, available in the online supplemental material. In Appendix B, available in the online supplemental material, we derive theoretical convergence guarantees for DTAA-RFA and DTAA-RFC.

6.1 DTAA-RFA

DTAA-RFA combines CTAA with an iterative solution for (23) based on the dual ascent method [33], where RFA stands for real-valued feedback from all links. Algorithm 2 outlines how DTAA-RFA solves (22). The parameters ρ and α are positive scalars, and the sequence $\{\tau^k\}_{k\in\mathbb{Z}_+} \subset \mathbb{Z}_+$ is an

increasing sequence of positive integers, i.e., $\tau^{k+1} \geq \tau^k$ for all $k \in \mathbb{Z}_+$. The scalar λ_l is a variable stored at link node l, and $\lambda = [\lambda_l]_{l\in\mathcal{L}}$. By link node l, we mean either one of the endpoints of the link l, i.e., if a link l connects nodes u and v, either u or v is called the link node l. Source nodes, $\mathcal{S}$, do not share their local information with any other network entity. Steps 3 and 10 in DTAA-RFA are similar to steps 3 and 5 in CTAA. On the other hand, in contrast to step 4 in CTAA, for every outer iteration k in DTAA-RFA, there exist τ^k inner iterations indexed by n intended for updating the actual transmission rates $\mathbf{z}$. In particular, at outer iteration k, the nth inner iteration of DTAA-RFA consists of the following:

i) Each link node $l \in \mathcal{L}$ sends its local variable $\lambda_l^{k,n}$ to the set of sources using link l, i.e., $\mathcal{S}_l$.
ii) Each source $i \in \mathcal{S}$ updates its transmission rate vector as in step 7 using the feedback information it receives from the links used by its paths.
iii) Each link node $l \in \mathcal{L}$ updates its stored variable $\lambda_l^{k,n}$ as in step 8.

It is important to note that the inner iterations are executed in *parallel* across source and link nodes. A source node updates its transmission rate $z_{i,p}^{k,n}$ along path $p \in \mathcal{P}_i$ using its updated desired transmission rate $x_{i,p}^{k+1}$, its local variable $u_{i,p}^k$, and the quantity $\sum_{l\in p} \lambda_l^{k,n}$, i.e., the sum of $\lambda_l^{k,n}$ over the links l used by the path p. Thus, only the sum for each path $p \in \mathcal{P}_i$ is expected to be fed back to the source i. On the other hand, a link node l updates its stored variable $\lambda_l^{k,n}$ using the difference between the sum data rate transmitted by all sources over that link and the link's capacity. Evidently, the link node knows the sum rate over the link that it is connected to, and needs to know this link's capacity. These observations along with steps 3 and 10 imply that DTAA-RFA provides distributed optimal traffic allocation, where all the computations are performed in parallel independently at network nodes using local information and real-valued feedback from link nodes.

Algorithm 2. DTAA-RFA
(ρ, α, $\{\tau^k\}_{k\in\mathbb{Z}_+}$, $\mathbf{z}^0$, $\mathbf{u}^0$, $\lambda^0 = [\lambda_l^0]_{l\in\mathcal{L}}$)

1 Initialize $\mathbf{z}^0$, $\mathbf{u}^0$, λ^0.
2 **for** $k = 0, 1, \ldots$ **do**
3 For each source $i \in \mathcal{S}$:
$(\mathbf{m}_i, \mathbf{x}_i, r_i)^{k+1} \leftarrow \operatorname{argmax} \mathbf{p}_i^\top \mathbf{m}_i - \frac{\rho}{2}\left\|\mathbf{x}_i - \mathbf{z}_i^k + \mathbf{u}_i^k\right\|^2$ s.t. $(\mathbf{m}_i, \mathbf{x}_i, r_i) \in \mathcal{K}_i$.
4 Initialize $\lambda^{k,1} \leftarrow \lambda^0$.
5 **for** $n = 1, \ldots, \tau^k$ **do**
6 Each link $l \in \mathcal{L}$ sends $\lambda_l^{k,n}$ to all $i \in \mathcal{S}_l$.
7 Each source $i \in \mathcal{S}$ transmits with rate vector $\mathbf{z}_i^{k,n} \leftarrow \mathbf{x}_i^{k+1} + \mathbf{u}_i^k - \sum_{l\in\mathcal{L}_i} \lambda_l^{k,n}\mathbf{a}_i^l$.
8 For each link $l \in \mathcal{L}$:
$\lambda_l^{k,n+1} \leftarrow \left(\lambda_l^{k,n} + \alpha\left[\sum_{i\in\mathcal{S}}(\mathbf{a}_i^l)^\top \mathbf{z}_i^{k,n} - c_l\right]\right)^+$.
9 $\mathbf{z}_i^{k+1} \leftarrow \mathbf{z}_i^{k,\tau^k}$ for all $i \in \mathcal{S}$.
10 $\mathbf{u}_i^{k+1} \leftarrow \mathbf{u}_i^k + \mathbf{x}_i^{k+1} - \mathbf{z}_i^{k+1}$ for all $i \in \mathcal{S}$.

