Utility Maximization for Communication Networks with Multi-path Routing

Xiaojun Lin and Ness B. Shroff

***Abstract*— In this paper, we study utility maximization problems for communication networks *where each user (or class) can have multiple alternative paths through the network*. This type of multi-path utility maximization problems appear naturally in several resource allocation problems in communication networks, such as the multi-path flow control problem, the optimal QoS routing problem, and the optimal network pricing problem. We develop a distributed solution to this problem that is amenable to online implementation. We analyze the convergence of our algorithm in both continuous-time and discrete-time, and with and without measurement noise. These analyses provide us with guidelines on how to choose the parameters of the algorithm to ensure efficient network control.**

I. Introduction

In this paper we are concerned with problems of the following form:

$$\max_{\substack{x_{ij} \geq 0,\, m_i \leq \sum_{j=1}^{J(i)} x_{ij} \leq M_i \\ i=1,\ldots,I}} \sum_{i=1}^{I} f_i\Big(\sum_{j=1}^{J(i)} x_{ij}\Big) \tag{1}$$

$$\text{subject to } \sum_{i=1}^{I}\sum_{j=1}^{J(i)} E_{ij}^l x_{ij} \leq R^l, \text{ for all } l = 1,\ldots,L. \tag{2}$$

As we will describe in Section II, optimization problems of this form appear in several resource allocation problems in communication networks, when each user or (class of users) can have multiple alternative paths through the network. Generically, the problem (1) amounts to allocating resources $R^1,\ldots,R^L$ from network components $l = 1,2,\ldots,L$ to users $i = 1,2,\ldots,I$ such that the total system "utility" is maximized. The "utility" function $f_i(\cdot)$ represents the performance, or level of "satisfaction," of user i when a certain amount of resource is allocated to it. In practice, this performance measure can be in terms of revenue, welfare, or admission probability, etc., depending on the problem setting. We assume throughout that $f_i(\cdot)$ is concave. Each user i can have $J(i)$ alternative paths (a *path* consists of a subset of the network components). Let x_{ij} denote the amount of resources allocated to user i on path j. Then the utility $f_i(\sum_{j=1}^{J(i)} x_{ij})$, subject to $m_i \leq \sum_{j=1}^{J(i)} x_{ij} \leq M_i$, is a function of the *sum* of the resources allocated to user i on all paths. Hence, the resources on alternative paths are considered equivalent and interchangeable for user i. The constants E_{ij}^l represent the routing structure of the network: each unit of resource allocated to user i on path j will consume E_{ij}^l units of resource on network component l. ($E_{ij}^l = 0$ for network components that are not on path j of user i.) The inequalities in (2) represent the resource constraints at the network components (hence R^l can be viewed as the capacity of network component l, and $\sum_{i=1}^{I}\sum_{j=1}^{J(i)} E_{ij}^l x_{ij}$ is the total amount of resources consumed at network component l summed over all users and all alternative paths). We assume that $R^l > 0$, $E_{ij}^l \geq 0$, $m_i \geq 0$ and $M_i > 0$ (M_i could possibly be $+\infty$).

We will refer to problem (1) as the *multi-path utility maximization problem*. In this paper, we are interested in solutions to this problem that are amenable to *online* implementation. In Section II, we will identify several resource allocation problems in communication networks that can be modeled as (1), including the multi-path flow control problem, the optimal QoS routing problem, and the optimal pricing problem. Essentially, once the network can support multi-path routing, the resource allocation problem changes from a *single-path* utility maximization problem to a *multi-path* utility maximization problem. As we will soon see, the multi-path nature of the problem leads to several difficulties in constructing solutions suitable for online implementation. One of the main difficulties is that, once some users have multiple alternative paths, the objective function of problem (1) is no longer strictly concave, and hence the dual of the problem may not be differentiable at every point. Note that this lack of strict concavity is mainly due to the linearity $\sum_{j=1}^{J(i)} x_{ij}$. (The objective function in (1) is still not strictly concave even if the utility functions f_i are strictly concave.) On the other hand, the requirement that the solutions must be implementable online also imposes a number of important restrictions on our design space. We outline these restrictions below:

- The solution has to be distributed because these communication networks can be very large and centralized solutions are not scalable.
- In order to lower the communication overhead, the solution has to limit the amount of information exchanged between the users and different network components. For example, a solution that can adjust resource allocation based on online measurements is preferable to one that requires explicit signaling mechanisms to communicate

This work has been partially supported by the NSF grant ANI-0099137, the Indiana 21st Century Fund 651-1285-074 and a grant through the Purdue Research Foundation. An earlier version of this paper has been presented at the 41st Annual Allerton Conference on Communication, Control, and Computing [1].

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA (email: {linx,shroff}@ecn.purdue.edu).

information.

- It is also important that the solution does not require the network components to store and maintain per-user information (or *per-flow* information, as it is referred to in some of the networking literature). Since the number of users sharing a network component can be large, solutions that require maintaining per-user information will be costly and will not scale to large networks.
- In the case where the solution uses online measurements to adjust the resource allocation, the solution should also be resilient to measurement noise due to estimation errors.

