SIAM JOURNAL ON OPTIMIZATION, VOL. 27, NO. 2, PP. 759–783, 2017

A SIMPLE PARALLEL ALGORITHM WITH AN $O(1/T)$ CONVERGENCE RATE FOR GENERAL CONVEX PROGRAMS*

HAO YU† AND MICHAEL J. NEELY†

Abstract. This paper considers convex programs with a general (possibly non-differentiable) convex objective function and Lipschitz continuous convex inequality constraint functions. A simple algorithm is developed and achieves an $O(1/t)$ convergence rate. Similar to the classical dual subgradient algorithm and the ADMM algorithm, the new algorithm has a parallel implementation when the objective and constraint functions are separable. However, the new algorithm has a faster $O(1/t)$ convergence rate compared with the best known $O(1/\sqrt{t})$ convergence rate for the dual subgradient algorithm with primal averaging. Further, it can solve convex programs with nonlinear constraints, which cannot be handled by the ADMM algorithm. The new algorithm is applied to a multipath network utility maximization problem and yields a decentralized flow control algorithm with the fast $O(1/t)$ convergence rate.

Key words. convex programs, parallel algorithms, convergence rates

AMS subject classifications. 90C25, 90C30

1. Introduction.

Fix positive integers n and m. Consider the general convex program:

$$\text{minimize} \quad f(\mathbf{x}) \tag{1}$$

$$\text{subject to} \quad g_k(\mathbf{x}) \leq 0, \forall k \in \{1, 2, \ldots, m\}, \tag{2}$$

$$\mathbf{x} \in \mathcal{X}, \tag{3}$$

where set $\mathcal{X} \subseteq \mathbb{R}^n$ is a closed convex set; function $f(\mathbf{x})$ is continuous and convex on $\mathcal{X}$; and functions $g_k(\mathbf{x}), \forall k \in \{1, 2, \ldots, m\}$ are convex and Lipschitz continuous on $\mathcal{X}$. Note that the functions $f(\mathbf{x}), g_1(\mathbf{x}), \ldots, g_m(\mathbf{x})$ are not necessarily differentiable. Denote the stacked vector of multiple functions $g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_m(\mathbf{x})$ as $\mathbf{g}(\mathbf{x}) = \big[g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_m(\mathbf{x})\big]^T$. The Lipschitz continuity of each $g_k(\mathbf{x})$ implies that $\mathbf{g}(\mathbf{x})$ is Lipschitz continuous on $\mathcal{X}$. Throughout this paper, we use $\|\cdot\|$ to denote the vector Euclidean norm and Lipschitz continuity is defined with respect to the Euclidean norm. The following assumptions are imposed on the convex program (1)-(3):

ASSUMPTION 1 (Basic Assumptions).

- *There exists a (possibly non-unique) optimal solution $\mathbf{x}^* \in \mathcal{X}$ that solves the convex program (1)-(3).*
- *There exists a constant β such that $\|\mathbf{g}(\mathbf{x}_1) - \mathbf{g}(\mathbf{x}_2)\| \leq \beta\|\mathbf{x}_1 - \mathbf{x}_2\|$ for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$, i.e., the Lipschitz continuous function $\mathbf{g}(\mathbf{x})$ has modulus β.*

ASSUMPTION 2 (Existence of Lagrange Multipliers). *The convex program (1)-(3) has Lagrange multipliers attaining the strong duality. That is, there exists a Lagrange multiplier vector $\boldsymbol{\lambda}^* = [\lambda_1^*, \lambda_2^*, \ldots, \lambda_m^*]^T \geq \mathbf{0}$ such that*

$$q(\boldsymbol{\lambda}^*) = \min_{\mathbf{x}\in\mathcal{X}} \Big\{ f(\mathbf{x}) : g_k(\mathbf{x}) \leq 0, \forall k \in \{1, 2, \ldots, m\} \Big\},$$

where $q(\boldsymbol{\lambda}) = \min_{\mathbf{x}\in\mathcal{X}}\{f(\mathbf{x}) + \sum_{k=1}^m \lambda_k g_k(\mathbf{x})\}$ is the Lagrangian dual function of problem (1)-(3).

*Using the methodology initiated in the current paper, we further developed a different primal-dual type algorithm with the same $O(1/t)$ convergence rate in the extended work [24].

†Department of Electrical Engineering, University of Southern California, Los Angeles, CA (yuhao@usc.edu, mjneely@usc.edu).

Assumption 2 is a mild assumption. For convex programs, Assumption 2 is implied by the existence of a vector $\mathbf{s} \in \mathcal{X}$ such that $g_k(\mathbf{s}) < 0$ for all $k \in \{1, \ldots, m\}$, called the *Slater condition* [1, 5]. However, there are convex programs where Assumption 2 holds but the Slater condition does not hold.

1.1. New Algorithm. Consider the following algorithm described in Algorithm 1. The algorithm computes vectors $\mathbf{x}(t) \in \mathcal{X}$ for iterations $t \in \{0, 1, 2, \ldots\}$. Define the average over the first $t > 0$ iterations as $\overline{\mathbf{x}}(t) = \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbf{x}(\tau)$. The algorithm uses an initial guess vector that is represented as $\mathbf{x}(-1)$ and that is chosen as any vector in $\mathcal{X}$. The algorithm also uses vector variables $\mathbf{Q}(t) = \big[Q_1(t), \ldots, Q_m(t)\big]^T$ in the computations. The main result of this paper is that, whenever the parameter α is chosen to satisfy $\alpha \geq \beta^2/2$, the vector $\overline{\mathbf{x}}(t)$ closely approximates a solution to the convex program and has an approximation error that decays like $O(1/t)$.

Algorithm 1

Let $\alpha > 0$ be a constant parameter. Choose any $\mathbf{x}(-1) \in \mathcal{X}$. Initialize $Q_k(0) = \max\{0, -g_k(\mathbf{x}(-1))\}, \forall k \in \{1, 2, \ldots, m\}$. At each iteration $t \in \{0, 1, 2, \ldots\}$, observe $\mathbf{x}(t-1)$ and $\mathbf{Q}(t)$ and do the following:

- Choose $\mathbf{x}(t)$ as

$$\mathbf{x}(t) = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{X}} \Big\{ f(\mathbf{x}) + \big[\mathbf{Q}(t) + \mathbf{g}(\mathbf{x}(t-1))\big]^T \mathbf{g}(\mathbf{x}) + \alpha\|\mathbf{x} - \mathbf{x}(t-1)\|^2 \Big\},$$

- Update virtual queues via

$$Q_k(t+1) = \max\{-g_k(\mathbf{x}(t)), Q_k(t) + g_k(\mathbf{x}(t))\}, \forall k \in \{1, 2, \ldots, m\}.$$

- Update the averages $\overline{\mathbf{x}}(t)$ via

$$\overline{\mathbf{x}}(t+1) = \frac{1}{t+1}\sum_{\tau=0}^{t} \mathbf{x}(\tau) = \overline{\mathbf{x}}(t)\frac{t}{t+1} + \mathbf{x}(t)\frac{1}{t+1}.$$

The variables $\mathbf{x}(t)$ are called *primal variables*. The variables $\mathbf{Q}(t)$ can be viewed as *dual variables* because they have a close connection to Lagrange multipliers. The variables $\mathbf{Q}(t)$ shall be called *virtual queues* because their update rule resembles a queueing equation. The virtual queue update used by Algorithm 1 is related to traditional virtual queue and dual variable update rules. However, there is an important difference. A traditional update rule is $Q_k(t+1) = \max\{Q_k(t) + g_k(\mathbf{x}(t)), 0\}$ [3, 16]. In contrast, the new algorithm takes a max with $-g_k(\mathbf{x}(t))$, rather than a max with 0. The primal update rule for $\mathbf{x}(t)$ is also new and is discussed in more detail in the next subsection.

Algorithm 1 has the following desirable property: If the functions $f(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ are separable with respect to components or blocks of $\mathbf{x}$, then the primal updates for $\mathbf{x}(t)$ can be decomposed into several smaller independent subproblems, each of which only involves a component or block of $\mathbf{x}(t)$.

1.2. The Dual Subgradient Algorithm and the Drift-Plus-Penalty Algorithm. The dual subgradient algorithm is a well known iterative technique that approaches optimality for certain strictly convex problems [3]. A modification of the dual subgradient algorithm that averages the resulting sequence of primal estimates

is known to solve general convex programs (without requiring strict convexity) and provides an $O(1/\sqrt{t})$ convergence rate[1] [15, 13, 17], where t is the number of iterations. Work [12] improves the convergence rate of the dual subgradient algorithm to $O(1/t)$ in the special case when the objective function $f(\mathbf{x})$ is strongly convex and second order differentiable and constraint functions $g_k(\mathbf{x})$ are second order differentiable and have bounded Jacobians. A recent work in [23] shows that the convergence rate of the dual subgradient algorithm with primal averaging is $O(1/t)$ without requiring the differentiability of $f(\mathbf{x})$ and $g_k(\mathbf{x})$ but still requiring the strong convexity of $f(\mathbf{x})$. (Further improvements are also possible under more restrictive assumptions, see [23].) The dual subgradient algorithm with primal averaging is also called the Drift-Plus-Penalty (DPP) algorithm. This is because it is a special case of a stochastic optimization procedure that minimizes a drift expression for a quadratic Lyapunov function [16]. One advantage of these dual subgradient and drift approaches is that the computations required at every iteration are simple and can yield parallel algorithms when $f(\mathbf{x})$ and $g_k(\mathbf{x})$ are separable.

Algorithm 1 developed in the current paper maintains this simplicity on every iteration, but provides fast $O(1/t)$ convergence for general convex programs, without requiring strict convexity or strong convexity. For example, the algorithm has the $O(1/t)$ convergence rate in the special case of linear $f(\mathbf{x})$. Algorithm 1 is similar to a DPP algorithm, or equivalently, a classic dual subgradient algorithm with primal averaging, with the following distinctions:

1. The Lagrange multiplier ("virtual queue") update equation for $Q_k(t)$ is modified to take a max with $-g_k(\mathbf{x}(t))$, rather than simply project onto the nonnegative real numbers.
2. The minimization step augments the $Q_k(t)$ weights with $g_k(\mathbf{x}(t-1))$ values obtained on the previous step. These $g_k(\mathbf{x}(t-1))$ quantities, when multiplied by constraint functions $g_k(\mathbf{x})$, yield a cross-product term in the primal update. This cross term together with another newly introduced quadratic term in the primal update can cancel a quadratic term in an upper bound of the Lyapunov drift such that a finer analysis of the drift-plus-penalty leads to the fast $O(1/t)$ convergence rate.
3. A quadratic term, which is similar to a term used in proximal algorithms [19], is introduced. This provides a strong convexity "pushback". The pushback is not sufficient to alone cancel the main drift components, but it cancels residual components introduced by the new $g_k(\mathbf{x}(t-1))$ weight.

1.3. The ADMM Algorithm. The Alternating Direction Method of Multipliers (ADMM) is an algorithm used to solve linear equality constrained convex programs in the following form:

$$\text{minimize} \quad f_1(\mathbf{x}) + f_2(\mathbf{y}) \tag{4}$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{c}, \tag{5}$$

$$\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}. \tag{6}$$

[1] In [15, 13, 17], the dual subgradient algorithm with primal averaging is proven to achieve an ϵ-approximate solution with $O(1/\epsilon^2)$ iterations by using an $O(\epsilon)$ step size. We say that the dual subgradient algorithm with primal averaging has an $O(1/\sqrt{t})$ convergence rate because an algorithm with $O(1/\sqrt{t})$ convergence requires the same $O(1/\epsilon^2)$ iterations to yield an ϵ-approximate solution. However, the dual subgradient algorithm in [15, 13, 17] does not have vanishing errors as does an algorithm with $O(1/\sqrt{t})$ convergence.