6.2 DTAA-RFC

DTAA-RFC combines CTAA with an iterative solution for (23) based on the primal dual subgradient method [28], where RFC stands for real-valued feedback from *congested* links. Algorithm 3 outlines how DTAA-RFC solves (22). The parameters of the algorithm are similar to those of DTAA-RFA – DTAA-RFC uses a diminishing sequence of positive scalars $\{\alpha^k\}$ as opposed to a constant $\alpha > 0$ in DTAA-RFA.

Algorithm 3. DTAA-RFC
(ρ, $\{\alpha^k\}_{k\in\mathbb{Z}_+}$, $\{\tau^k\}_{k\in\mathbb{Z}_+}$, $\mathbf{z}^0$, $\mathbf{u}^0$, λ^0)

1 Initialize $\mathbf{z}^0$, $\mathbf{u}^0$, λ^0.
2 **for** $k = 0, 1, \ldots$ **do**
3 For each source $i \in \mathcal{S}$:
$(\mathbf{m}_i, \mathbf{x}_i, r_i)^{k+1} \leftarrow \operatorname{argmax} \mathbf{p}_i^\top \mathbf{m}_i - \frac{\rho}{2}\left\|\mathbf{x}_i - \mathbf{z}_i^k + \mathbf{u}_i^k\right\|^2$ s.t. $(\mathbf{m}_i, \mathbf{x}_i, r_i) \in \mathcal{K}_i$.
4 Set $\lambda^{k,0} \leftarrow \lambda^0$, $\mathbf{z}^{k,0} \leftarrow \mathbf{z}^k$, $\bar{\mathbf{z}}^k \leftarrow \mathbf{x}^{k+1} + \mathbf{u}^k$.
5 **for** $n = 0, \ldots, \tau^k - 1$ **do**
6 Each source $i \in \mathcal{S}$ transmits data at rate $z_{i,p}^{k,n}$ along path $p \in \mathcal{P}_i$.
7 Each link $l \in \mathcal{L}_c^{k,n}$ sends $\lambda_l^{k,n}$ to all $i \in \mathcal{S}_l$.
8 Each source $i \in \mathcal{S}$ updates its rate vector $\mathbf{z}_i^{k,n+1} \leftarrow \Pi_{\mathcal{Z}^k}((1-\alpha^k)\mathbf{z}_i^{k,n} + \alpha^k(\bar{\mathbf{z}}_i^k - \sum_{l\in\mathcal{L}_c^{k,n}} \lambda_l^{k,n}\mathbf{a}_i^l))$.
9 For each link $l \in \mathcal{L}_c^{k,n}$:
$\lambda_l^{k,n+1} \leftarrow \Pi_{\mathcal{D}^k}(\lambda_l^{k,n} + \alpha^k(\sum_{i\in\mathcal{S}}(\mathbf{a}_i^l)^\top \mathbf{z}_i^{k,n} - c_l)^+)$.
10 $\mathbf{z}_i^{k+1} \leftarrow \frac{1}{\tau^k}\sum_{n=0}^{\tau^k-1} \mathbf{z}_i^{k,n}$ for all $i \in \mathcal{S}$.
11 $\mathbf{u}_i^{k+1} \leftarrow \bar{\mathbf{z}}_i^k - \mathbf{z}_i^{k+1}$ for all $i \in \mathcal{S}$.

Let $\mathcal{Z}^k = \{\mathbf{z} : \|\mathbf{z} - \mathbf{x}^{k+1} - \mathbf{u}^k\|_\infty \leq \eta^k\}$, and $\mathcal{D}^k = \{\lambda : 0 \preceq \lambda \preceq \zeta^k\}$ for some η^k and ζ^k given in Appendix D, available in the online supplemental material. Steps 3 and 11 in DTAA-RFC are similar to steps 3 and 5 in CTAA. At outer iteration k, the nth inner iteration of DTAA-RFC consists of the following:

i) Each $i \in \mathcal{S}$ transmits at rate $z_{i,p}^{k,n}$ along path $p \in \mathcal{P}_i$.
ii) A link l that belongs to the set of congested links $\mathcal{L}_c^{k,n} = \{l : \sum_{i\in\mathcal{S}}(\mathbf{a}_i^l)^\top \mathbf{z}_i^{k,n} \geq c_l\}$ sends its local variable $\lambda_l^{k,n}$ to all $i \in \mathcal{S}_l$.
iii) Each source $i \in \mathcal{S}$ uses the feedback information it receives from the congested links used by its paths to update its rate vector $\mathbf{z}_i$ as in step 8.
iv) Each link $l \in \mathcal{L}_c^{k,n}$ updates its variable $\lambda_l^{k,n}$ using the sum rate on the link and its capacity as in step 9.