In this paper, we develop a distributed solution to the multi-path utility maximization problem. Our distributed solution has the aforementioned attributes desirable for online implementation. The main technical contributions of the paper are as follows:

1) We provide a rigorous analysis of the convergence of our distributed algorithm. This analysis is done without requiring the *two-level convergence structure* that is typical in standard techniques in the convex programming literature for dealing with the lack of strict concavity of the problem. Note that algorithms based on these standard techniques are required to have an *outer level* of iterations where each outer iteration consists of an *inner level* of iterations. For the convergence of this class of algorithms to hold, the inner level of iterations must converge before each outer iteration can proceed. Such a two-level convergence structure may be acceptable for off-line computation, but not suitable for online implementation because in practice it is difficult for the network to decide in a distributive fashion when the inner level of iterations can stop. A main contribution of this paper is to establish the convergence of our distributed algorithm without requiring such a two-level convergence structure.

2) By proving convergence, we are able to provide easy-to-verify bounds on the parameters (i.e., step-sizes) of our algorithm to ensure convergence. Note that when distributed algorithms based on our solution are implemented online, a practically important question is how to choose the parameters of the algorithm to ensure efficient network control. Roughly speaking, the step-sizes used in the algorithm should be small enough to ensure stability and convergence, but not so small such that the convergence becomes unnecessarily slow. The main part of this paper addresses the question of parameter selection by providing a rigorous analysis of the convergence of the distributed algorithm.

3) We also study the convergence of the algorithm in the presense of measurement noise and provide guidelines on how to choose the step-sizes to reduce the disturbance in the resource allocation due to noise.

4) Our studies reveal how the inherent multi-path nature of the problem can potentially lead to such difficulties as instability and oscillation, and how these difficulties should be addressed by the selection of the parameters in the distributed algorithm.

We believe that these results and insights are important for network designers who face these types of resource allocation problems. In the rest of the paper, we will first present a number of examples that provide motivation for the multi-path utility maximization problem, and discuss related works in Section II. We will then present our distributed solution in Section III. The convergence of the distributed algorithm will be studied in Sections IV when there is no measurement noise in the system, and in Section V when there is measurement noise. Simulation results are presented in Section VI.

II. Applications and Related Work

We now give a few examples of the networking contexts where optimization problems of the form (1) appear.

Example 1: A canonical example is the *multi-path flow control problem* in a wireline network (sometimes referred to as the multi-path routing and congestion control problem) [2], [3], [4], [5]. The network has L links and I users. The capacity of each link l is R^l. Each user i has $J(i)$ alternative paths through the network. Let $H^l_{ij} = 1$ if the path j of user i uses link l, $H^l_{ij} = 0$, otherwise. Let s_{ij} be the rate at which user i sends data on path j. The total rate sent by user i is then $\sum_{j=1}^{J(i)} s_{ij}$. Let $U_i(\sum_{j=1}^{J(i)} s_{ij})$ be the utility received by the user i at rate $\sum_{j=1}^{J(i)} s_{ij}$. The utility function $U_i(\cdot)$ characterizes how satisfied user i is when it can transmit at a certain data rate. We assume that $U_i(\cdot)$ is a concave function, which reflects the "law of diminishing returns." The flow control problem can be formulated as the following optimization problem [2], [3], [4], [5], which is essentially in the same form as (1):

$$\begin{aligned} \max_{[s_{ij}]\geq 0} \quad & \sum_{i=1}^{I} U_i(\sum_{j=1}^{J(i)} s_{ij}) \\ \text{subject to} \quad & \sum_{i=1}^{I}\sum_{j=1}^{J(i)} H^l_{ij} s_{ij} \leq R^l \quad \text{for all } l = 1, ..., L. \end{aligned}$$

Example 2: We next consider the *optimal routing problem*, which deals with a dynamic group of users instead of a static set of users as in Example 1. The network has L links. The capacity of each link l is R^l. There are I classes of users. Users of class i arrive to the network according to a Poisson process with rate λ_i. Each user of class i can be routed to one of the $J(i)$ alternative paths through the network. Let $H^l_{ij} = 1$ if path j of class i uses link l, $H^l_{ij} = 0$, otherwise. Each user of class i, if admitted, will hold r_i amount of resource on each of the links along the path that it is routed to, and generate v_i amount of revenue per unit time. The service time distribution for users of class i is *general* with mean $1/\mu_i$. The service times are *i.i.d.* and independent of the arrivals. The objective of the network is, by making admission and routing decisions for each incoming user, to maximize the revenue collected from the users that are admitted into the network.

The above optimal routing problem is important for networks that attempt to support users with rigid Quality of Service (QoS) requirements. In fact, maximizing the global revenue can be translated into minimizing the blocking probability, which has been used as the main performance measure in numerous past comparison studies of QoS routing proposals [6], [7], [8]. Unfortunately, the global optimal routing policy is usually difficult to solve. When the service time distribution is exponential, we can formulate the problem as a Dynamic Programming problem. However, the complexity of the solution grows exponentially with the number of classes. When the service time distribution is general, there are few tools available to obtain the optimal routing policy.