Define the augmented Lagrangian as $L_\rho(\mathbf{x},\mathbf{y},\boldsymbol{\lambda}) = f_1(\mathbf{x}) + f_2(\mathbf{y}) + \boldsymbol{\lambda}^T\big(\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}-\mathbf{c}\big) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}-\mathbf{c}\|^2$. At each iteration t, the ADMM algorithm consists of the following steps:

- Update $\mathbf{x}(t) = \text{argmin}_{\mathbf{x}\in\mathcal{X}}\, L_\rho(\mathbf{x},\mathbf{y}(t-1),\boldsymbol{\lambda}(t-1))$.
- Update $\mathbf{y}(t) = \text{argmin}_{\mathbf{y}\in\mathcal{Y}}\, L_\rho(\mathbf{x}(t),\mathbf{y},\boldsymbol{\lambda}(t-1))$.
- Update $\boldsymbol{\lambda}(t) = \boldsymbol{\lambda}(t-1) + \rho\big(\mathbf{A}\mathbf{x}(t)+\mathbf{B}\mathbf{y}(t)-\mathbf{c}\big)$.

Thus, the ADMM algorithm yields a distributed algorithm where the updates of $\mathbf{x}$ and $\mathbf{y}$ only involve local sub-problems and is suitable to solve large scale convex programs in machine learning, network scheduling, computational biology and finance [4].

The best known convergence rate of ADMM algorithm for convex program with general convex $f_1(\cdot)$ and $f_2(\cdot)$ is recently shown to be $O(1/t)$ [6, 9]. An asynchronous ADMM algorithm with the same $O(1/t)$ convergence rate is studied in [21]. Note that we can apply Algorithm 1 to solve the problem (4)-(6) after replacing the equality constraint $\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y} = \mathbf{c}$ by two linear inequality constraints $\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y} \le \mathbf{c}$ and $\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y} \ge \mathbf{c}$.[2] It can be observed that the algorithm yielded by Algorithm 1 is also separable for $\mathbf{x}$ and $\mathbf{y}$. In fact, the updates of $\mathbf{x}$ and $\mathbf{y}$ in Algorithm 1 are fully parallel while the ADMM algorithm updates $\mathbf{x}$ and $\mathbf{y}$ sequentially. The remaining part of this paper shows that the convergence rate of Algorithm 1 is also $O(1/t)$.

However, a significant limitation of the ADMM algorithm is that it can only solve problems with linear constraints. In contrast, Algorithm 1 proposed in this paper can solve general convex programs with non-linear constraints.

1.4. Decentralized Multipath Network Flow Control Problems. Section 4 presents an example application to multipath network flow control problems. The algorithm has a queue-based interpretation that is natural for networks. Prior work on distributed optimization for networks is in [11, 15, 2, 23, 22]. It is known that the DPP algorithm achieves $O(1/\sqrt{t})$ convergence for general networks [15], and several algorithms show faster $O(1/t)$ convergence for special classes of strongly convex problems [2, 23]. However, multipath network flow problems fundamentally fail to satisfy the strong convexity property because they have routing variables that participate in the constraints but not in the objective function. The algorithm of the current paper does not require strong convexity. It easily solves this multi-path scenario with fast $O(1/t)$ convergence. The algorithm also has a simple distributed implementation and allows for general convex but nonlinear constraint functions.

2. Preliminaries and Basic Analysis. This section presents useful preliminaries in convex analysis and important facts of Algorithm 1.

2.1. Preliminaries.

DEFINITION 1 (Lipschitz Continuity). *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a convex set. Function $h : \mathcal{X} \to \mathbb{R}^m$ is said to be Lipschitz continuous on $\mathcal{X}$ with modulus L if there exists $L > 0$ such that $\|h(\mathbf{y}) - h(\mathbf{x})\| \le L\|\mathbf{y}-\mathbf{x}\|$ for all $\mathbf{x},\mathbf{y} \in \mathcal{X}$.*

DEFINITION 2 (Strongly Convex Functions). *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a convex set. Function h is said to be strongly convex on $\mathcal{X}$ with modulus α if there exists a constant $\alpha > 0$ such that $h(\mathbf{x}) - \frac{1}{2}\alpha\|\mathbf{x}\|^2$ is convex on $\mathcal{X}$.*

[2] In fact, we do not need to replace the linear equality constraint with two inequality constraints. We can apply Algorithm 1 to problem (4)-(6) directly by modifying the virtual queue update equations as $Q_k(t+1) = Q_k(t) + g_k(\mathbf{x}(t)), \forall k \in \{1,2,\ldots,m\}$. In this case, a simple adaption of the convergence rate analysis in this paper can establish the same $O(1/t)$ convergence rate. However, to simplify the presentation, this paper just considers the general convex program in the form of (1)-(3) since any linear equality can be equivalently represented by two linear inequalities.

By the definition of strongly convex functions, it is easy to show that if $h(\mathbf{x})$ is convex and $\alpha > 0$, then $h(\mathbf{x}) + \alpha\|\mathbf{x} - \mathbf{x}_0\|^2$ is strongly convex with modulus 2α for any constant $\mathbf{x}_0$.

LEMMA 1 (Theorem 6.1.2 in [7]). *Let $h(\mathbf{x})$ be strongly convex on $\mathcal{X}$ with modulus α. Let $\partial h(\mathbf{x})$ be the set of all subgradients of h at point $\mathbf{x}$. Then $h(\mathbf{y}) \geq h(\mathbf{x}) + \mathbf{d}^T(\mathbf{y} - \mathbf{x}) + \frac{\alpha}{2}\|\mathbf{y} - \mathbf{x}\|^2$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and all $\mathbf{d} \in \partial h(\mathbf{x})$.*

LEMMA 2 (Proposition B.24 (f) in [3]). *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a convex set. Let function h be convex on $\mathcal{X}$ and $\mathbf{x}^{opt}$ be a global minimum of h on $\mathcal{X}$. Let $\partial h(\mathbf{x})$ be the set of all subgradients of h at point $\mathbf{x}$. Then, there exists $\mathbf{d} \in \partial h(\mathbf{x}^{opt})$ such that $\mathbf{d}^T(\mathbf{x} - \mathbf{x}^{opt}) \geq 0$ for all $\mathbf{x} \in \mathcal{X}$.*

COROLLARY 1. *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a convex set. Let function h be strongly convex on $\mathcal{X}$ with modulus α and $\mathbf{x}^{opt}$ be a global minimum of h on $\mathcal{X}$. Then, $h(\mathbf{x}^{opt}) \leq h(\mathbf{x}) - \frac{\alpha}{2}\|\mathbf{x}^{opt} - \mathbf{x}\|^2$ for all $\mathbf{x} \in \mathcal{X}$.*

Proof. A special case when h is differentiable and $\mathcal{X} = \mathbb{R}^n$ is Theorem 2.1.8 in [18]. The proof for general h and $\mathcal{X}$ is as follows: Fix $\mathbf{x} \in \mathcal{X}$. By Lemma 2, there exists $\mathbf{d} \in \partial h(\mathbf{x}^{opt})$ such that $\mathbf{d}^T(\mathbf{x} - \mathbf{x}^{opt}) \geq 0$. By Lemma 1, we also have

$$\begin{aligned} h(\mathbf{x}) &\geq h(\mathbf{x}^{opt}) + \mathbf{d}^T(\mathbf{x} - \mathbf{x}^{opt}) + \frac{\alpha}{2}\|\mathbf{x} - \mathbf{x}^{opt}\|^2 \\ &\overset{(a)}{\geq} h(\mathbf{x}^{opt}) + \frac{\alpha}{2}\|\mathbf{x} - \mathbf{x}^{opt}\|^2, \end{aligned}$$

where (a) follows from the fact that $\mathbf{d}^T(\mathbf{x} - \mathbf{x}^{opt}) \geq 0$. ∎

2.2. Properties of the Virtual Queues.

LEMMA 3. *In Algorithm 1, we have*

1. *At each iteration $t \in \{0, 1, 2, \ldots\}$, $Q_k(t) \geq 0$ for all $k \in \{1, 2, \ldots, m\}$.*
2. *At each iteration $t \in \{0, 1, 2, \ldots\}$, $Q_k(t) + g_k(\mathbf{x}(t-1)) \geq 0$ for all $k \in \{1, 2 \ldots, m\}$.*
3. *At iteration $t = 0$, $\|\mathbf{Q}(0)\|^2 \leq \|\mathbf{g}(\mathbf{x}(-1))\|^2$. At each iteration $t \in \{1, 2, \ldots\}$, $\|\mathbf{Q}(t)\|^2 \geq \|\mathbf{g}(\mathbf{x}(t-1))\|^2$.*

Proof.

1. Fix $k \in \{1, 2, \ldots, m\}$. Note that $Q_k(0) \geq 0$ by the initialization rule $Q_k(0) = \max\{0, -g_k(\mathbf{x}(-1))\}$. Assume $Q_k(t) \geq 0$ and consider time $t+1$. If $g_k(\mathbf{x}(t)) \geq 0$, then $Q_k(t+1) = \max\{-g_k(\mathbf{x}(t)), Q_k(t) + g_k(\mathbf{x}(t))\} \geq Q_k(t) + g_k(\mathbf{x}(t)) \geq 0$. If $g_k(\mathbf{x}(t)) < 0$, then $Q_k(t+1) = \max\{-g_k(\mathbf{x}(t)), Q_k(t) + g_k(\mathbf{x}(t))\} \geq -g_k(\mathbf{x}(t)) > 0$. Thus, $Q_k(t+1) \geq 0$. The result follows by induction.
2. Fix $k \in \{1, 2, \ldots, m\}$. Note that $Q_k(0) + g_k(\mathbf{x}(-1)) \geq 0$ by the initialization rule $Q_k(0) = \max\{0, -g_k(\mathbf{x}(-1))\} \geq -g_k(\mathbf{x}(-1))$. For $t \geq 1$, by the virtual queue update equation, we have

$$Q_k(t) = \max\{-g_k(\mathbf{x}(t-1)), Q_k(t-1) + g_k(\mathbf{x}(t-1))\} \geq -g_k(\mathbf{x}(t-1)),$$

   which implies that $Q_k(t) + g_k(\mathbf{x}(t-1)) \geq 0$.
3. - For $t = 0$. Fix $k \in \{1, 2, \ldots, m\}$. Consider the cases $g_k(\mathbf{x}(-1)) \geq 0$ and $g_k(\mathbf{x}(-1)) < 0$ separately. If $g_k(\mathbf{x}(-1)) \geq 0$, then $Q_k(0) = \max\{0, -g_k(\mathbf{x}(-1))\} = 0$ and so $|Q_k(0)| \leq |g_k(\mathbf{x}(-1))|$. If $g_k(\mathbf{x}(-1)) < 0$, then $Q_k(0) = \max\{0, -g_k(\mathbf{x}(-1))\} = -g_k(\mathbf{x}(-1))$. Thus, in both cases, we have $|Q_k(0)| \leq |g_k(\mathbf{x}(-1))|$. Squaring both sides and summing over $k \in \{1, 2, \ldots, m\}$ yields $\|\mathbf{Q}(0)\|^2 \leq \|\mathbf{g}(\mathbf{x}(-1))\|^2$.