The inner iterations are executed in parallel across all source and link nodes. The variable updates done at link nodes require local information, specifically, the sum data rate transmitted on the link and its capacity which are assumed known to the node connected to this link. In contrast to DTAA-RFA that requires feedback from all links, DTAA-RFC only requires congested links to update their variables and feed them back to the sources using these links. Source nodes update their transmission rates locally and independently and need not broadcast any of their local information or share it with any network entity. Hence, DTAA-RFC indeed provides distributed traffic allocation.

6.3 DTAA-BFC

DTAA-BFC combines CTAA with an iterative solution for (23) based on a dual subgradient method applied to a penalty formulation of the projection problem, where BFC stands for binary feedback from congested links. Algorithm 4 outlines how DTAA-BFC solves (22). The parameters ρ and $\{\tau^k\}_{k\in\mathbb{Z}_+}$ are similar to those of DTAA-RFA. $\{\alpha^n\}_{n\in\mathbb{Z}_+}$ and $\{\theta^k\}_{k\in\mathbb{Z}_+} \subset \mathbb{R}_{++}$ are some positive scalar sequences. Steps 3 and 11 in DTAA-BFC are similar to steps 3 and 5 in

CTAA. At outer iteration k, the nth inner iteration of DTAA-BFC consists of the following operations:

i) Each source $i \in \mathcal{S}$ transmits data over its paths $\mathcal{P}_i$ according to a rate vector $\mathbf{z}_i^{k,n}$.
ii) Any congested link $l \in \mathcal{L}$ sends $b_l^{k,n}$ to the set of sources using that link, i.e., $\mathcal{S}_l$. The binary feedback bit $b_l^{k,n}$ determines the status of link $l \in \mathcal{L}$, and is sent by one or both of the nodes connected by l.
iii) Each source $i \in \mathcal{S}$ updates its actual data rate vector $\mathbf{z}_i$ using its local information and the received binary feedback as in step 9.

DTAA-BFC provides a near-optimal traffic allocation using binary feedback only, and such that no computations are required at link nodes and that all source computations are performed in parallel independently at each source node and need not be broadcast or shared with any network entity. Furthermore, each source node uses local information and the only non-local information needed is binary feedback from congested links.

Algorithm 4. DTAA-BFC
$(\rho, \{\alpha^n\}_{n\in\mathbb{Z}_+}, \{\tau^k\}_{k\in\mathbb{Z}_+}, \{\theta^k\}_{k\in\mathbb{Z}_+}, \mathbf{z}^0, \mathbf{u}^0)$

1 Initialize $\mathbf{z}^0$, $\mathbf{u}^0$.
2 **for** $k = 0, 1, \ldots$ **do**
3 For each source $i \in \mathcal{S}$:
$(\mathbf{m}_i, \mathbf{x}_i, r_i)^{k+1} \leftarrow \operatorname{argmax} \mathbf{p}_i^\top \mathbf{m}_i - \frac{\rho}{2}\left\|\mathbf{x}_i - \mathbf{z}_i^k + \mathbf{u}_i^k\right\|^2$ s.t. $(\mathbf{m}_i, \mathbf{x}_i, r_i) \in \mathcal{K}_i$.
4 Set $\mathbf{z}^{k,1} \leftarrow \mathbf{z}^k$, $\bar{\mathbf{z}}^k \leftarrow \mathbf{x}^{k+1} + \mathbf{u}^k$.
5 **for** $n = 1, \ldots, \tau^k - 1$ **do**
6 For each source $i \in \mathcal{S}$ and for each $p \in \mathcal{P}_i$:
Source i transmits at rate $\mathbf{z}_{i,p}^{k,n}$ along path p.
7 For each link $l \in \mathcal{L}$:
if link l is congested, **then** $b_l^{k,n} \leftarrow 1$;
else $b_l^{k,n} \leftarrow 0$.
8 Each link $l \in \mathcal{L}$ sends one bit $b_l^{k,n}$ to all $i \in \mathcal{S}_l$.
9 For each source $i \in \mathcal{S}$:
$\mathbf{g}_i^{k,n} \leftarrow \mathbf{z}_i^{k,n} - \bar{\mathbf{z}}_i^k + \theta^k \sum_{l\in\mathcal{L}} b_l^{k,n} \mathbf{a}_i^l$.
$\mathbf{z}_i^{k,n+1} \leftarrow (\mathbf{z}_i^{k,n} - \alpha^n \mathbf{g}_i^{k,n})^+$.
10 $\mathbf{z}_i^{k+1} \leftarrow \mathbf{z}_i^{k,\tau^k}$ for all $i \in \mathcal{S}$.
11 $\mathbf{u}_i^{k+1} \leftarrow \bar{\mathbf{z}}_i^k - \mathbf{z}_i^{k+1}$ for all $i \in \mathcal{S}$.

7 Numerical Simulations

This section presents an application of the distributed algorithms developed in this paper. Numerical simulations are conducted on a large-scale network to validate our findings. In particular, the main objectives of this section are summarized as follows.