Despite the difficulty in solving the optimal routing policy, there exists a *simple* solution to the optimal routing problem that is *asymptotically optimal when the capacity of the system is large*. It can be shown that the optimal long term average revenue that the network can achieve is bounded by the following upper bound [9]:

$$\max_{\substack{p_{ij} \geq 0,\ \sum_{j=1}^{J(i)} p_{ij} \leq 1 \\ i=1,\dots,I}} \sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} \sum_{j=1}^{J(i)} p_{ij} v_i \tag{3}$$

$$\text{subject to } \sum_{i=1}^{I} \sum_{j=1}^{J(i)} \frac{\lambda_i}{\mu_i} r_i p_{ij} H_{ij}^l \leq R^l \quad \text{for all } l.$$

The solution to the upper bound, $\vec{p} = [p_{ij}]$, induces the following simple *proportional routing scheme*. In this scheme, each user of class i will be admitted with probability $\sum_{j=1}^{J(i)} p_{ij}$, and once admitted, will be routed to path j with probability $p_{ij} / \sum_{j=1}^{J(i)} p_{ij}$. One can show that the above scheme asymptotically achieves the upper bound when the capacity of the network is large [9]. *This result holds even if the service time distribution is general.* Hence, solving (3) gives us a simple and asymptotically optimal routing policy. Note that (3) is a special case of (1) where the utility function $f_i(\cdot)$ is linear. It is also possible to generalize the result to the case of strictly concave utility functions [10].

Readers can refer to [1] for another application of the multi-path utility maximization problem where the network uses pricing to control the users' behavior.

A. Related Work

The *single-path* utility maximization problem, i.e., when each user (or class) has only one path, has been extensively studied in the past, mainly in the context of Internet flow control (see, for example, [2], [11], [12], [13], [14] and the reference therein). However, the *multi-path* utility maximization problem has received less attention in the literature [2], [3], [4], [5]. In [2], after studying the single-path utility maximization problem, the authors briefly discuss the extension to the multi-path case. They categorize the solutions into primal algorithms and dual algorithms. Global convergence of the primal algorithms is studied in [2] for the case when feedback delays are negligible. (On the other hand, the dual algorithms there have an oscillation problem as we will discuss soon). Local stability of primal algorithms with feedback delays is further studied in [5]. Since [2] and [5] use a penalty-function approach, in general the algorithms there can only produce *approximate* solutions to the original problem (1).

The method in [3] can be viewed as an extension of the primal algorithms in [2] for computing *exact* solutions to problem (1). It employs a (discontinuous) binary feedback mechanism from the network components to the users: each network component will send a feedback signal of 1 when the total amount of resources consumed at the network component is greater than its capacity, and it sends a feedback signal of 0 otherwise. The authors of [3] show that, if each network component can measure the total amount of consumed resources *precisely*, their algorithm will converge to the *exact* optimal solution of problem (1). However, their algorithm will not work properly *in the presence of measurement noise*: if the network component can only estimate the amount of consumed resources with some random error (we will see in Section III-A how such situations arise), it could send a feedback signal of 1 erroneously even if the true amount of resources consumed is less than its capacity (or, a feedback signal of 0 even if the true amount of resources consumed exceeds its capacity). Therefore, the algorithm in [3] cannot produce the exact optimal solution when there is measurement noise. The *AVQ* algorithm [12], [15] is also worth mentioning as an extension of the primal algorithms in [2] for computing *exact* solutions. However, the literature on the *AVQ* algorithm has focused on the single-path case. The extension to the multi-path case has not been rigorously studied.

Dual algorithms that can produce *exact* solutions are developed in [11] for the single-path case. When extended to the multi-path case, both this algorithm and the dual algorithm in [2] share the same *oscillation problem*. That is, although the dual variables in their algorithms may converge, the more meaningful primal variables (i.e., the resource allocation x_{ij}) will not converge. (We will illustrate this problem further in Section III.) This difficulty arises mainly because the objective function of problem (1) is not *strictly concave* in the primal variables x_{ij} once some users have multiple paths. The authors in [4] attempt to address the oscillation problem by adding a quadratic term onto the objective function. Their approach bears some similarities to the idea that we use in this paper. However, they do not provide rigorous proofs of convergence for their algorithms.

Another method that is standard in convex programming for dealing with the lack of strict concavity is the Alternate Direction Method of Multipliers (ADMM) [16, p249, P253]. It has known convergence property (when there is no measurement noise) and can also be implemented in a distributed fashion. However, when implemented in a network setting, the ADMM algorithm requires substantial communication overhead. At each iteration, the ADMM algorithm requires that each network component divides the amount of unallocated capacity equally among all users sharing the network component and communicates the share back to each user. Each user not only needs to know the cost of each path (as in the distributed algorithm we will propose in this paper), but also needs to know its share of unallocated capacity at each network component. Further, in a practical network scenario where the set of active users in the system keep changing, unless the network has a reliable signaling mechanism, even keeping track of the current number of active users in the system requires maintaining per-user information. It is also unclear how the ADMM algorithm would behave in the presence of measurement noise. In this paper, we will study new solutions that are specifically designed for online implementation, and that do not require each network component to store and maintain per-user information.