- For $t \geq 1$. Fix $k \in \{1, 2, \ldots, m\}$. Consider the cases $g_k(\mathbf{x}(t-1)) \geq 0$ and $g_k(\mathbf{x}(t-1)) < 0$ separately. If $g_k(\mathbf{x}(t-1)) \geq 0$, then

$$\begin{aligned} Q_k(t) &= \max\{-g_k(\mathbf{x}(t-1)), Q_k(t-1) + g_k(\mathbf{x}(t-1))\} \\ &\geq Q_k(t-1) + g_k(\mathbf{x}(t-1)) \\ &\overset{(a)}{\geq} g_k(\mathbf{x}(t-1)) \\ &= |g_k(\mathbf{x}(t-1))|, \end{aligned}$$

where (a) follows from part 1. If $g_k(\mathbf{x}(t-1)) < 0$, then

$$\begin{aligned} Q_k(t) &= \max\{-g_k(\mathbf{x}(t-1)), Q_k(t-1) + g_k(\mathbf{x}(t-1))\} \\ &\geq -g_k(\mathbf{x}(t-1)) \\ &= |g_k(\mathbf{x}(t-1))|. \end{aligned}$$

Thus, in both cases, we have $|Q_k(t)| \geq |g_k(\mathbf{x}(t-1))|$. Squaring both sides and summing over $k \in \{1, 2, \ldots, m\}$ yields $\|\mathbf{Q}(t)\|^2 \geq \|\mathbf{g}(\mathbf{x}(t-1))\|^2$. ∎

2.3. Properties of the Drift. Recall that $\mathbf{Q}(t) = \big[Q_1(t), \ldots, Q_m(t)\big]^T$ is the vector of virtual queue backlogs. Define $L(t) = \frac{1}{2}\|\mathbf{Q}(t)\|^2$. The function $L(t)$ shall be called a *Lyapunov function*. Define the Lyapunov drift as

$$\Delta(t) = L(t+1) - L(t) = \frac{1}{2}\big[\|\mathbf{Q}(t+1)\|^2 - \|\mathbf{Q}(t)\|^2\big]. \tag{7}$$

LEMMA 4. *At each iteration $t \in \{0, 1, 2, \ldots\}$ in Algorithm 1, an upper bound of the Lyapunov drift is given by*

$$\Delta(t) \leq \mathbf{Q}^T(t)\mathbf{g}(\mathbf{x}(t)) + \|\mathbf{g}(\mathbf{x}(t))\|^2. \tag{8}$$

Proof. The virtual queue update equations $Q_k(t+1) = \max\{-g_k(\mathbf{x}(t)), Q_k(t) + g_k(\mathbf{x}(t))\}, \forall k \in \{1, 2, \ldots, m\}$ can be rewritten as

$$Q_k(t+1) = Q_k(t) + \tilde{g}_k(\mathbf{x}(t)), \forall k \in \{1, 2, \ldots, m\}, \tag{9}$$

where

$$\tilde{g}_k(\mathbf{x}(t)) = \begin{cases} g_k(\mathbf{x}(t)), & \text{if } Q_k(t) + g_k(\mathbf{x}(t)) \geq -g_k(\mathbf{x}(t)) \\ -Q_k(t) - g_k(\mathbf{x}(t)), & \text{else} \end{cases} \quad \forall k.$$

Fix $k \in \{1, 2, \ldots, m\}$. Squaring both sides of (9) and dividing by 2 yields:

$$\begin{aligned} &\frac{1}{2}(Q_k(t+1))^2 \\ =&\frac{1}{2}(Q_k(t))^2 + \frac{1}{2}\big(\tilde{g}_k(\mathbf{x}(t))\big)^2 + Q_k(t)\tilde{g}_k(\mathbf{x}(t)) \\ =&\frac{1}{2}(Q_k(t))^2 + \frac{1}{2}\big(\tilde{g}_k(\mathbf{x}(t))\big)^2 + Q_k(t)g_k(\mathbf{x}(t)) + Q_k(t)\big(\tilde{g}_k(\mathbf{x}(t)) - g_k(\mathbf{x}(t))\big) \\ \overset{(a)}{=}&\frac{1}{2}(Q_k(t))^2 + \frac{1}{2}\big(\tilde{g}_k(\mathbf{x}(t))\big)^2 + Q_k(t)g_k(\mathbf{x}(t)) \\ &- \big(\tilde{g}_k(\mathbf{x}(t)) + g_k(\mathbf{x}(t))\big)\big(\tilde{g}_k(\mathbf{x}(t)) - g_k(\mathbf{x}(t))\big) \\ =&\frac{1}{2}(Q_k(t))^2 - \frac{1}{2}\big(\tilde{g}_k(\mathbf{x}(t))\big)^2 + Q_k(t)g_k(\mathbf{x}(t)) + \big(g_k(\mathbf{x}(t))\big)^2 \\ \leq&\frac{1}{2}(Q_k(t))^2 + Q_k(t)g_k(\mathbf{x}(t)) + \big(g_k(\mathbf{x}(t))\big)^2, \end{aligned}$$

where (a) follows from the fact that $Q_k(t)\big(\tilde{g}_k(\mathbf{x}(t)) - g_k(\mathbf{x}(t))\big) = -\big(\tilde{g}_k(\mathbf{x}(t)) + g_k(\mathbf{x}(t))\big)\big(\tilde{g}_k(\mathbf{x}(t)) - g_k(\mathbf{x}(t))\big)$, which can be shown by considering $\tilde{g}_k(\mathbf{x}(t)) = g_k(\mathbf{x}(t))$ and $\tilde{g}_k(\mathbf{x}(t)) \neq g_k(\mathbf{x}(t))$. Summing over $k \in \{1, 2, \ldots, m\}$ yields

$$\frac{1}{2}\|\mathbf{Q}(t+1)\|^2 \leq \frac{1}{2}\|\mathbf{Q}(t)\|^2 + \mathbf{Q}^T(t)\mathbf{g}(\mathbf{x}(t)) + \|\mathbf{g}(\mathbf{x}(t))\|^2.$$

Rearranging the terms yields the desired result. ∎

3. Convergence Rate Analysis of Algorithm 1 . This section analyzes the convergence rate of Algorithm 1 for the problem (1)-(3).

3.1. An Upper Bound of the Drift-Plus-Penalty Expression.

LEMMA 5. *Let $\mathbf{x}^*$ be an optimal solution of the problem (1)-(3). If $\alpha \geq \frac{1}{2}\beta^2$ in Algorithm 1, then for all $t \geq 0$, we have*

$$\begin{aligned}&\Delta(t) + f(\mathbf{x}(t))\\ \leq& f(\mathbf{x}^*) + \alpha\big[\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 - \|\mathbf{x}^* - \mathbf{x}(t)\|^2\big] + \frac{1}{2}\big[\|\mathbf{g}(\mathbf{x}(t))\|^2 - \|\mathbf{g}(\mathbf{x}(t-1))\|^2\big],\end{aligned}$$

where β is defined in Assumption 1.

Proof. Fix $t \geq 0$. Note that Lemma 3 implies that $\mathbf{Q}(t)+\mathbf{g}(\mathbf{x}(t-1))$ is componentwise nonnegative. Hence, the function $f(\mathbf{x}) + \big[\mathbf{Q}(t) + \mathbf{g}(\mathbf{x}(t-1))\big]^T\mathbf{g}(\mathbf{x})$ is convex with respect to $\mathbf{x}$ on $\mathcal{X}$. Since $\alpha\|\mathbf{x} - \mathbf{x}(t-1)\|^2$ is strongly convex with respect to $\mathbf{x}$ with modulus 2α, it follows that

$$f(\mathbf{x}) + \big[\mathbf{Q}(t) + \mathbf{g}(\mathbf{x}(t-1))\big]^T\mathbf{g}(\mathbf{x}) + \alpha\|\mathbf{x} - \mathbf{x}(t-1)\|^2$$

is strongly convex with respect to $\mathbf{x}$ with modulus 2α.

Since $\mathbf{x}(t)$ is chosen to minimize the above strongly convex function, by Corollary 1, we have

$$\begin{aligned}&f(\mathbf{x}(t)) + \big[\mathbf{Q}(t) + \mathbf{g}(\mathbf{x}(t-1))\big]^T\mathbf{g}(\mathbf{x}(t)) + \alpha\|\mathbf{x}(t) - \mathbf{x}(t-1)\|^2\\ \leq& f(\mathbf{x}^*) + \underbrace{\big[\mathbf{Q}(t) + \mathbf{g}(\mathbf{x}(t-1))\big]^T\mathbf{g}(\mathbf{x}^*)}_{\leq 0} + \alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 - \alpha\|\mathbf{x}^* - \mathbf{x}(t)\|^2\\ \overset{(a)}{\leq}& f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 - \alpha\|\mathbf{x}^* - \mathbf{x}(t)\|^2,\end{aligned} \tag{10}$$

where (a) follows by using the fact that $g_k(\mathbf{x}^*) \leq 0$ for all $k \in \{1, 2, \ldots, m\}$ and $Q_k(t) + g_k(\mathbf{x}(t-1)) \geq 0$ (i.e., part 2 in Lemma 3) to eliminate the term marked by an underbrace.

Note that $\mathbf{u}_1^T\mathbf{u}_2 = \frac{1}{2}\big[\|\mathbf{u}_1\|^2 + \|\mathbf{u}_2\|^2 - \|\mathbf{u}_1 - \mathbf{u}_2\|^2\big]$ for any $\mathbf{u}_1, \mathbf{u}_2 \in \mathbb{R}^m$. Thus, we have

$$\mathbf{g}(\mathbf{x}(t-1))^T\mathbf{g}(\mathbf{x}(t)) = \frac{1}{2}\big[\|\mathbf{g}(\mathbf{x}(t-1))\|^2 + \|\mathbf{g}(\mathbf{x}(t))\|^2 - \|\mathbf{g}(\mathbf{x}(t-1)) - \mathbf{g}(\mathbf{x}(t))\|^2\big]. \tag{11}$$

Substituting (11) into (10) and rearranging terms yields

$$\begin{aligned}
&f(\mathbf{x}(t)) + \mathbf{Q}^T(t)\mathbf{g}(\mathbf{x}(t)) \\
\le& f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 - \alpha\|\mathbf{x}^* - \mathbf{x}(t)\|^2 - \alpha\|\mathbf{x}(t) - \mathbf{x}(t-1)\|^2 \\
&+ \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t-1)) - \mathbf{g}(\mathbf{x}(t))\|^2 - \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t-1))\|^2 - \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t))\|^2 \\
\overset{(a)}{\le}& f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 - \alpha\|\mathbf{x}^* - \mathbf{x}(t)\|^2 + (\frac{1}{2}\beta^2 - \alpha)\|\mathbf{x}(t) - \mathbf{x}(t-1)\|^2 \\
&- \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t-1))\|^2 - \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t))\|^2 \\
\overset{(b)}{\le}& f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 - \alpha\|\mathbf{x}^* - \mathbf{x}(t)\|^2 - \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t-1))\|^2 - \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t))\|^2,
\end{aligned}$$

where (a) follows from the fact that $\|\mathbf{g}(\mathbf{x}(t-1)) - \mathbf{g}(\mathbf{x}(t))\| \le \beta\|\mathbf{x}(t) - \mathbf{x}(t-1)\|$, which further follows from the assumption that $\mathbf{g}(\mathbf{x})$ is Lipschitz continuous with modulus β; and (b) follows from the fact $\alpha \ge \frac{1}{2}\beta^2$.

Summing (8) with the above inequality yields

$$\begin{aligned}
&\Delta(t) + f(\mathbf{x}(t)) \\
\le& f(\mathbf{x}^*) + \alpha\big[\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 - \|\mathbf{x}^* - \mathbf{x}(t)\|^2\big] + \frac{1}{2}\big[\|\mathbf{g}(\mathbf{x}(t))\|^2 - \|\mathbf{g}(\mathbf{x}(t-1))\|^2\big]. \quad \square
\end{aligned}$$

3.2. Objective Value Violations.

LEMMA 6. *Let $\mathbf{x}^*$ be an optimal solution of the problem (1)-(3) and β be defined in Assumption 1.*

1. *If $\alpha \ge \frac{1}{2}\beta^2$ in Algorithm 1, then for all $t \ge 1$, we have $\sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) \le tf(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2$.*
2. *If $\alpha > \frac{1}{2}\beta^2$ in Algorithm 1, then for all $t \ge 1$, we have $\sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) \le tf(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 + \frac{\alpha}{2\alpha-\beta^2}\|\mathbf{g}(\mathbf{x}^*)\|^2 - \frac{1}{2}\|\mathbf{Q}(t)\|^2$.*

Proof. By Lemma 5, we have $\Delta(\tau) + f(\mathbf{x}(\tau)) \le f(\mathbf{x}^*) + \alpha[\|\mathbf{x}^* - \mathbf{x}(\tau-1)\|^2 - \|\mathbf{x}^* - \mathbf{x}(\tau)\|] + \frac{1}{2}[\|\mathbf{g}(\mathbf{x}(\tau))\|^2 - \|\mathbf{g}(\mathbf{x}(\tau-1))\|^2]$ for all $\tau \in \{0, 1, 2, \ldots\}$. Summing over $\tau \in \{0, 1, \ldots, t-1\}$ yields

$$\begin{aligned}
\sum_{\tau=0}^{t-1}\Delta(\tau) + \sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) \le tf(\mathbf{x}^*) + \alpha\sum_{\tau=0}^{t-1}[\|\mathbf{x}^* - \mathbf{x}(\tau-1)\|^2 - \|\mathbf{x}^* - \mathbf{x}(\tau)\|^2] \\
+ \frac{1}{2}\sum_{\tau=0}^{t-1}[\|\mathbf{g}(\mathbf{x}(\tau))\|^2 - \|\mathbf{g}(\mathbf{x}(\tau-1))\|^2].
\end{aligned}$$

Recalling that $\Delta(\tau) = L(\tau+1) - L(\tau)$ and simplifying summations yields

$$\begin{aligned}
&L(t) - L(0) + \sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) \\
\le& tf(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 - \alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 + \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t-1))\|^2 - \frac{1}{2}\|\mathbf{g}(\mathbf{x}(-1))\|^2.
\end{aligned}$$

Rearranging terms; and substituting $L(0) = \frac{1}{2}\|\mathbf{Q}(0)\|^2$ and $L(t) = \frac{1}{2}\|\mathbf{Q}(t)\|^2$ yields

$$\begin{aligned}
&\sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) \\
\leq& t f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 - \alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 + \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t-1))\|^2 \\
&- \frac{1}{2}\|\mathbf{g}(\mathbf{x}(-1))\|^2 + \frac{1}{2}\|\mathbf{Q}(0)\|^2 - \frac{1}{2}\|\mathbf{Q}(t)\|^2 \qquad (12)\\
\overset{(a)}{\leq}& t f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 - \alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 + \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t-1))\|^2 - \frac{1}{2}\|\mathbf{Q}(t)\|^2,
\end{aligned}$$

where (a) follows from the fact that $\|\mathbf{Q}(0)\| \leq \|\mathbf{g}(\mathbf{x}(-1))\|$, i.e., part 3 in Lemma 3.