- We show that in practice DTAA-RFA, DTAA-RFC, and DTAA-BFC rate allocations converge to traffic allocations that yield the same utility function value obtained by CTAA.
- We demonstrate the robustness of the proposed algorithms to sudden link failures, and emphasize their scalability by showing their ability to automatically accommodate new users joining the network.

We generate a random connected graph consisting of 500 intermediate nodes, where the probability of having an edge connecting any pair of nodes is 0.04. We then add 500 source/destination pairs, i.e., the total number of vertices in the network is 1500. Each source node's degree is 1 and its single neighbor is randomly chosen from the intermediate nodes. Each pair of sources is connected to one intermediate node. A similar setup holds for the destinations. No direct connections are allowed between any two sources, any two destinations, or between a source and a destination. More precisely, the set of sources and destinations form an independent set, i.e., they are pairwise non-adjacent. The network model considered allows for multiple paths to be available for the data of each source. For each $i \in \mathcal{S}$, we pick 10 unique paths between source i and its destination, i.e., $|\mathcal{P}_i| = 10$ for all $i \in \mathcal{S}$. Path $p \in \mathcal{P}_i$ is generated by running a randomized depth-first search (DFS) algorithm on the graph starting at source i and terminating when destination i is discovered. A unique identifier is given to each link in the network, and the generated path $p \in \mathcal{P}_i$ is mapped to a column of binary entries in the matrix $\mathbf{A}_i$. The capacity of each link is randomly assigned a value drawn from a uniform distribution on the interval $[2, 4]$ Mbps. We show simulation results for utility functions given by

$$U_i(r_i) = \begin{cases} 0.4r_i, & \text{if } 0 \le r_i < 0.5 \\ r_i - 0.3, & \text{if } 0.5 \le r_i < 1.5 \\ 0.5r_i + 0.45, & \text{if } 1.5 \le r_i \le 3 \end{cases} \tag{24}$$

for all $i \in \mathcal{S}$. Specifically, for the utility function given by (24), we have $b_i = 0$, and $B_i = 3$ Mbps, and $[0, 3]$ is partitioned to $\bar{w}_i = 3$ segments: $\mathcal{I}_i^1 = [0, 0.5)$, $\mathcal{I}_i^2 = [0.5, 1.5)$, and $\mathcal{I}_i^3 = [1.5, 3]$. We choose $\bar{s} = \ell = 2$ and $s = 8$; hence, $\mathbf{p}_i^1 = [0\ 0\ 0.4]^\top$, $\mathbf{p}_i^2 = [-0.3\ 0\ 1]^\top$, and $\mathbf{p}_i^3 = [0.45\ 0\ 0.5]^\top$. We now proceed with outlining the choice of the parameters of the proposed distributed algorithms. For DTAA-RFA, we set $\rho = 1$, $\alpha = 1/\|\mathbf{A}\|^2$, and $\tau^k = 100$ for all $k \in \mathbb{Z}_+$. On the other hand, DTAA-RFC uses $\rho = 1$, $\alpha^k = 0.02$, and $\tau^k = 500$ for all $k \in \mathbb{Z}_+$. Finally, for DTAA-BFC, $\rho = 1$, $\tau^k = 250$ and $\theta^k = 1$ for $k \in \mathbb{Z}_+$, and $\alpha^n = 1/n$.

The convex relaxation (18) is solved using CTAA assuming the availability of global network information. CTAA serves as a benchmark with which we compare the performance of the proposed distributed algorithms. However, we emphasize that CTAA is a centralized solution that is not practical for implementation on large-scale networks. Fig. 1 shows that all proposed distributed algorithms yield traffic allocations which attain network utility function values that are almost indistinguishable from the one obtained by the centralized algorithm, CTAA. Although no global network information is used by the distributed algorithms, they converge to the same utility function value obtained by CTAA that uses global network information. As a side comment, at the initialization, the utility function value may exceed the optimal one due to infeasible traffic allocations caused by the arbitrary initialization of the algorithms. In general, as also pointed out in the appendices, the consensus constraint $\mathbf{x} = \mathbf{z}$ in (22) need not be satisfied for every iteration of the algorithm, furthermore, the capacity constraints enforced in CTAA through the projection problem onto the set $\mathcal{C}$ need not be resolved exactly in the distributed algorithms; hence, infeasible rate allocations can occur before the convergence. Nonetheless, the distributed algorithms eventually converge to feasible allocations that yield the optimal utility function value obtained by CTAA as shown in Fig. 1.

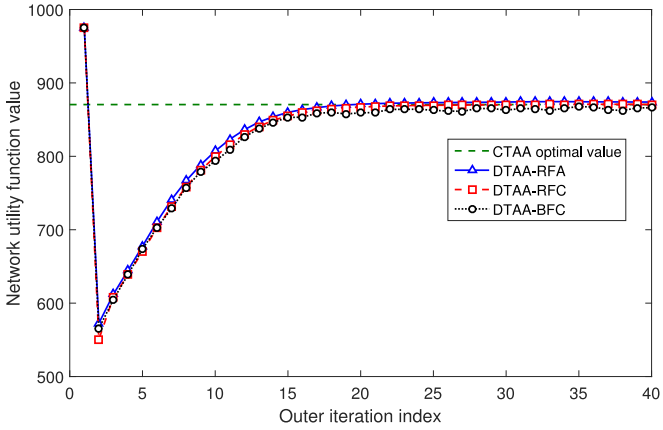

Fig. 1. Performance comparison of the proposed distributed algorithms to CTAA.