III. The Distributed Algorithm

As we have pointed out earlier, one of the main difficulties in solving (1) is that, once some users have multiple alternative paths, the objective function of (1) is not strictly concave. As we go into the details of the analysis, we will see the manifestation of this difficulty in different aspects. At a high level, since the primal problem is not strictly concave, the dual problem may not be differentiable at every point. In this paper, we would still like to use a duality based approach, because the dual problem usually has simpler constraints and is easily decomposable. To circumvent the difficulty due to the lack of strict concavity, we use ideas from Proximal Optimization Algorithms [16, p232]. The idea is to add a quadratic term to the objective function. Let $\vec{x}_i = [x_{ij}, j = 1, ..., J(i)]$ and

$$C_i = \{\vec{x}_i | x_{ij} \geq 0 \text{ for all } j \text{ and } \sum_{j=1}^{J(i)} x_{ij} \in [m_i, M_i]\}, \quad i = 1, ..., I. \tag{4}$$

Let $\vec{x} = [\vec{x}_1, ..., \vec{x}_I]^T$ and let C denote the Cartesian product of C_i, i.e., $C = \bigotimes_{i=1}^{I} C_i$. We now introduce an auxiliary variable y_{ij} for each x_{ij}. Let $\vec{y}_i = [y_{ij}, j = 1, ..., J(i)]$ and $\vec{y} = [\vec{y}_1, .., \vec{y}_I]^T$. The optimization then becomes:

$$\begin{aligned} \max_{\vec{x}\in C, \vec{y}\in C} \quad & \sum_{i=1}^{I} f_i(\sum_{j=1}^{J(i)} x_{ij}) - \sum_{i=1}^{I}\sum_{j=1}^{J(i)} \frac{c_i}{2}(x_{ij} - y_{ij})^2 \\ \text{subject to} \quad & \sum_{i=1}^{I}\sum_{j=1}^{J(i)} E_{ij}^l x_{ij} \leq R^l \quad \text{for all } l, \end{aligned} \tag{5}$$

where c_i is a positive number chosen for each i. It is easy to show that the optimal value of (5) coincides with that of (1). In fact, Let $\vec{x^*}$ denote the maximizer of (1), then $\vec{x} = \vec{x^*}, \vec{y} = \vec{x^*}$ is the maximizer of (5). Note that although a maximizer of (1) always exists, it is usually not unique since the objective function is not strictly concave.

The standard Proximal Optimization Algorithm then proceeds as follows:

Algorithm $\mathcal{P}$:

At the t-th iteration,

P1) Fix $\vec{y} = \vec{y}(t)$ and maximize the augmented objective function with respect to $\vec{x}$. To be precise, this step solves:

$$\begin{aligned} \max_{\vec{x}\in C} \quad & \sum_{i=1}^{I} f_i(\sum_{j=1}^{J(i)} x_{ij}) - \sum_{i=1}^{I}\sum_{j=1}^{J(i)} \frac{c_i}{2}(x_{ij} - y_{ij})^2 \\ \text{subject to} \quad & \sum_{i=1}^{I}\sum_{j=1}^{J(i)} E_{ij}^l x_{ij} \leq R^l \quad \text{for all } l. \end{aligned} \tag{6}$$

Note that the maximization is taken over $\vec{x}$ only. With the addition of the quadratic term $\sum_{j=1}^{J(i)} \frac{c_i}{2}(x_{ij} - y_{ij})^2$, for any fixed $\vec{y}$, the primal objective function is now strictly concave with respect to $\vec{x}$. Hence, the maximizer of (6) exists and is unique. Let $\vec{x}(t)$ be the solution to this optimization.

P2) Set $\vec{y}(t+1) = \vec{x}(t)$.

It is easy to show that such iterations will converge to the optimal solution of problem (1) as $t \to \infty$ [16, p233].

Step P1 still needs to solve a global non-linear programming problem at each iteration. Since the objective function in (6) is now strictly concave, we can use standard duality techniques. Let $q^l, l = 1, ..., L$ be the Lagrange Multipliers for the constraints in (6). Let $\vec{q} = [q^1, ..., q^L]^T$. Define the Lagrangian as:

$$\begin{aligned} & L(\vec{x}, \vec{q}, \vec{y}) \\ = & \sum_{i=1}^{I} f_i(\sum_{j=1}^{J(i)} x_{ij}) - \sum_{l=1}^{L} q^l (\sum_{i=1}^{I}\sum_{j=1}^{J(i)} E_{ij}^l x_{ij} - R^l) \\ & - \sum_{i=1}^{I}\sum_{j=1}^{J(i)} \frac{c_i}{2}(x_{ij} - y_{ij})^2 \\ = & \sum_{i=1}^{I} \left\{ f_i(\sum_{j=1}^{J(i)} x_{ij}) - \sum_{j=1}^{J(i)} x_{ij} \sum_{l=1}^{L} E_{ij}^l q^l \right. \\ & \left. - \sum_{j=1}^{J(i)} \frac{c_i}{2}(x_{ij} - y_{ij})^2 \right\} + \sum_{l=1}^{L} q^l R^l. \end{aligned} \tag{7}$$

Let $q_{ij} = \sum_{l=1}^{L} E_{ij}^l q^l$, $\vec{q}_i = [q_{ij}, j = 1, ..., J(i)]$. The objective function of the dual problem is then:

$$D(\vec{q}, \vec{y}) = \max_{\vec{x}\in C} L(\vec{x}, \vec{q}, \vec{y}) = \sum_{i=1}^{I} B_i(\vec{q}_i, \vec{y}_i) + \sum_{l=1}^{L} q^l R^l, \tag{8}$$

where

$$\begin{aligned} B_i(\vec{q}_i, \vec{y}_i) = & \max_{\vec{x}_i \in C_i} \left\{ f_i(\sum_{j=1}^{J(i)} x_{ij}) - \sum_{j=1}^{J(i)} x_{ij} q_{ij} \right. \\ & \left. - \sum_{j=1}^{J(i)} \frac{c_i}{2}(x_{ij} - y_{ij})^2 \right\}. \end{aligned} \tag{9}$$