Next, we present the proof of both parts:

1. This part follows from the observation that the equation (12) can be further simplified as

$$\begin{aligned}
\sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) &\overset{(a)}{\leq} t f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 + \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t-1))\|^2 - \frac{1}{2}\|\mathbf{Q}(t)\|^2 \\
&\overset{(b)}{\leq} t f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2,
\end{aligned}$$

   where (a) follows by ignoring the non-positive term $-\alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2$ on the right side and (b) follows from the fact that $\|\mathbf{Q}(t)\| \geq \|\mathbf{g}(\mathbf{x}(t-1))\|$, i.e., part 3 in Lemma 3.

2. This part follows by rewriting the equation (12) as

$$\begin{aligned}
&\sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) \\
\leq& t f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 - \alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 + \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t-1)) - \mathbf{g}(\mathbf{x}^*) + \mathbf{g}(\mathbf{x}^*)\|^2 \\
&- \frac{1}{2}\|\mathbf{Q}(t)\|^2 \\
=& t f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 - \alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 + \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t-1)) - \mathbf{g}(\mathbf{x}^*)\|^2 \\
&+ \mathbf{g}^T(\mathbf{x}^*)[\mathbf{g}(\mathbf{x}(t-1)) - \mathbf{g}(\mathbf{x}^*)] + \frac{1}{2}\|\mathbf{g}(\mathbf{x}^*)\|^2 - \frac{1}{2}\|\mathbf{Q}(t)\|^2 \\
\overset{(a)}{\leq}& t f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 - \alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 + \frac{1}{2}\|\mathbf{g}(\mathbf{x}(t-1)) - \mathbf{g}(\mathbf{x}^*)\|^2 \\
&+ \|\mathbf{g}(\mathbf{x}^*)\|\|\mathbf{g}(\mathbf{x}(t-1)) - \mathbf{g}(\mathbf{x}^*)\| + \frac{1}{2}\|\mathbf{g}(\mathbf{x}^*)\|^2 - \frac{1}{2}\|\mathbf{Q}(t)\|^2 \\
\overset{(b)}{\leq}& t f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 - \alpha\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 + \frac{1}{2}\beta^2\|\mathbf{x}^* - \mathbf{x}(t-1)\|^2 \\
&+ \beta\|\mathbf{g}(\mathbf{x}^*)\|\|\mathbf{x}^* - \mathbf{x}(t-1)\| + \frac{1}{2}\|\mathbf{g}(\mathbf{x}^*)\|^2 - \frac{1}{2}\|\mathbf{Q}(t)\|^2 \\
=& t f(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 - \big(\alpha - \frac{1}{2}\beta^2\big)\Big[\|\mathbf{x}^* - \mathbf{x}(t-1)\| - \frac{1}{2}\frac{\beta}{\alpha - \frac{1}{2}\beta^2}\|\mathbf{g}(\mathbf{x}^*)\|\Big]^2 \\
&+ \frac{\alpha}{2\alpha - \beta^2}\|\mathbf{g}(\mathbf{x}^*)\|^2 - \frac{1}{2}\|\mathbf{Q}(t)\|^2
\end{aligned}$$

$$\overset{(c)}{\leq} tf(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 + \frac{\alpha}{2\alpha - \beta^2}\|\mathbf{g}(\mathbf{x}^*)\|^2 - \frac{1}{2}\|\mathbf{Q}(t)\|^2,$$

where (a) follows from Cauchy-Schwarz inequality; (b) follows from the fact that $\|\mathbf{g}(\mathbf{x}(t-1)) - \mathbf{g}(\mathbf{x}^*)\| \leq \beta\|\mathbf{x}^* - \mathbf{x}(t-1)\|$, which further follows from the assumption that $\mathbf{g}(\mathbf{x})$ is Lipschitz continuous with modulus β; and (c) follows from the fact that $\alpha > \frac{1}{2}\beta^2$. ∎

THEOREM 1 (Objective Value Violations). *Let $\mathbf{x}^*$ be an optimal solution of the problem (1)-(3). If $\alpha \geq \frac{1}{2}\beta^2$ in Algorithm 1, for all $t \geq 1$, we have*

$$f(\overline{\mathbf{x}}(t)) \leq f(\mathbf{x}^*) + \frac{\alpha}{t}\|\mathbf{x}^* - \mathbf{x}(-1)\|^2,$$

where β is defined in Assumption 1.

Proof. Fix $t \geq 1$. By part 1 in Lemma 6, we have

$$\sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) \leq tf(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2$$
$$\Rightarrow \frac{1}{t}\sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) \leq f(\mathbf{x}^*) + \frac{\alpha}{t}\|\mathbf{x}^* - \mathbf{x}(-1)\|^2.$$

Since $\overline{\mathbf{x}}(t) = \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbf{x}(\tau)$ and $f(\mathbf{x})$ is convex, by Jensen's inequality it follows that

$$f(\overline{\mathbf{x}}(t)) \leq \frac{1}{t}\sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)).$$ ∎

The above theorem shows that the error gap between $f(\overline{\mathbf{x}}(t))$ and the optimal value $f(\mathbf{x}^*)$ is at most $O(1/t)$. This holds for any initial guess vector $\mathbf{x}(-1) \in \mathcal{X}$. Of course, choosing $\mathbf{x}(-1)$ close to $\mathbf{x}^*$ is desirable because it reduces the coefficient $\alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2$.

3.3. Constraint Violations.

LEMMA 7. *Let $\mathbf{Q}(t), t \in \{0, 1, \ldots\}$ be the sequence generated by Algorithm 1. For any $t \geq 1$,*

$$Q_k(t) \geq \sum_{\tau=0}^{t-1} g_k(\mathbf{x}(\tau)), \forall k \in \{1, 2, \ldots, m\}.$$

Proof. Fix $k \in \{1, 2, \ldots, m\}$ and $t \geq 1$. For any $\tau \in \{0, \ldots, t-1\}$ the update rule of Algorithm 1 gives:

$$\begin{aligned} Q_k(\tau + 1) &= \max\{-g_k(\mathbf{x}(\tau)), Q_k(\tau) + g_k(\mathbf{x}(\tau))\} \\ &\geq Q_k(\tau) + g_k(\mathbf{x}(\tau)). \end{aligned}$$

Hence, $Q_k(\tau+1) - Q_k(\tau) \geq g_k(\mathbf{x}(\tau))$. Summing over $\tau \in \{0, \ldots, t-1\}$ and using $Q_k(0) \geq 0$ gives the result. ∎

LEMMA 8. *Let $\mathbf{x}^*$ be an optimal solution of the problem (1)-(3) and $\boldsymbol{\lambda}^*$ be a Lagrange multiplier vector satisfying Assumption 2. Let $\mathbf{x}(t), \mathbf{Q}(t), t \in \{0, 1, \ldots\}$ be sequences generated by Algorithm 1. Then,*

$$\sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) \geq tf(\mathbf{x}^*) - \|\boldsymbol{\lambda}^*\|\|\mathbf{Q}(t)\|, \quad \forall t \geq 1.$$

Proof. The proof is similar to a related result in [23] for the DPP algorithm. Define Lagrangian dual function $q(\boldsymbol{\lambda}) = \min_{\mathbf{x}\in\mathcal{X}}\{f(\mathbf{x}) + \sum_{k=1}^{m}\lambda_k g_k(\mathbf{x})\}$. For all $\tau \in \{0, 1, \ldots\}$, by Assumption 2, we have

$$f(\mathbf{x}^*) = q(\boldsymbol{\lambda}^*) \overset{(a)}{\leq} f(\mathbf{x}(\tau)) + \sum_{k=1}^{m}\lambda_k^* g_k(\mathbf{x}(\tau)),$$

where (a) follows the definition of $q(\boldsymbol{\lambda}^*)$. Thus, we have

$$f(\mathbf{x}(\tau)) \geq f(\mathbf{x}^*) - \sum_{k=1}^{m}\lambda_k^* g_k(\mathbf{x}(\tau)), \forall \tau \in \{0, 1, \ldots\}.$$

Summing over $\tau \in \{0, 1, \ldots, t-1\}$ yields

$$\begin{aligned}\sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) \geq& tf(\mathbf{x}^*) - \sum_{\tau=0}^{t-1}\sum_{k=1}^{m}\lambda_k^* g_k(\mathbf{x}(\tau))\\ =& tf(\mathbf{x}^*) - \sum_{k=1}^{m}\lambda_k^*\Big[\sum_{\tau=0}^{t-1} g_k(\mathbf{x}(\tau))\Big]\\ \overset{(a)}{\geq}& tf(\mathbf{x}^*) - \sum_{k=1}^{m}\lambda_k^* Q_k(t)\\ \overset{(b)}{\geq}& tf(\mathbf{x}^*) - \|\boldsymbol{\lambda}^*\|\|\mathbf{Q}(t)\|,\end{aligned}$$

where (a) follows from Lemma 7 and the fact that $\lambda_k^* \geq 0, \forall k \in \{1, 2, \ldots, m\}$; and (b) follows from the Cauchy-Schwarz inequality. ∎

LEMMA 9. *Let $\mathbf{x}^*$ be an optimal solution of the problem (1)-(3) and $\boldsymbol{\lambda}^*$ be a Lagrange multiplier vector satisfying Assumption 2. If $\alpha > \frac{\beta^2}{2}$ in Algorithm 1, then for all $t \geq 1$, the virtual queue vector satisfies*

$$\|\mathbf{Q}(t)\| \leq 2\|\boldsymbol{\lambda}^*\| + \sqrt{2\alpha}\|\mathbf{x}^* - \mathbf{x}(-1)\| + \sqrt{\frac{\alpha}{\alpha - \frac{1}{2}\beta^2}}\|\mathbf{g}(\mathbf{x}^*)\|,$$

where β is defined in Assumption 1.