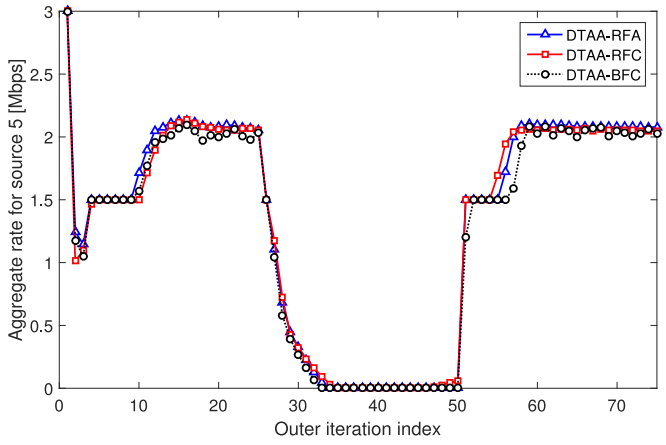

Fig. 3. Data rate allocation for source 5.

Finally, Figs. 2 and 3 show that the proposed distributed algorithms are robust to sudden link failures and they automatically scale out to accommodate new users joining the network. This feature is attributed to the adaptive nature of updating the data rates. When a link failure is detected, the algorithms reroute the traffic such that routes using that link are avoided. The distributed algorithms handle a link failure by treating it as congestion, i.e., the source nodes are oblivious to the failure and require no additional information other than the usual required feedback. Figs. 2 and 3 show the trajectories of the aggregate data rate allocations obtained by all distributed algorithms for sources 1 and 5, respectively. The two links connecting destinations 1 and 5 to their respective unique neighbors fail after 25 outer iterations of running the algorithm and recover after 50 outer iterations. These particular links are chosen to fail since their failure implies that sources 1 and 5 are disconnected from their destinations. Thus, their failure results in a considerable performance degradation for sources 1 and 5, and their recovery opens room for checking if the algorithm is capable of accommodating new users joining the network. The figures show that the distributed algorithms quickly react to both the failure and recovery of the links. When the links fail, the data of sources 1 and 5 cannot reach their respective destinations and hence, the algorithms are shown to automatically shut down transmission from these sources, i.e., r_1 and r_5 approach zero. On the other hand, when the links recover, rates converge to their optimal values. It is worth mentioning that there might be multiple optimal rate allocations and hence, all the algorithms need not converge to the same traffic allocation.

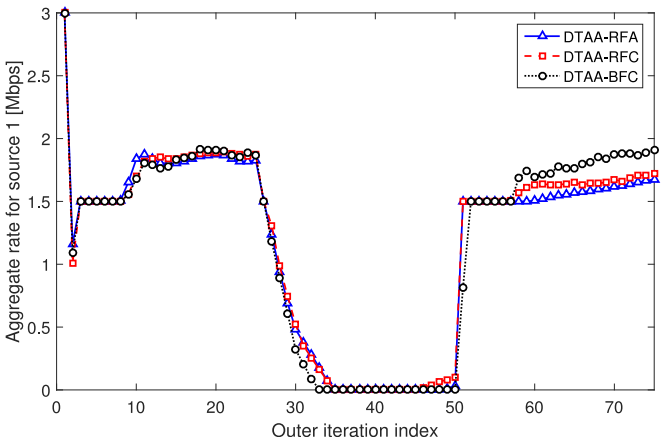

Fig. 2. Data rate allocation for source 1.

8 Discussion

This paper addresses the optimization of network utility functions that can be expressed as a sum of local user utilities. In practical applications, a nonconcave utility function is a better model of the user-perceived QoE. Hence, unlike most of the work in related literature, we focus on the case in which the utility of each user is a nonconcave function of its aggregate data rate. The two main contributions of this paper are: i) designing a sequence of convex relaxations whose solutions converge to a point that characterizes an optimal solution of the original nonconvex NUM problem, and ii) developing decentralized algorithms that attain an optimal solution of such convex relaxations.

8.1 Approach

The approach adopted in this paper relies mainly on two kind of approximations. First, the approximation of general utility functions U_i by piecewise polynomial-like functions $\tilde{U}_i$. Second, the approximation of the solution of the original nonconvex problem (15) by the solution of the convex relaxation (18). Pertaining to the accuracy of these approximations, one chooses four design parameters, namely, $\bar{w}_i$ for each $i \in \mathcal{S}$, ℓ, s, and $\bar{s}$.