The dual problem of (6), given $\vec{y}$, then corresponds to minimizing D over the dual variables $\vec{q}$, i.e.,

$$\min_{\vec{q}\geq 0} D(\vec{q}, \vec{y}).$$

Since the objective function of the primal problem (6) is strictly concave, the dual is always differentiable. Let $\vec{q} = \vec{q}(t')$. The gradient of D is

$$\frac{\partial D}{\partial q^l} = R^l - \sum_{i=1}^{I}\sum_{j=1}^{J(i)} E_{ij}^l x_{ij}^0(t'),$$

where $x_{ij}^0(t')$ solves (9) for $\vec{q} = \vec{q}(t')$. The step P1 can then be solved by gradient descent iterations on the dual variables

$\vec{q}$, i.e.,

$$q^l(t'+1) = \left[q^l(t') + \alpha^l (\sum_{i=1}^{I} \sum_{j=1}^{J(i)} E_{ij}^l x_{ij}^0(t') - R^l) \right]^+ , \quad (10)$$

where $[\cdot]^+$ denotes the projection to $[0, +\infty)$. It is again easy to show that, given $\vec{y}$, the dual update (10) will converge to the minimizer of $D(\vec{q}, \vec{y})$ as $t' \to \infty$, provided that the step-sizes α^l are sufficiently small [16, p214].

Remark (The Oscillation Problem Addressed): From (9) we can observe the potential *oscillation problem* caused by the multi-path nature of problem (1), and the crucial role played by the additional quadratic term in dampening this oscillation. Assume that there is no additional quadratic term, i.e., $c_i = 0$. Readers can verify that, when (9) is solved for any user i that has multiple alternative paths, only paths that have the least q_{ij} will have positive x_{ij}. That is, $q_{ij} > \min_k q_{ik} \Rightarrow x_{ij} = 0$. This property can easily lead to oscillation of x_{ij} when the dual variables $\vec{q}$ are being updated. To see this, assume that a user i has two alternative paths, and the sum of the dual variables on these two paths, $q_{i,1} = \sum_{l=1}^{L} E_{i,1}^l q^l$ and $q_{i,2} = \sum_{l=1}^{L} E_{i,2}^l q^l$, are close to each other. At one time instant $q_{i,1}$ could be greater than $q_{i,2}$, in which case the maximum point of (9) satisfies $x_{i,1} = 0$ and $x_{i,2} > 0$. At the next time instant, since more resources are consumed on network components on path 2, the dual variables $\vec{q}$ could be updated such that $q_{i,2} > q_{i,1}$ (see the update equation (10)). In this case, the maximum point of (9) will require that $x_{i,1} > 0$ and $x_{i,2} = 0$, i.e., the resource allocation will move *entirely* from path 2 over to path 1. This kind of flip-floping can continue forever and is detrimental to network control. When $c_i > 0$, however, the maximum point $\vec{x}_i$ of (9) is continuous in $\vec{q}_i$ (shown later in Lemma 1). Hence, the quadratic term serves a crucial role to dampen the oscillation and stablize the system.

A. Towards Constructing Online Solutions

The algorithm $\mathcal{P}$ that we have constructed requires the *two-level convergence* structure typical in proximal optimization algorithms. The algorithm $\mathcal{P}$ consists of an outer level of iterations, i.e., iterations P1 and P2, where each outer iteration P1 consists of an inner level of iterations (10). For the convergence of algorithm $\mathcal{P}$ to hold, the inner level of iterations (10) must converge before each outer iteration can proceed from step P1 to P2. Such a two-level convergence structure is unsuitable for online implementation because in practice, it is difficult for the network elements to decide in a distributive fashion when the inner level of iterations should stop.

Despite this difficulty, the main building blocks (9) and (10) of algorithm $\mathcal{P}$ have several attractive attributes desirable for online implementation. In particular, all computation can be carried out based on local information, and hence can be easily distributed. More precisely, in the definition of the dual objective function $D(\vec{q}, \vec{y})$ in (8), we have decomposed the original problem into I separate subproblems for each user $i = 1, ..., I$. Given $\vec{q}$, each subproblem B_i (9) can now be solved independently. If we interpret q^l as the *implicit cost* per unit resource on network component l, then q_{ij} is the cost per unit resource on path j of user i. We can call q_{ij} the cost of path j of user i. Thus the costs $q_{ij}, j = 1, ..., J(i)$, capture all the information that each user i needs to know in order to determine its resource allocation x_{ij}. Given q_{ij}, an efficient algorithm can solve the subproblem (9) in at most $O[J(i) \log J(i)]$ steps. (Please refer to [17] for details of the algorithm.)