Proof. Fix $t \geq 1$. By part 2 in Lemma 6, we have

$$\sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) \leq tf(\mathbf{x}^*) + \alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 + \frac{\alpha}{2\alpha - \beta^2}\|\mathbf{g}(\mathbf{x}^*)\|^2 - \frac{1}{2}\|\mathbf{Q}(t)\|^2.$$

By Lemma 8, we have

$$\sum_{\tau=0}^{t-1} f(\mathbf{x}(\tau)) \geq tf(\mathbf{x}^*) - \|\boldsymbol{\lambda}^*\|\|\mathbf{Q}(t)\|.$$

Combining the last two inequalities and cancelling the common term $tf(\mathbf{x}^*)$ on both

sides yields

$$\begin{aligned}
&\frac{1}{2}\|\mathbf{Q}(t)\|^2 - \left(\alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 + \frac{\alpha}{2\alpha-\beta^2}\|\mathbf{g}(\mathbf{x}^*)\|^2\right) \le \|\boldsymbol{\lambda}^*\|\|\mathbf{Q}(t)\| \\
\Rightarrow &\left(\|\mathbf{Q}(t)\| - \|\boldsymbol{\lambda}^*\|\right)^2 \le \|\boldsymbol{\lambda}^*\|^2 + 2\alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 + \frac{\alpha}{\alpha-\frac{1}{2}\beta^2}\|\mathbf{g}(\mathbf{x}^*)\|^2 \\
\Rightarrow &\|\mathbf{Q}(t)\| \le \|\boldsymbol{\lambda}^*\| + \sqrt{\|\boldsymbol{\lambda}^*\|^2 + 2\alpha\|\mathbf{x}^* - \mathbf{x}(-1)\|^2 + \frac{\alpha}{\alpha-\frac{1}{2}\beta^2}\|\mathbf{g}(\mathbf{x}^*)\|^2} \\
\overset{(a)}{\Rightarrow} &\|\mathbf{Q}(t)\| \le 2\|\boldsymbol{\lambda}^*\| + \sqrt{2\alpha}\|\mathbf{x}^* - \mathbf{x}(-1)\| + \sqrt{\frac{\alpha}{\alpha-\frac{1}{2}\beta^2}}\|\mathbf{g}(\mathbf{x}^*)\|,
\end{aligned}$$

where (a) follows from the basic inequality $\sqrt{a+b+c} \le \sqrt{a} + \sqrt{b} + \sqrt{c}$ for any $a, b, c \ge 0$. ∎

THEOREM 2 (Constraint Violations). *Let $\mathbf{x}^*$ be an optimal solution of the problem (1)-(3) and $\boldsymbol{\lambda}^*$ be a Lagrange multiplier vector satisfying Assumption 2. If $\alpha > \frac{\beta^2}{2}$ in Algorithm 1, then for all $t \ge 1$, the constraint functions satisfy*

$$g_k(\overline{\mathbf{x}}(t)) \le \frac{1}{t}\left(2\|\boldsymbol{\lambda}^*\| + \sqrt{2\alpha}\|\mathbf{x}^* - \mathbf{x}(-1)\| + \sqrt{\frac{\alpha}{\alpha-\frac{1}{2}\beta^2}}\|\mathbf{g}(\mathbf{x}^*)\|\right), \forall k \in \{1, 2, \ldots, m\},$$

where β is defined in Assumption 1.

Proof. Fix $t \ge 1$ and $k \in \{1, 2, \ldots, m\}$. Recall that $\overline{\mathbf{x}}(t) = \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbf{x}(\tau)$. Thus,

$$\begin{aligned}
g_k(\overline{\mathbf{x}}(t)) &\overset{(a)}{\le} \frac{1}{t}\sum_{\tau=0}^{t-1} g_k(\mathbf{x}(\tau)) \\
&\overset{(b)}{\le} \frac{Q_k(t)}{t} \\
&\le \frac{\|\mathbf{Q}(t)\|}{t} \\
&\overset{(c)}{\le} \frac{1}{t}\left(2\|\boldsymbol{\lambda}^*\| + \sqrt{2\alpha}\|\mathbf{x}^* - \mathbf{x}(-1)\| + \sqrt{\frac{\alpha}{\alpha-\frac{1}{2}\beta^2}}\|\mathbf{g}(\mathbf{x}^*)\|\right),
\end{aligned}$$

where (a) follows from the convexity of $g_k(\mathbf{x}), k \in \{1, 2, \ldots, m\}$ and Jensen's inequality; (b) follows from Lemma 7; and (c) follows from Lemma 9. ∎

3.4. Convergence Rate of Algorithm 1. The next theorem summarizes the last two subsections.

THEOREM 3. *Let $\mathbf{x}^*$ be an optimal solution of the problem (1)-(3) and $\boldsymbol{\lambda}^*$ be a Lagrange multiplier vector satisfying Assumption 2. If $\alpha > \frac{\beta^2}{2}$ in Algorithm 1, then for all $t \ge 1$, we have*

$$\begin{aligned}
f(\overline{\mathbf{x}}(t)) \le& f(\mathbf{x}^*) + \frac{\alpha}{t}\|\mathbf{x}^* - \mathbf{x}(-1)\|^2, \\
g_k(\overline{\mathbf{x}}(t)) \le& \frac{1}{t}\left(2\|\boldsymbol{\lambda}^*\| + \sqrt{2\alpha}\|\mathbf{x}^* - \mathbf{x}(-1)\| + \sqrt{\frac{\alpha}{\alpha-\frac{1}{2}\beta^2}}\|\mathbf{g}(\mathbf{x}^*)\|\right), \forall k \in \{1, 2, \ldots, m\},
\end{aligned}$$

where β is defined in Assumption 1. In summary, Algorithm 1 ensures error decays like $O(1/t)$ and provides an ϵ-approximate solution with convergence time $O(1/\epsilon)$.

4. Application: Decentralized Network Utility Maximization. This section considers the application of Algorithm 1 to decentralized multipath network utility maximization problems.

4.1. Decentralized Multipath Flow Control. Network flow control can be formulated as the convex optimization of maximizing network utility subject to link capacity constraints [8]. In this view, many existing TCP protocols can be interpreted as distributed solutions to network utility maximization (NUM) problems [10].

In single path network flow control problems, if the utility functions are strictly convex, work [11] shows that the dual subgradient algorithm (with convergence rate $O(1/\sqrt{t})$) can yield a distributed flow control algorithm. If the utility functions are only convex but not necessarily strictly convex, the DPP algorithm (or dual subgradient algorithm with primal averaging) can yield a distributed flow control algorithm with an $O(1/\sqrt{t})$ convergence rate [15, 13, 17]. If utility functions are strongly convex, a faster network flow control algorithm with an $O(1/t)$ convergence rate is proposed in [2]. A recent work [23] shows that the distributed network flow control based on the DPP algorithm also has convergence rate $O(1/t)$ if utility functions are strongly convex. Other Newton method based distributed algorithms for network flow control with strictly convex utility functions are considered in [22].

However, in multipath network flow control problems, even if the utility function is strictly or strongly convex with respect to the source rate, it is no longer strictly or strongly convex with respect to path rates. Thus, many of the above algorithms requiring strict or strong convexity can no longer be applied. The DPP algorithm can still be applied but the convergence rate is only $O(1/\sqrt{t})$. Distributed algorithms based on the primal-dual subgradient method, also known as the Arrow-Hurwicz-Uzawa subgradient method, have been considered in [20]. However, the convergence rate[3] of the primal-dual subgradient method for general convex programs without strong convexity is known to be $O(1/\sqrt{t})$ [14].

As shown in the previous sections, Algorithm 1 has an $O(1/t)$ convergence rate for general convex programs and yields a distributed algorithm if the objective function and constraint functions are separable. The next subsection applies Algorithm 1 to the multipath network flow control problem. The resulting algorithm has structural properties and implementation characteristics similar to the subgradient-based algorithm of [10] and to the DPP algorithm of [23], but has additional decision variables due to the multipath formulation. (The algorithms in [10, 23] rely on strict and strong convexity of the objective function, respectively, and apply only to single path situations.)

4.2. Decentralized Multipath Flow Control Based on Algorithm 1. Suppose there are S sources enumerated by $\mathcal{S} = \{1, 2, \ldots, S\}$ and L links enumerated by $\mathcal{L} = \{1, 2, \ldots, L\}$. Each link $l \in \mathcal{L}$ has a link capacity of c_l bits/slot. Each source sends data from a specific origin to a specific destination, and has multiple path options. The paths for each source can use overlapping links, and they can also overlap with paths of other sources. Further, two distinct sources can have identical paths. However, it is useful to give distinct path indices to paths associated with each source (even if their corresponding paths are physically the same). Specifically, for each source s, define $\mathcal{P}_s$ as the set of path indices used by source s. The index sets

[3]Similar to the dual subgradient method, the primal-dual subgradient method has an $O(1/\sqrt{t})$ convergence rate for general convex programs in the sense that $O(1/\epsilon^2)$ iterations are required to obtain an ϵ-approximate solution. However, the primal-dual subgradient method does not have vanishing errors.

$\mathcal{P}_1, \ldots, \mathcal{P}_S$ are disjoint and $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \cdots \cup \mathcal{P}_S = \mathcal{K} = \{1, 2, \ldots, K\}$, where K is the number of path indices.

For each source s, let $U_s(y_s)$ be a real-valued, concave, continuous and nondecreasing utility function defined for $y_s \geq 0$. This represents the satisfaction source s receives by communicating with a total rate of y_s, where the total rate sums over all paths it used.

Note that different paths can share links in common. Define $\mathcal{D}_l \subseteq \mathcal{K}$ as the set of paths that use link l and $\mathcal{E}_k \subseteq \mathcal{L}$ as the set of links used by path k. Let $\mathbf{x} = [x_1, \ldots, x_K]^T$ be the vector that specifies the flow rate on each path; and $\mathbf{y} = [y_1, \ldots, y_S]^T$ be the vector that specifies the rate of each source.

The goal is to allocate flow rates on each path so that no link is overloaded and network utility is maximized. This multipath network utility maximization problem can be formulated as follows:

$$\text{maximize} \quad \sum_{s=1}^{S} U_s(y_s) \tag{13}$$

$$\text{subject to} \quad \sum_{k \in \mathcal{D}_l} x_k \leq c_l, \forall l \in \mathcal{L}, \tag{14}$$

$$y_s = \sum_{k \in \mathcal{P}_s} x_k, \forall s \in \mathcal{S}, \tag{15}$$

$$0 \leq x_k \leq x_k^{\max}, \forall k \in \mathcal{K}, \tag{16}$$

$$0 \leq y_s \leq y_s^{\max}, \forall s \in \mathcal{S}, \tag{17}$$

where $x^{\max}$ and $y^{\max}$ are the allowed maximum path rate and maximum source rate, respectively. The expression (13) represents the network utility; inequality (14) specifies the link capacity constraints; and equality (15) enforces the definition of y_s. The linear equality constraints (15) can be formally treated by writing each one as two linear inequality constraints. However, since the utility functions $U_s(\cdot)$ are nondecreasing and seek to maximize the y_s values, it is clear that the above problem is equivalent to the following:

$$\text{minimize} \quad -\sum_{s=1}^{S} U_s(y_s) \tag{18}$$

$$\text{subject to} \quad \sum_{k \in \mathcal{D}_l} x_k \leq c_l, \forall l \in \mathcal{L}, \tag{19}$$

$$y_s \leq \sum_{k \in \mathcal{P}_s} x_k, \forall s \in \mathcal{S}, \tag{20}$$

$$0 \leq x_k \leq x_k^{\max}, \forall k \in \mathcal{K}, \tag{21}$$

$$0 \leq y_s \leq y_s^{\max}, \forall s \in \mathcal{S}. \tag{22}$$

Note that the constraints (19)-(20) are linear and can be written as $\mathbf{A} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \leq \begin{bmatrix} \mathbf{c} \\ \mathbf{0} \end{bmatrix}$, where $\mathbf{A}$ is an $(L+S) \times (K+S)$ matrix of which each entry is in $\{0, \pm 1\}$. This inequality constraint $\mathbf{A} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \leq \begin{bmatrix} \mathbf{c} \\ \mathbf{0} \end{bmatrix}$ can be treated as the inequality constraint (2) defined as $\mathbf{g}(\mathbf{z}) = \mathbf{A}\mathbf{z} - \mathbf{b} \leq \mathbf{0}$ with $\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} \mathbf{c} \\ \mathbf{0} \end{bmatrix}$ in the general

convex program (1)-(3). The box constraints (21)-(22) can be treated as the set constraint (3) in the general convex program (1)-(3). Thus, the multipath network utility maximization problem (18)-(22) is a special case of the general convex program (1)-(3).

The Lipschitz continuity of $\mathbf{g}(\mathbf{z})$ is summarized in the next lemma.

LEMMA 10. *If the constraints* (19)-(20) *are written as* $\mathbf{g}(\mathbf{z}) \leq \mathbf{0}$, *then we have*

1. *The function* $\mathbf{g}(\mathbf{z})$ *is Lipschitz continuous with modulus*
$$\beta \leq \sqrt{S + K + \sum_{k \in \mathcal{K}} d_k},$$
where d_k *is the length, i.e., number of hops, of path* $k \in \mathcal{K}$.
2. *The function* $\mathbf{g}(\mathbf{z})$ *is Lipschitz continuous with modulus* $\beta \leq \sqrt{(L+1)K + S}$.

Proof.