The parameters $\bar{w}_i$ and ℓ influence the accuracy of the utility function approximation, i.e., how closely $\tilde{U}_i$ approximates U_i. More precisely, larger values of $\bar{w}_i$ leads to finer partitioning of the interval $[b_i, B_i]$, and larger values of ℓ introduce more monomial-like terms $r^{j/\ell}$ and hence, allow for optimizing over a larger space of coefficients $p^w_{i,j}$ in the utility function approximation process. On the other hand, the parameters s and $\bar{s}$ affect how close the solution of (18) is to that of (15). Indeed, $\bar{s}$ pertains to the accuracy of linearizing $\tilde{U}_i$ in (17) via the change of variables $y_i = r_i^{1/\ell}$, for details see Appendix A, available in the online supplemental material. In summary, for arbitrary approximation accuracy, one needs $\bar{w}_i$, ℓ, s, and $\bar{s}$ to go to infinity. However, one practically chooses finite values for these parameters that provide satisfactory performance, as shown in Section 7.

8.2 Complexity of the Algorithms

There are two aspects regarding the complexity of the proposed algorithms: i) the per-node computational complexity, and ii) the network communication overhead. In terms of the computational complexity, we note the following:

- All the computational burden is moved to the sources, whereas intermediate nodes are only required to forward link variables' feedback without doing any computations, except for summation and simple thresholding.
- All algorithms require each source i to solve an SDP. Remarkably, the dimension of such SDP does not directly depend on the scale of the network, yet it depends on the number of paths between a source and its destination and on $\bar{s}$, s and $\bar{w}_i$.
- As shown in [34], most interior point methods designed to solve linear programs have been generalized to SDPs with polynomial worst-case complexity.

It is clear that all the proposed algorithms share the same ADMM parent algorithm and only differ in their inner loops executed per ADMM iteration. These loops aim at providing a distributed approximate solution to the projection problem in Step 4 of CTAA. In all algorithms, messages are passed from intermediate nodes to the sources in each inner iteration resulting in a communication overhead. There are two factors affecting the amount of overhead generated by each algorithm: i) the number of messages passed per inner iteration, and ii) the number of inner iterations required by each algorithm to retain the overall convergence of the inexact ADMM algorithm. A unified bound on the number of messages to be fed back from link nodes to the sources per inner iteration is $\mathcal{O}(|\mathcal{V}|^2)$, where $\mathcal{V}$ is the set of vertices in the network. This is attributed to the fact that the worst case scenario dictates that all links feed back their stored λ variables to the sources, and the number of edges in a graph with a set of vertices $\mathcal{V}$ is $\mathcal{O}(|\mathcal{V}|^2)$. However, the amount of overhead generated by each algorithm significantly differs from one another as shown in the following comparisons.

8.2.1 DTAA-RFA versus DTAA-RFC

DTAA-RFA requires all links $l \in \mathcal{L}$ to feed their stored variables λ_l back to the set of sources using them in every inner iteraton. In contrast, DTAA-RFC requires such feedback from congested links only, where the number of congested links is expected to be significantly less than $|\mathcal{L}|$. Thus, the overhead generated by DTAA-RFA per inner iteration is larger than that produced by DTAA-RFC. That said, Theorem 5 in Appendix C, available in the online supplemental material shows that the inner loop iterates of DTAA-RFA, $\{\mathbf{z}^{k,n}\}_n$, satisfies $\|\mathbf{z}^{k,n} - \mathbf{z}_*^k\|^2 = \mathcal{O}(1/n)$, where $\mathbf{z}_*^k$ is the solution to Step 4 in CTAA, and n is the number of inner iterations. In contrast, Theorem 7 in Appendix D, available in the online supplemental material indicates that DTAA-RFC iterates satisfy $\|\mathbf{z}^{k,n} - \mathbf{z}_*^k\|^2 = \mathcal{O}(\frac{1}{\alpha^k n} + \alpha^k)$ for any stepsize $\alpha^k > 0$; hence, to compute an ϵ^k-optimal solution, DTAA-RFC requires $\alpha^k = \Theta(\epsilon^k)$ which implies $n = \mathcal{O}(1/(\epsilon^k)^2)$ inner iterations in contrast to $\mathcal{O}(1/\epsilon^k)$ needed for DTAA-RFA to compute an ϵ^k-optimal solution.

8.2.2 DTAA-RFA versus DTAA-BFC

In each inner iteration, DTAA-RFA requires a real-valued feedback from all links to the set of sources using them, whereas DTAA-BFC only requires congested links to send binary feedback to their sources. Thus, the overhead generated by DTAA-BFC per inner iteration is significantly less than that produced by DTAA-RFA. On the other hand, as discussed in Appendix E, available in the online supplemental material, for each k, θ^k should be sufficiently large for DTAA-BFC to converge—though there is no specific guidance on how to choose it. Over estimation of θ^k leads to the degradation of the convergence speed of DTAA-BFC; hence, the number of inner iterations required by DTAA-BFC can be larger than that needed by DTAA-RFA.