Further, according to (10), the implicit cost q^l can be updated at each network component l based on the difference between the capacity R^l and the aggregate load $\sum_{i=1}^{I} \sum_{j=1}^{J(i)} E_{ij}^l x_{ij}^0(t')$. In many applications, this aggregate load can be measured by each network component directly. For example, in the multi-path flow control problem (Example 1), the aggregate load $\sum_{i=1}^{I} \sum_{j=1}^{J(i)} H_{ij}^l s_{ij}$ is simply the aggregate data rate going through link l, which can be estimated by counting the total amount of data forwarded on the link over a certain time window. Hence, *no per-user information needs to be stored or maintained*. In some applications, there is yet another reason why the measurement-based approach is advantageous. That is, by measuring the aggregate load directly, the algorithm does not need to rely on prior knowledge of the parameters of the system, and hence can automatically adapt to the changes of these parameters. For example, in the optimal routing problem (Example 2), each link l needs to estimate the aggregate load $\sum_{i=1}^{I} \sum_{j=1}^{J(i)} \frac{\lambda_i}{\mu_i} r_i p_{ij} H_{ij}^l$. Since the probability that a user of class i is routed to path j is p_{ij}, the arrival process of users of class i on link l is a Poisson process with rate $\lambda_i p_{ij}$. Assume that neither the mean arrival rate λ_i nor the mean service time $1/\mu_i$ are known *a priori*, but each user knows its own service time in advance. Each link can then estimate the aggregate load as follows: over a certain time window W, each link l collects the information of the arriving users *from all classes* to link l. Let w be the total number of arrivals during W. Let $r_k, T_k, k = 1, ...w$ denote the bandwidth requirement and the service time, respectively, of the k-th arrival. (This information can be carried along with the connection setup message when the user arrives.) Let

$$\theta = \frac{\sum_{k=1}^{w} r_k T_k}{W}.$$

Then, it is easy to check that

$$\mathbf{E}[\theta] = \sum_{i=1}^{I} \sum_{j=1}^{J(i)} \frac{\lambda_i}{\mu_i} r_i p_{ij} H_{ij}^l,$$

i.e., θ is an unbiased estimate of the aggregate load. Note that no prior knowledge on the demand parameters λ_i and μ_i is needed in the estimator. Hence, the algorithm can automatically track the changes in the arrival rates and service times of the users [10].

B. The New Algorithm

In the rest of the paper, we will study the following algorithm that generalizes algorithm $\mathcal{P}$.

Algorithm $\mathcal{A}$:

Fix $K \geq 1$. At the t-th iteration:

A1) Fix $\vec{y} = \vec{y}(t)$ and use gradient descent iteration (10) on the dual variable $\vec{q}$ for K times. To be precise, let $\vec{q}(t,0) = \vec{q}(t)$. Repeat for each $k = 0, 1, ...K-1$:
Let $\vec{x}(t,k)$ be the primal variable that solves (9) given the dual variable $\vec{q}(t,k)$, i.e., $\vec{x}(t,k) = \underset{\vec{x}\in C}{\operatorname{argmax}} L(\vec{x}, \vec{q}(t,k), \vec{y}(t))$. Update the dual variables by

$$q^l(t,k+1) = \Big[q^l(t,k) + \alpha^l\Big(\sum_{i=1}^{I}\sum_{j=1}^{J(i)} E^l_{ij} x_{ij}(t,k) - R^l\Big)\Big]^+, \text{ for all } l. \tag{11}$$

A2) Let $\vec{q}(t+1) = \vec{q}(t,K)$. Let $\vec{z}(t)$ be the primal variable that solves (9) given the new dual variable $\vec{q}(t+1)$, i.e., $\vec{z}(t) = \underset{\vec{x}\in C}{\operatorname{argmax}} L(\vec{x}, \vec{q}(t+1), \vec{y}(t))$. Set

$$y_{ij}(t+1) = y_{ij}(t) + \beta_i(z_{ij}(t) - y_{ij}(t)), \text{ for all } i, j, \tag{12}$$

where $0 < \beta_i \leq 1$ for each i.

As discussed in Section III-A, in certain applications the aggregate load $\sum_{i=1}^{I}\sum_{j=1}^{J(i)} E^l_{ij} x_{ij}(t,k)$ is estimated through online measurement with non-negligible noises. The update (11) should then be replaced by:

$$q^l(t,k+1) = \Big[q^l(t,k) + \alpha^l\Big(\sum_{i=1}^{I}\sum_{j=1}^{J(i)} E^l_{ij} x_{ij}(t,k) - R^l + n^l(t,k)\Big)\Big]^+, \tag{13}$$

where $n^l(t,k)$ represents the measurement noise at link l.

From now on, we will refer to (11) or (13) as the *dual update*, and (12) as the *primal update*. A *stationary point of the algorithm* $\mathcal{A}$ is defined to be a primal-dual pair $(\vec{y^*}, \vec{q^*})$ such that

$$\vec{y^*} = \underset{\vec{x}\in C}{\operatorname{argmax}} L(\vec{x}, \vec{q^*}, \vec{y^*}), \quad \sum_{i=1}^{I}\sum_{j=1}^{J(i)} E^l_{ij} y^*_{ij} \leq R^l \text{ for all } l,$$

$$q^{l,*} \geq 0, \text{ and } q^{l,*}\Big(\sum_{i=1}^{I}\sum_{j=1}^{J(i)} E^l_{ij} y^*_{ij} - R^l\Big) = 0 \text{ for all } l. \tag{14}$$

These are precisely the *complementary slackness* conditions for the problem (1). By standard duality theory, for any stationary point $(\vec{y^*}, \vec{q^*})$ of the algorithm $\mathcal{A}$, $\vec{x} = \vec{y^*}$ solves the problem (1).