1. Recall that $\mathbf{g}(\mathbf{z}) = \mathbf{A}\mathbf{z} - \mathbf{b}$ is Lipschitz continuous with modulus $\beta = \sigma_{\max}(\mathbf{A})$, where $\sigma_{\max}(\mathbf{A})$ is the maximum singular value of matrix $\mathbf{A}$, and a simple upper bound of $\sigma_{\max}(\mathbf{A})$ is the Frobenius norm given by $\|\mathbf{A}\|_F = \mathrm{tr}(\mathbf{A}^T\mathbf{A}) = \sqrt{\sum_{i,j} A_{ij}^2}$. Note that $\mathbf{A}$ can be written as
$$\mathbf{A} = \begin{bmatrix} \mathbf{R} & \mathbf{0}_{L \times S} \\ -\mathbf{T} & \mathbf{I}_S \end{bmatrix},$$
where $\mathbf{R}$ is a $\{0,1\}$ matrix of size $L \times K$, $\mathbf{0}_{L \times S}$ is an $L \times S$ zero matrix, $\mathbf{T}$ is a $\{0,1\}$ matrix of size $S \times K$ and $\mathbf{I}_S$ is an $S \times S$ identity matrix. Note that the (l,k)-th entry of $\mathbf{R}$ is 1 if and only if path k uses link l; the (s,k)-th entry of $\mathbf{T}$ is 1 if and only if path k is a path for source s, i.e, $k \in \mathcal{P}_s$. Matrix $\mathbf{R}$ has $\sum_{k \in \mathcal{K}} d_k$ non-zero entries since each column has exactly d_k non-zero entries. Matrix $\mathbf{T}$ has K non-zero entries since there are in total K network paths. Thus, matrix $\mathbf{A}$ in total has $S + K + \sum_{k \in \mathcal{K}} d_k$ non-zero entries whose absolute values are equal to 1. It follows that $\beta \leq \sqrt{S + K + \sum_{k \in \mathcal{K}} d_k}$.
2. This part follows from the fact that the length of each path is at most L. ∎

Note that the second bound in the above lemma is more loose than the first one but holds regardless of the flow path configurations in the network. A straightforward application of Algorithm 1 yields the following decentralized network flow control algorithm described in Algorithm 2. Similar to the flow control based on the dual subgradient algorithm, Algorithm 2 is decentralized and can be easily implemented within the current TCP protocols [10].

By Theorem 3, if we choose $\alpha \geq \frac{1}{2}\big(S + K + \sum_{k \in \mathcal{K}} d_k\big)$, then for any $t \geq 1$,

$$\sum_{s=1}^{S} U_s(\overline{y}_s(t)) \geq \sum_{s=1}^{S} U_s(y_s^*) - O(1/t), \tag{23}$$

$$\sum_{k \in \mathcal{D}_l} \overline{x}_k(t) \leq c_l + O(1/t), \qquad \forall l \in \mathcal{L}, \tag{24}$$

$$\overline{y}_s(t) \leq \sum_{k \in \mathcal{P}_s} \overline{x}_k(t) + O(1/t), \qquad \forall s \in \mathcal{S}, \tag{25}$$

where $(\mathbf{x}^*, \mathbf{y}^*)$ is an optimal solution of the problem (18)-(22). Note that if Algorithm 2 has been run for a sufficiently long time and we are satisfied with the current

Algorithm 2

- Initialization: Let $x_k(-1) \in [0, x_k^{\max}], \forall k \in \mathcal{K}$ be arbitrary, $y_s(-1) \in [0, y_s^{\max}], \forall s \in \mathcal{S}$ be arbitrary, $Q_l(0) = \max\{0, -\sum_{k\in\mathcal{D}_l} x_k(-1) + c_l\}, \forall l \in \mathcal{L}$, $Y_l(0) = Q_l(0) + \sum_{k\in\mathcal{D}_l} x_k(-1) - c_l, \forall l \in \mathcal{L}$, $R_s(0) = \max\{0, \sum_{k\in\mathcal{P}_s} x_k(-1) - y_s(-1)\}, \forall s \in \mathcal{S}$ and $Z_s(0) = R_s(0) + y_s(-1) - \sum_{k\in\mathcal{P}_s} x_k(-1), \forall s \in \mathcal{S}$.
- Each link l's algorithm: At each time $t \in \{0, 1, \ldots\}$, link l does the following:
  1. Receive the path rates $x_k(t)$ that use link l. Update $Q_l(t)$ via:
  $$Q_l(t+1) = \max\Big\{ - \sum_{k\in\mathcal{D}_l} x_k(t) + c_l, Q_l(t) + \sum_{k\in\mathcal{D}_l} x_k(t) - c_l\Big\}.$$
  2. The price of this link is given by $Y_l(t+1) = Q_l(t+1) + \sum_{k\in\mathcal{D}_l} x_k(t) - c_l$.
  3. Communicate the link price $Y_l(t+1)$ to sources that use link l.
- Each source s's algorithm: At each time $t \in \{0, 1, \ldots\}$, source s does the following:
  1. Receive from the network the link prices $Y_l(t)$ for all links l that are used by any path of source s.
  2. Update the path rates $x_k, k \in \mathcal{P}_s$ by
  $$\begin{aligned} x_k(t) &= \operatorname*{argmin}_{0\le x_k\le x_k^{\max}} \Big\{ \big[\sum_{l\in\mathcal{E}_k} Y_l(t) - Z_s(t)\big] x_k + \alpha(x_k - x_k(t-1))^2 \Big\} \\ &= \Big[x_k(t-1) - \frac{1}{2\alpha}\big(\sum_{l\in\mathcal{E}_k} Y_l(t) - Z_s(t)\big)\Big]_0^{x_k^{\max}}, \end{aligned}$$
  where $[z]_a^b = \min\{\max\{z, a\}, b\}$.
  3. Communicate the path rates $x_k(t), k \in \mathcal{P}_s$ to all links that are used by path k.
  4. Update the source rate $y_s(t)$ by
  $$y_s(t) = \operatorname*{argmin}_{0\le y_s\le y_s^{\max}} \Big\{ - U_s(y_s) + Z_s(t) y_s + \alpha(y_s - y_s(t-1))^2 \Big\},$$
  which usually has a closed form solution for differentiable utilities by taking derivatives.
  5. Update virtual queue $R_s(t)$ and source price $Z_s(t)$ locally by
  $$\begin{aligned} R_s(t+1) &= \max\Big\{ - y_s(t) + \sum_{k\in\mathcal{P}_s} x_k(t), R_s(t) + y_s(t) - \sum_{k\in\mathcal{P}_s} x_k(t)\Big\}, \\ Z_s(t+1) &= R_s(t+1) + y_s(t) - \sum_{k\in\mathcal{P}_s} x_k(t). \end{aligned}$$

performance, then we can fix $\mathbf{x} = \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbf{x}(\tau)$ and $\mathbf{y} = \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbf{y}(\tau)$ such that the performance is still within $O(1/t)$ sub-optimality for all future time.

4.3. Decentralized Joint Flow and Power Control. Since Algorithm 1 allows for general nonlinear convex constraint functions, it can also be applied to solve the joint flow and power control problem. In this case, the capacity of each link l is not fixed but depends concavely on a power allocation variable p_l. Assuming each

link capacity is logarithmic in p_l results in the following problem:

$$\text{maximize} \quad \sum_{s=1}^{S} U_s(y_s) - \sum_{l=1}^{L} V_l(p_l) \tag{26}$$

$$\text{subject to} \quad \sum_{k\in\mathcal{D}_l} x_k \le \log(1+p_l), \forall l \in \mathcal{L}, \tag{27}$$

$$y_s \le \sum_{k\in\mathcal{P}_s} x_k, \forall s \in \mathcal{S}, \tag{28}$$

$$0 \le x_k \le x_k^{\max}, \forall k \in \mathcal{K}, \tag{29}$$

$$0 \le y_s \le y_s^{\max}, \forall s \in \mathcal{S}, \tag{30}$$

$$0 \le p_l \le p_l^{\max}, \forall l \in \mathcal{L}, \tag{31}$$

where p_l is the power allocated at link l, $\log(1+p_l)$ is the corresponding link capacity as a function of p_l, and $V_l(p_l)$ is the associated power cost (assumed to be a convex function of p_l). A decentralized joint flow and power control algorithm for this problem can be similarly developed by applying Algorithm 1.

5. Numerical Results. This section considers numerical experiments to verify the convergence rate results shown in this paper.

5.1. Decentralized Multiplath Flow Control. Consider the simple multipath network flow problem described in Figure 1. Assume each link has capacity 1. Let y_1, y_2 and y_3 be the data rates of source 1, 2 and 3; $x_1, x_2, x_3, x_4, x_5, x_6$ and x_7 be the data rates of the paths indicated in the figure; and the network utility be maximizing $\log(y_1) + 2\log(y_2) + 2\log(y_3)$. The NUM problem can be formulated as follows:

$$\begin{aligned} &\text{maximize} && \log(y_1) + 2\log(y_2) + 2\log(y_3) \\ &\text{subject to} && \mathbf{Rx} \le \mathbf{c}, \\ &&& \mathbf{y} \le \mathbf{Tx}, \\ &&& 0 \le x_i \le x_i^{\max}, i \in \{1, 2, \ldots, 4\}, \\ &&& 0 \le y_i \le y_i^{\max}, i \in \{1, 2, 3\}, \end{aligned}$$

where $\mathbf{R} = \begin{bmatrix} 1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0\\ 0&0&1&0&0&0&0\\ 1&0&1&0&0&0&0\\ 0&1&0&1&0&0&0\\ 0&0&0&0&1&0&0\\ 0&0&0&0&1&1&0\\ 0&0&0&0&0&1&0\\ 0&0&0&0&0&0&1 \end{bmatrix}$, $\mathbf{c} = \begin{bmatrix} 1\\1\\1\\1\\1\\1\\1\\1\\1 \end{bmatrix}$, $\mathbf{T} = \begin{bmatrix} 1&1&0&0&0&0&0\\ 0&0&1&1&1&0&0\\ 0&0&0&0&0&1&1 \end{bmatrix}$.

The optimal value to this NUM problem is $f^* = 1.65687$.

To verify the convergence of Algorithm 2, Figure 2 shows the values of objective and constraint functions yielded by Algorithm 2 with $\alpha = \frac{1}{2}\big(K+S+\sum_{k\in\mathcal{K}} d_k\big)+1 = 10$ and $\mathbf{x}(-1) = \mathbf{0}$. (By writing constraints $\mathbf{Rx} \le \mathbf{c}$ and $\mathbf{y} \le \mathbf{Tx}$ in the compact form $\mathbf{Az} \le \mathbf{b}$, it can be checked that $\beta = \sigma_{\max}(\mathbf{A}) = 2.4307$. If we choose a smaller α, e.g., $\alpha = \frac{1}{2}\beta^2 + 1 = 3.9543$, then Algorithm 2 converges even faster. In this simulation, we choose a loose α whose value can be easily estimated from Lemma 10 without

knowing the detailed network topology.) We also compare our algorithm with the dual subgradient algorithm (with primal averaging) with step size 0.01 in [13]. (Or equivalently, the DPP algorithm with $V = 100$ in [15, 17, 23].) Recall that the dual subgradient algorithm in [15, 13, 17, 23] does not converge to an exact optimal solution but only converges to an approximate solution with an error level determined by the step size. In contrast, our algorithm can eventually converge to the exact optimality. Figure 2 shows that Algorithm 2 converges faster than the dual subgradient algorithm with primal averaging.

To verify the convergence rate of Algorithm 2, Figure 3 plots $f(\overline{\mathbf{x}}(t)) - f^*$, all constraint values, function $1/t$, and bounds from Theorem 3 with both x-axis and y-axis in $\log_{10}$ scales. It can be observed that the curves of $f(\overline{\mathbf{x}}(t)) - f^*$ and all the source rate constraint values are parallel to the curve of $1/t$ for large t. Note that all the link capacity constraints are satisfied early (i.e., negative), and hence are not drawn in $\log_{10}$ scales. Figure 3 verifies that the error of Algorithm 2 decays like $O(1/t)$ and suggests that it is actually $\Theta(1/t)$ for this multipath NUM problem.