8.2.3 DTAA-RFC versus DTAA-BFC

Both DTAA-RFC and DTAA-BFC require feedback from congested links to the sources using them in every inner iteration. However, such feedback is binary for DTAA-BFC as opposed to real-valued for DTAA-RFC. Thus, the overhead generated by DTAA-BFC per inner iteration is significantly less than that produced by DTAA-RFC.

Acknowledgments

This work was partially supported by NSF grants CNS-1329422, XPS-1629625, SHF-1704504, CMMI-1400217, CMMI-1635106, ARO grant W911NF-17-1-0298, and The Chancellor Fund of Xiamen University under Grant No. 20720180090. The third and fourth authors contributed equally.

References

[1] M. Ashour, J. Wang, C. Lagoa, N. S. Aybat, and C. Hao, "Non-concave network utility maximization: A distributed optimization approach," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1908–1916.

[2] Cisco VNI Forecast and Methodology, 2016-2021, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[3] M. Allman, V. Paxson, and E. Blanton, "TCP congestion control," RFC 5681, Sep. 2009, doi: 10.17487/RFC5681.

[4] C. Hopps, "Analysis of an equal-cost multi-path algorithm," RFC 2992, Nov. 2000, doi: 10.17487/RFC2992.

[5] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 401–418, Jan.-Mar. 2016.

[6] M. Fazel and M. Chiang, "Network utility maximization with nonconcave utilities using sum-of-squares method, in *Proc. IEEE Conf. Decis. Control Eur. Control Conf.* 2005, pp. 1867–1874.

[7] L. Ye, Z. Wang, H. Che, and C. Lagoa, "TERSE: A unified end-to-end traffic control mechanism to enable elastic, delay adaptive, and rate adaptive services," *IEEE J. Select. Areas Commun.*, vol. 29, no. 5, pp. 938–950, May 2011.

[8] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat, "WCMP: Weighted cost multipathing for improved fairness in data centers," in *Proc. 9th Eur. Conf. Comput. Syst.*, 2014, Art. no. 5.

[9] S. Golestani and S. Bhattacharyya, "A class of end-to-end congestion control algorithms for the Internet," in *Proc. 6th Int. Conf. Netw. Protocols*, 1998, pp. 137–150.

[10] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS adaptive traffic engineering," in *Proc. 20th Annu. Joint Conf. IEEE Comput. Commun. Societies*, 2001, pp. 1300–1309.

[11] F. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Mar. 1, 1998.

[12] F. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 2, pp. 5–12, 2015.

[13] H. Han, S. Shakkottai, C. Hollot, R. Srikant, and D. Towsley, "Overlay TCP for multi-path routing and congestion control," *IMA Workshop Meas. Model. Internet*. Jan. 2004. [Online]. Available: https://www.researchgate.net/profile/CV_Hollot/publication/2871439_Overlay_TCP_for_Multi-Path_Routing_and_Congestion_Control/links/53ece79b0cf23733e804dbbe/Overlay-TCP-for-Multi-Path-Routing-and-Congestion-Control.pdf
[14] W. Wang, M. Palaniswami, and S. Low, "Optimal flow control and routing in multi-path networks," *Perform. Eval.*, vol. 52, no. 2, pp. 119–132, 2003.
[15] C. Lagoa and H. Che, "Decentralized optimal traffic engineering in the Internet," *ACM SIGCOMM Comput. Commun. Rev*, vol. 30, no. 5, pp. 39–47, 2000.
[16] C. Lagoa, H. Che, and B. Movsichoff, "Adaptive control algorithms for decentralized optimal traffic engineering in the internet," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, pp. 415–428, Jun. 2004.
[17] W. Su, C. Liu, C. Lagoa, H. Che, K. Xu, and Y. Cui, "Integrated, distributed traffic control in multidomain networks," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 4, pp. 1373–1386, Jul. 2015.
[18] B. Movsichoff, C. Lagoa, and H. Che, "Decentralized optimal traffic engineering in connectionless networks," *IEEE J. Select. Areas Commun.*, vol. 23, no. 2, pp. 293–303, Feb. 2005.
[19] B. Movsichoff, C. Lagoa, and H. Che, "End-to-end optimal algorithms for integrated QoS, traffic engineering, and failure recovery," *IEEE/ACM Trans. Netw.*, vol. 15, no. 4, pp. 813–823, Aug. 2007.
[20] M. Fazel and M. Chiang, "Network utility maximization with nonconcave utilities using sum-of-squares method," in *Proc. 44th IEEE Conf. Decis. Control.*, pp. 1867–1874, 2005.
[21] J. W. Lee, R. R. Mazumdar, and N. B. Shroff, "Non-convex optimization and rate control for multi-class services in the Internet," *IEEE/ACM Trans. Netw.*, vol. 13, no. 4, pp. 827–840, Aug. 2005.
[22] P. Hande, S. Zhang, and M. Chiang, "Distributed rate allocation for inelastic flows," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1240–1253, Dec. 2007.
[23] J. Wang, M. Ashour, C. Lagoa, N. Aybat, H. Che, and Z. Duan, "Non-concave network utility maximization in connectionless networks: A fully distributed traffic allocation algorithm," in *Proc. Amer. Control Conf.*, 3980–3985, 2017.
[24] J. B. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM J. Optimization*, vol. 11, no. 3, pp. 796–817, 2001.
[25] J. B. Lasserre, *Moments, Positive Polynomials and Their Applications*. Singapore: World Scientific, 2009.
[26] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–22, Jan. 1, 2011.
[27] J. Eckstein and D. P. Bertsekas, "On the DouglasRachford splitting method and the proximal point algorithm for maximal monotone operators," *Math. Program.*, vol. 55, no. 1, pp. 293–318, Apr. 1, 1992.
[28] A. Nedic and A. Ozdaglar, "Subgradient methods for saddle-point problems," *J. Optimization Theory Appl.*, vol. 142, no. 1, pp. 205–228, 2009.
[29] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 325–338, 2015.
[30] N. S. Aybat, Z. Wang, T. Lin, and S. Ma, "Distributed linearized alternating direction method of multipliers for composite convex consensus optimization," *IEEE Trans. Automat. Control*, vol. 63, no. 1, pp. 5–20, Jan. 2018.
[31] J. Sun, H. Li, and Z. Xu, "Deep ADMM-Net for compressive sensing MRI," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 10–18, 2016.
[32] J. Eckstein and W. Yao, "Approximate versions of the alternating direction method of multipliers," Tech. Rep. 2016-01-5276, Optimization Online, 2016. [Online] Available: http://www.optimization-online.org/DB_FILE/2016/01/5276.pdf
[33] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA, USA: Athena Scientific, Sep. 1, 1999.
[34] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Rev.*, vol. 38, no. 1, pp. 49–95, Mar. 1996.
[35] Y. Nesterov, Introductory lectures on convex programming volume i: Basic course. Lecture notes. Jul. 2, 1998, https://pdfs.semanticscholar.org/0479/353510fc33adcc7acf69211a455b518de477.pdf
[36] A. M. Jasour, N. S. Aybat, and C. M. Lagoa, "Semidefinite programming for chance constrained optimization over semialgebraic sets," *SIAM J. Optimization*, vol. 25, no. 3, pp. 1411–1440, Jul. 2015.