The main components of algorithm $\mathcal{A}$ (i.e., the primal and dual updates) are essentially the same as that of the standard proximal optimization algorithm $\mathcal{P}$. Therefore, our new algorithm $\mathcal{A}$ inherits from algorithm $\mathcal{P}$ those attributes desirable for online implementation. However, the main difference is that, in algorithm $\mathcal{A}$, only K number of dual updates are executed at each iteration of step A1. If $K = \infty$, then algorithm $\mathcal{A}$ and algorithm $\mathcal{P}$ will be equivalent, i.e., at each iteration of step A1 the optimization (6) is solved exactly. As we discussed earlier, such a two-level convergence structure is inappropriate for online implementation because it would be impractical to carry out an algorithm in phases where each phase consists of an *infinite* number of dual updates. Further, because each phase only serves to solve the augmented problem (6), such a two-level convergence structure is also likely to slow the convergence of the entire algorithm as too many dual updates are wasted at each phase.

On the other hand, when $K < \infty$, at best an approximate solution to (6) is obtained at each iteration of step A1. If the accuracy of the approximate solution can be controlled appropriately (see [18]), one can still show convergence of algorithm $\mathcal{A}$. However, in this case the number of dual updates K in step A1 has to depend on the required accuracy and usually needs to be large. Further, for online implementation, it is also difficult to control the accuracy of the approximate solution to (6) in a distributed fashion.

In this work, we take a different approach. We do not require a two-level convergence structure and we allow an arbitrary choice of $K \geq 1$. Hence, our approach does not impose any requirement on the accuracy of the approximate solution to (6). As we just discussed, relaxing the algorithm in such a way is a crucial step in making the algorithm amenable to online distributed implementation. Somewhat surprisingly, we will show in the next section that algorithm $\mathcal{A}$ will converge for any $K \geq 1$[1].

IV. Convergence without Measurement Noise

In this section, we study the convergence of algorithm $\mathcal{A}$ when there is no measurement noise, i.e., when the dynamics of the system are described by (11) and (12). The convergence of algorithm $\mathcal{A}$ can be most easily understood by looking at its continuous-time approximation as follows:

Algorithm $\mathcal{AC}$:

A1-C) dual update:

$$\frac{d}{dt}q^l(t) = \begin{cases} \hat{\alpha}^l\Big(\sum_{i=1}^{I}\sum_{j=1}^{J(i)} E^l_{ij} x_{ij}(t) - R^l\Big) \\ \quad \text{if } q^l(t) > 0 \text{ or } \sum_{i=1}^{I}\sum_{j=1}^{J(i)} E^l_{ij} x_{ij}(t) \geq R^l, \\ 0 \quad \text{otherwise,} \end{cases} \tag{15}$$

where $\vec{x}(t) = \underset{\vec{x}\in C}{\operatorname{argmax}} L(\vec{x}, \vec{q}(t), \vec{y}(t))$.

A2-C) primal update:

$$\frac{d}{dt}y_{ij}(t) = \hat{\beta}_i(x_{ij}(t) - y_{ij}(t)). \tag{16}$$

[1]We note that the idea of carrying out only a small number of steps before terminating the inner iterations has appeared in other optimization methods. For example, the short-step path-following interior-point method for linear programming [19, p86] uses only one Newton step towards the central path between successive reductions of the barrier parameter. Under appropriate conditions, the short-step path-following method converges even though the algorithm does not produce points that are exactly on the central path. By a similar token, the Alternate Direction Method of Multipliers (ADMM) [16, p253] uses only one iteration in minimizing the Augmented Lagrangian before alternating the minimization variable, and is also shown to converge. Nonetheless, the structure of our algorithm $\mathcal{A}$ is quite different from these other methods in the literature, and hence requires a new convergence study.

We will first provide the convergence proof of this continuous-time algorithm, which is of independent interest. Further, it will also provide some important insights to the convergence of the original discrete-time algorithm. Note that $\hat{\alpha}^l\,dt$ and $\hat{\beta}_i\,dt$ would correspond to the step-sizes α^l and β_i in the discrete-time algorithm $\mathcal{A}$. Thus, the continuous-time algorithm $\mathcal{AC}$ can be view as the functional limit of the discrete-time algorithm by *driving the step-sizes α^l and β_i to zero* and by appropriately rescaling time (see [17]).