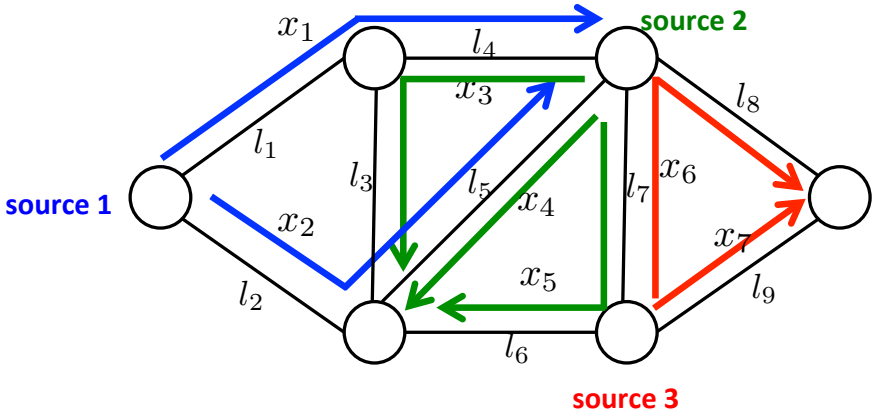

FIG. 1. *A simple multipath NUM problem with 3 sources and 7 paths.*

5.2. Decentralized Joint Flow and Power Control. Consider the joint flow and power control over the same network described in Figure 1. We assume the power cost of each link is given by $V_l(p_l) = 0.25p_l$. The optimal value of this joint flow and power control problem is $f^* = -0.521318$.

To verify the convergence of Algorithm 1, Figure 4 shows the values of objective and constraint functions yielded by Algorithm 1 with $\alpha = 10$, $\mathbf{x}(-1) = \mathbf{0}$, $\mathbf{y}(-1) = \mathbf{0}$ and $\mathbf{p}(-1) = \mathbf{0}$. (In fact, by writing constraints $\mathbf{Rx} \le \log(1 + \mathbf{p})$ and $\mathbf{y} \le \mathbf{Tx}$ in the compact form $\mathbf{g}(\mathbf{z}) \le \mathbf{0}$, it can be checked that $\beta = 2.5229$. If we choose a smaller α, e.g., $\alpha = \frac{1}{2}\beta^2 + 1 = 4.1826$, then Algorithm 1 converges even faster.) We also compare our algorithm with the dual subgradient algorithm (with primal averaging) with step size 0.01 in [15, 13, 17, 23]. Figure 4 shows that Algorithm 1 converges faster than the dual subgradient algorithm with primal averaging.

To verify the convergence rate of Algorithm 1, Figure 5 plots $f(\overline{\mathbf{x}}(t)) - f^*$, all constraint values, function $1/t$, and bounds from Theorem 3 with both x-axis and y-axis in $\log_{10}$ scales. It can be observed that the curves of $f(\overline{\mathbf{x}}(t)) - f^*$ and all the source rate constraint values are parallel to the curve of $1/t$ for large t.

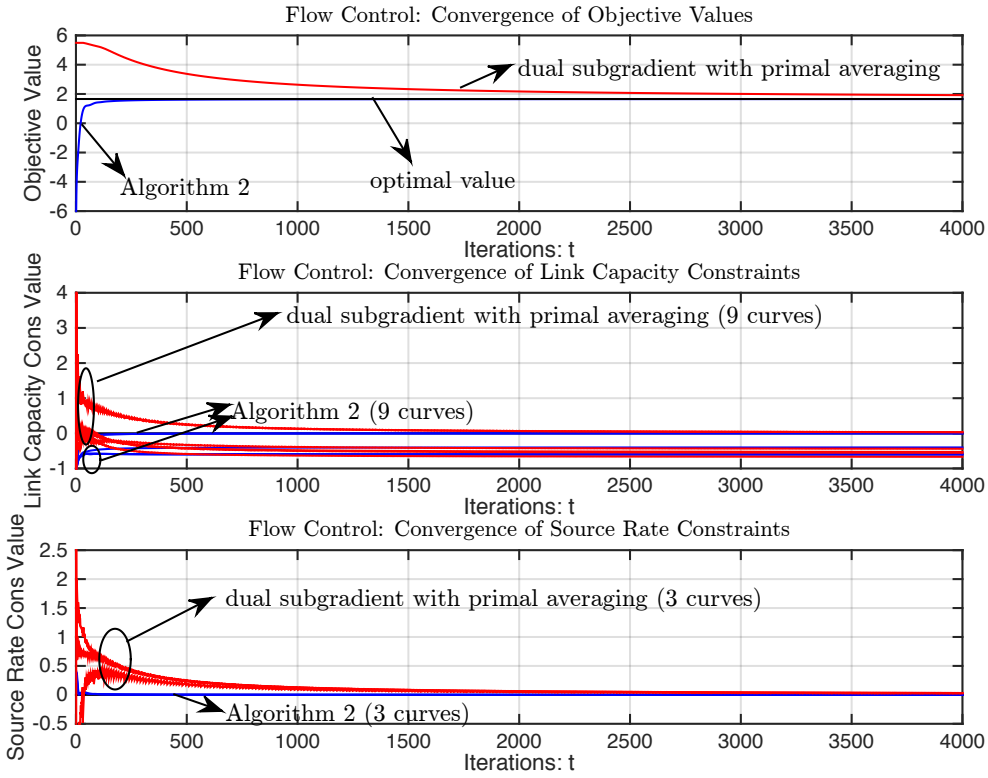

FIG. 2. *The convergence of Algorithm 2 and the dual subgradient algorithm with primal averaging for a multipath flow control problem.*

5.3. Quadratic Programs. Consider the following quadratic program

$$\begin{aligned} \min \quad & \mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{d}^T\mathbf{x} \le e \\ & \mathbf{x}^{\min} \le \mathbf{x} \le \mathbf{x}^{\max} \end{aligned}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are positive semidefinite to ensure the convexity of the quadratic program.

We randomly generate a large scale example where $\mathbf{x} \in \mathbb{R}^{100}$, $\mathbf{P} \in \mathbb{R}^{100\times 100}$ is diagonal with entries from uniform $[0, 4]$, $\mathbf{c} \in \mathbb{R}^{100}$ with entries from uniform $[-15, 20]$, $\mathbf{Q} \in \mathbb{R}^{100\times 100}$ is diagonal with entries from uniform $[0, 1]$, $\mathbf{d} \in \mathbb{R}^{100}$ with entries from uniform $[-1, 1]$, e is a scalar from uniform $[4, 5]$, $\mathbf{x}^{\min} = \mathbf{0}$ and $\mathbf{x}^{\max} = \mathbf{1}$. Note that Algorithm 1 and the dual subgradient algorithm (with primal averaging) can deal with general semidefinite positive matrices $\mathbf{P}$ and $\mathbf{Q}$. However, if $\mathbf{P}$ and $\mathbf{Q}$ are diagonal or block diagonal, then the primal update in both algorithms can be decomposed into independent smaller problems and hence has extremely low complexity.

To verify the convergence of Algorithm 1, Figure 6 shows the values of objective and constraint functions yielded by Algorithm 1 with $\alpha = \frac{1}{2}\beta^2 + 1$, where β the is Lipschitz modulus of the constraint function, and $\mathbf{x}(-1) = \mathbf{x}^{\min}$. We also compare our algorithm with the dual subgradient algorithm (with primal averaging) with step

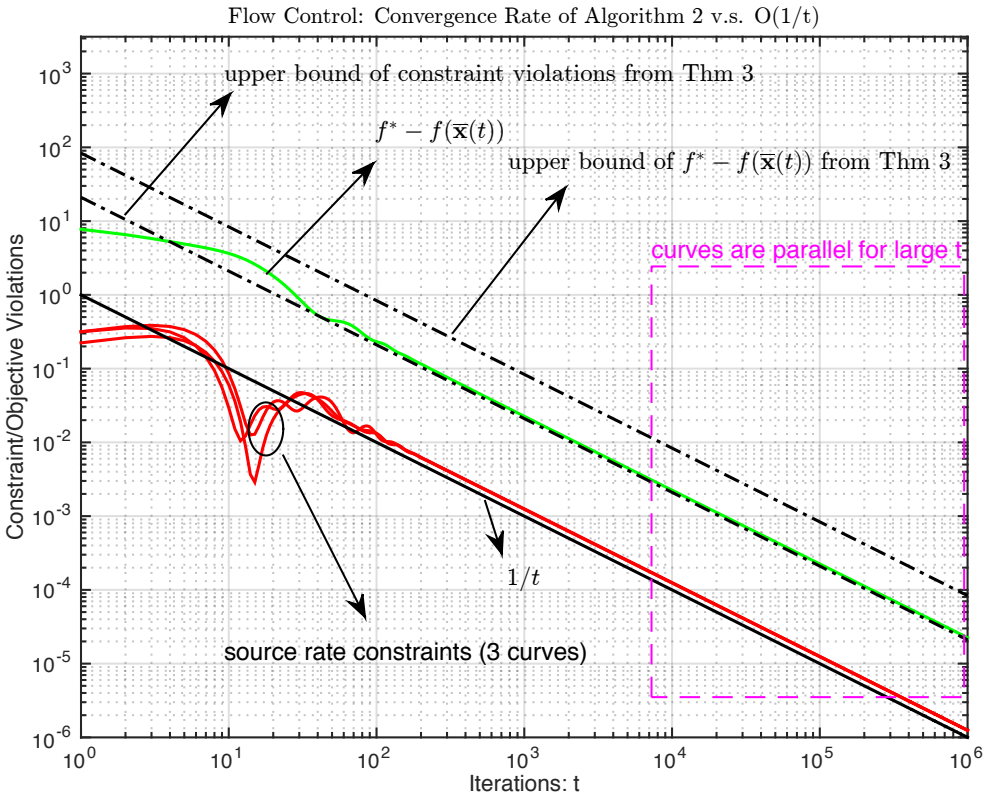

FIG. 3. *The convergence rate of Algorithm 2 for a multipath flow control problem.*

size 0.01 in [15, 13, 17, 23]. Figure 6 shows that Algorithm 1 converges faster than the dual subgradient algorithm with primal averaging.

To verify the convergence rate of Algorithm 1, Figure 7 plots $f(\overline{\mathbf{x}}(t)) - f^*$, function $1/t$, and the bound from Theorem 3 with both x-axis and y-axis in $\log_{10}$ scales. It can be observed that the curves of $f(\overline{\mathbf{x}}(t)) - f^*$ and all the source rate constraint values are parallel to the curve of $1/t$ for large t. Note that the constraint violation is not plotted since the constraint function is satisfied for all iterations as observed in Figure 6.

6. Conclusions. This paper proposes a novel but simple algorithm to solve convex programs with a possibly non-differentiable objective function and Lipschitz continuous constraint functions. The new algorithm has a parallel implementation when the objective function and constraint functions are separable. The convergence rate of the proposed algorithm is shown to be $O(1/t)$. This is faster than the $O(1/\sqrt{t})$ convergence rate of the dual subgradient algorithm with primal averaging. The ADMM algorithm has the same $O(1/t)$ convergence rate but can only deal with linear equality constraint functions. The new algorithm is further applied to solve multipath network flow control problems and yields a decentralized flow control algorithm which converges faster than existing dual subgradient or primal-dual subgradient based flow control algorithms. The $O(1/t)$ convergence rate of the proposed algorithm is also verified by numerical experiments.

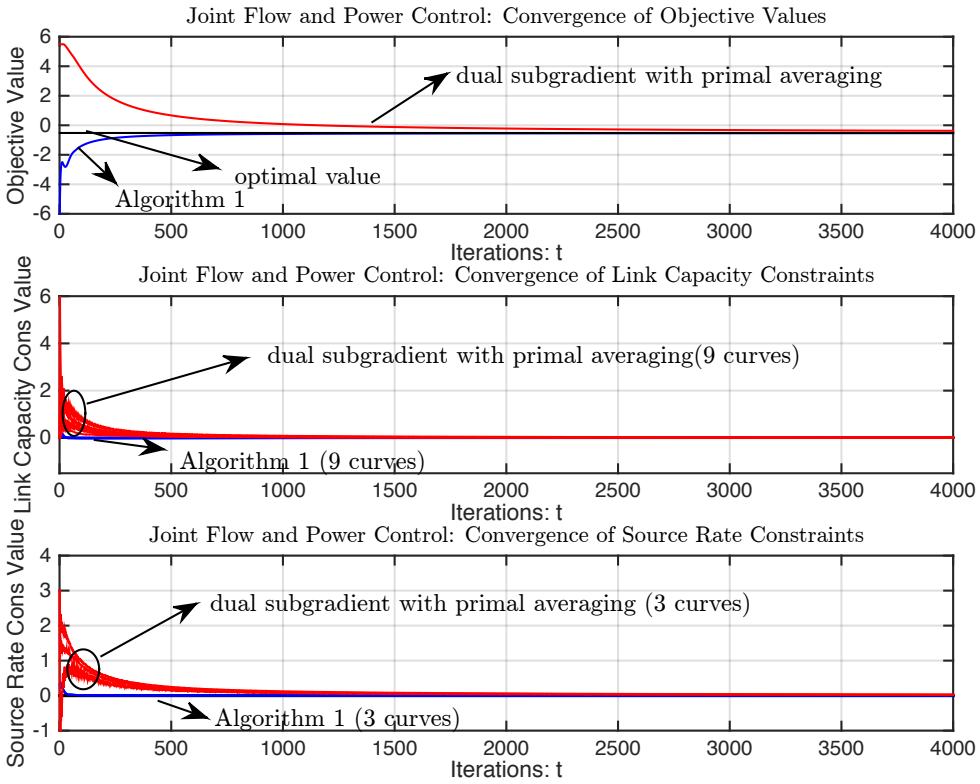

FIG. 4. *The convergence of Algorithm 1 and the dual subgradient algorithm with primal averaging for a multipath joint flow and power control problem.*

REFERENCES

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, Wiley-Interscience, 2006.