Mahmoud Ashour received the BSc and MSc degrees in electrical engineering from Cairo University and Nile University, Egypt, in 2010 and 2013, respectively. He was a research assistant with the Computer Science and Engineering Department, Qatar University, Qatar, for one year in 2014. He is currently working toward the PhD degree in the Electrical Engineering and Computer Science Department, Penn State University, University Park, Pennsylvania. His research interests lie in the broad area of communication networks with an emphasis on distributed optimization algorithms.

Jingyao Wang received the BS degree from Inner Mongolia University, Hohhot, China, in 2010, and the PhD degree in engineering from Peking University, Beijing, in 2017. Since July 2017, she has been with the Department of Automation, Xiamen University at Xiamen. Her current research interests include coordination control of multi-agent systems and network resource management.

Necdet Serhat Aybat received the BS and MS degrees in industrial engineering from Bogazici University, Istanbul, Turkey, in 2003 and 2005, respectively, and the PhD degree in operations research from Columbia University, New York, in 2011. He joined the Department of Industrial Engineering, Penn State University, State College, Pennsylvania, in August 2011. His current research interests include developing first-order algorithms for large-scale convex optimization problems from diverse application areas, such as compressed sensing, matrix completion, convex regression, and distributed optimization. He is a runner-up of the INFORMS Computing Society Student Paper Award in 2010, and received the SIAM Student Paper Prize in 2011.

Constantino Lagoa received the BS and MS degrees from the Instituto Superior Tecnico, Technical University of Lisbon, Portugal, in 1991 and 1994, respectively, and the PhD degree from the University of Wisconsin at Madison, in 1998. He joined the Electrical Engineering Department of Penn State University, University Park, Pennsylvania, in August 1998, where he currently holds the position of professor. He has a wide range of research interests including robust optimization and control, chance constrained optimization, controller design under risk specifications, system identification, control of computer networks and discrete event dynamical systems. He is currently an associate editor of the *IEEE Trans. Automat. Control Automatica*.

Hao Che received the BS degree from Nanjing University, Nanjing, China, in 1984, the MS degree in physics from the University of Texas at Arlington, Texas, in 1994, and the PhD degree in electrical engineering from the University of Texas at Austin, Texas, in 1998. He was an assistant professor of electrical engineering with the Pennsylvania State University, University Park, Pennsylvania, from 1998 to 2000, and a system architect with Santera Systems, Inc., Plano, Texas, from 2000 to 2002. Since September 2002, he has been with the Department of Computer Science and Engineering, University of Texas at Arlington, Texas. His current research interests include network architecture and network resource management, performance analysis of large-scale distributed computing systems, including many-core processors, warehouse-scale computing, and cloud computing.