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

[3] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, second ed., 1999.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning, 3 (2011), pp. 1–122.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[6] B. He and X. Yuan, *On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method*, SIAM Journal on Numerical Analysis, 50 (2012), pp. 700–709.

[7] J.-B. Hiriart-Urruty and C. Lemaréchal, *Fundamentals of Convex Analysis*, Springer, 2001.

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

[9] T.-Y. Lin, S.-Q. Ma, and S.-Z. Zhang, *On the sublinear convergence rate of multi-block ADMM*, Journal of the Operations Research Society of China, 3 (2015), pp. 251–274.

[10] S. H. Low, *A duality model of TCP and queue management algorithms*, IEEE/ACM Transactions on Networking, 11 (2003), pp. 525–536.

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

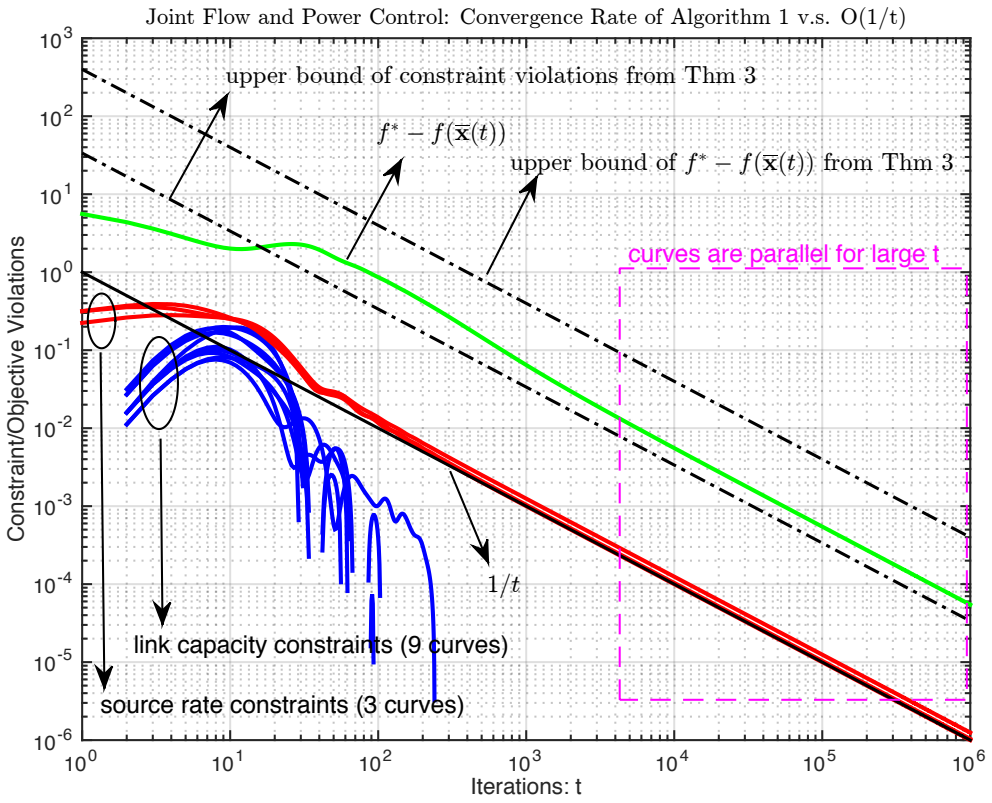

FIG. 5. *The convergence rate of Algorithm 1 for a multipath joint flow and power control problem.*

[12] I. NECOARA AND V. NEDELCU, *Rate analysis of inexact dual first-order methods application to dual decomposition*, IEEE Transactions on Automatic Control, 59 (2014), pp. 1232–1243.

[13] A. NEDIĆ AND A. OZDAGLAR, *Approximate primal solutions and rate analysis for dual subgradient methods*, SIAM Journal on Optimization, 19 (2009), pp. 1757–1780.

[14] A. NEDIĆ AND A. OZDAGLAR, *Subgradient methods for saddle-point problems*, Journal of Optimization Theory and Applications, 142 (2009), pp. 205–228.

[15] M. J. NEELY, *Distributed and secure computation of convex programs over a network of connected processors*, in DCDIS Conference Guelph, July 2005.

[16] M. J. NEELY, *Stochastic Network Optimization with Application to Communication and Queueing Systems*, Morgan & Claypool Publishers, 2010.

[17] M. J. NEELY, *A simple convergence time analysis of drift-plus-penalty for stochastic optimization and convex programs*, arXiv:1412.0791, (2014).

[18] Y. NESTEROV, *Introductory Lectures on Convex Optimization: A Basic Course*, Springer Science & Business Media, 2004.

[19] N. PARIKH AND S. BOYD, *Proximal algorithms*, Foundations and Trends in Optimization, 1 (2013), pp. 123–231.

[20] W.-H. WANG, M. PALANISWAMI, AND S. H. LOW, *Optimal flow control and routing in multi-path networks*, Performance Evaluation, 52 (2003), pp. 119–132.

[21] E. WEI AND A. OZDAGLAR, *On the $O(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers*, in Proceedings of IEEE Global Conference on Signal and Information Processing, 2013.

[22] E. WEI, A. OZDAGLAR, AND A. JADBABAIE, *A distributed Newton method for network utility maximization–I: algorithm*, IEEE Transactions on Automatic Control, 58 (2013), pp. 2162–2175.

[23] H. YU AND M. J. NEELY, *On the convergence time of the drift-plus-penalty algorithm for*

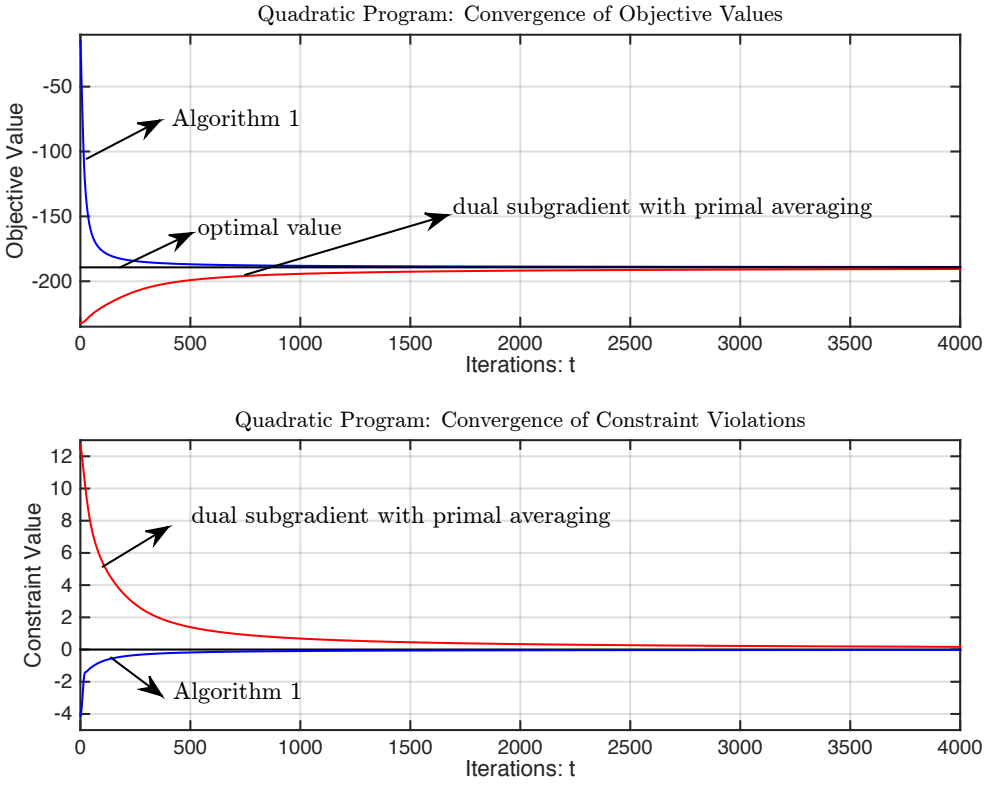

FIG. 6. *The convergence of Algorithm 1 and the dual subgradient algorithm with primal averaging for a quadratic program.*

strongly convex programs, in Proceedings of IEEE Conference on Decision and Control (CDC), 2015.

[24] H. YU AND M. J. NEELY, *A primal-dual type algorithm with the $O(1/t)$ convergence rate for large scale constrained convex programs*, in Proceedings of IEEE Conference on Decision and Control (CDC), 2016.

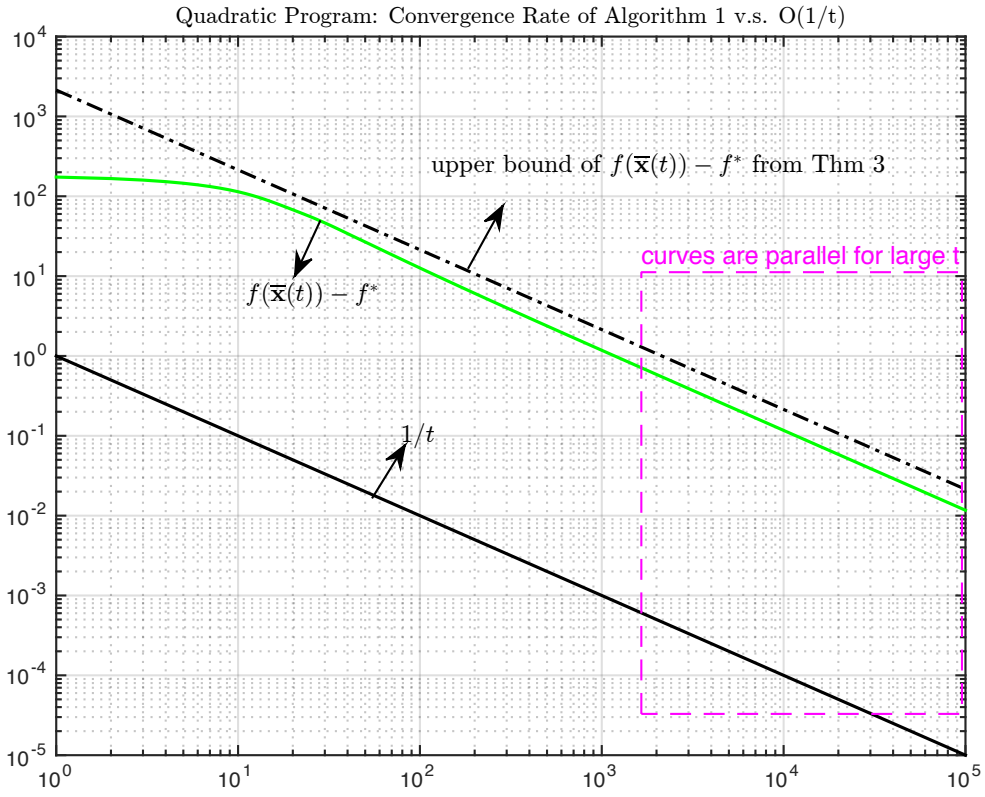

FIG. 7. *The convergence rate of Algorithm 1 for a quadratic program.*