We begin with a supporting lemma (Lemma 1), which will also be used in later parts of the paper. For the sake of brevity, we will use the following vector notation for the rest of the paper. Let E denote the matrix with L rows and $\sum_{i=1}^{I} J(i)$ columns such that the $(l, \sum_{k=1}^{i-1} J(k) + j)$ element is E_{ij}^l. Let $R = [R^1, R^2, ...R^l]^T$. Then the constraint of problem (1) can be written as $E\vec{x} \le R$. Let V and $\hat{B}$ be $\sum_{i=1}^{I} J(i) \times \sum_{i=1}^{I} J(i)$ diagonal matrices, where the $(\sum_{k=1}^{i-1} J(k)+1)$-th to $(\sum_{k=1}^{i} J(k))$-th diagonal elements are c_i and $\hat{\beta}_i$, respectively (i.e., each c_i or $\hat{\beta}_i$ is repeated $J(i)$ times). Let $\hat{A}$ be the $L \times L$ diagonal matrix whose l-th diagonal element is $\hat{\alpha}^l$. It will also be convenient to view the objective function in (1) as a concave function of $\vec{x}$, i.e, $\mathbf{f}(\vec{x}) = \sum_{i=1}^{I} f_i(\sum_{j=1}^{J(i)} x_{ij})$. Further, we can incorporate the constraint $\vec{x} \in C$ into the definition of the function $\mathbf{f}$ by setting $\mathbf{f}(\vec{x}) = -\infty$ if $\vec{x} \notin C$. Then the function $\mathbf{f}$ is still concave, and the problem (1) can be simply rephrased as maximizing $\mathbf{f}(\vec{x})$ subject to $E\vec{x} \le R$. The Lagrangian (7) also becomes:

$$\begin{aligned} & L(\vec{x}, \vec{q}, \vec{y}) \\ = \;& \mathbf{f}(\vec{x}) - \vec{x}^T E^T \vec{q} - \frac{1}{2}(\vec{x}-\vec{y})^T V (\vec{x}-\vec{y}) + \vec{q}^T R. \end{aligned} \tag{17}$$

The continuous time algorithm $\mathcal{AC}$ can then be viewed as the *projected forward iteration* for solving the zeros of the following *monotone mapping* [20]:

$$\mathcal{T}: \quad [\vec{y}, \vec{q}] \to [\vec{u}, \vec{v}], \tag{18}$$

with

$$\vec{u}(\vec{y},\vec{q}) = -V(\vec{x^0}(\vec{y},\vec{q}) - \vec{y}),\ \vec{v}(\vec{y},\vec{q}) = -(E\vec{x^0}(\vec{y},\vec{q}) - R),$$

where $\vec{x^0}(\vec{y},\vec{q}) = \operatorname{argmax}_{\vec{x}} L(\vec{x},\vec{q},\vec{y})$. Define the inner product $\langle\, [\vec{y},\vec{q}], [\vec{u},\vec{v}]\,\rangle = \vec{y}^T\vec{u} + \vec{q}^T\vec{v}$, and the following norms:

$$||\vec{q}||_{\hat{A}} = \vec{q}^T \hat{A}^{-1}\vec{q},\ ||\vec{y}||_V = \vec{y}^T V \vec{y},\ ||\vec{y}||_{\hat{B}V} = \vec{y}^T \hat{B}^{-1} V \vec{y}. \tag{19}$$

Part 2 of the following Lemma shows that the mapping $\mathcal{T}$ is *monotone* [20]. Note that a mapping $\mathcal{T}$ is *monotone* if

$$\langle\, X_1 - X_2, \mathcal{T}X_1 - \mathcal{T}X_2 \,\rangle \ge 0 \text{ for any } X_1 \text{ and } X_2. \tag{20}$$

Lemma 1: Fix $\vec{y} = \vec{y}(t)$. Let $\vec{q}_1, \vec{q}_2$ be two implicit cost vectors, and let $\vec{x}_1$, $\vec{x}_2$ be the corresponding maximizers of the Lagrangian (17), i.e., $\vec{x}_1 = \operatorname{argmax}_{\vec{x}} L(\vec{x},\vec{q}_1,\vec{y}(t))$ and $\vec{x}_2 = \operatorname{argmax}_{\vec{x}} L(\vec{x},\vec{q}_2,\vec{y}(t))$. Then,

1) $(\vec{x}_2-\vec{x}_1)^T V(\vec{x}_2-\vec{x}_1) \le (\vec{q}_2-\vec{q}_1)^T E V^{-1} E^T (\vec{q}_2-\vec{q}_1)$, and
2) $\langle\, [\vec{y}_1-\vec{y}_2, \vec{q}_1-\vec{q}_2], \mathcal{T}[\vec{y}_1,\vec{q}_1] - \mathcal{T}[\vec{y}_2,\vec{q}_2]\,\rangle \ge 0$ for any $(\vec{y}_1,\vec{q}_1)$ and $(\vec{y}_2,\vec{q}_2)$.

Remark: Part 1 of Lemma 1 also shows that, given $\vec{y}$, the mapping from $\vec{q}$ to $\vec{x}$ is continuous.

Proof: We start with some additional notation. For $\vec{x^0} = \operatorname{argmax}_{\vec{x}} L(\vec{x},\vec{q},\vec{y})$, by taking subgradients (see [18]) of the Lagrangian (17) with respect to $\vec{x}$, we can conclude that there must exist a subgradient $\nabla\mathbf{f}(\vec{x^0})$ of $\mathbf{f}$ at $\vec{x^0}$ such that

$$\nabla\mathbf{f}(\vec{x^0}) - E^T\vec{q} - V(\vec{x^0} - \vec{y}) = 0. \tag{21}$$

Similarly, let $(\vec{y^*}, \vec{q^*})$ denote a stationary point of algorithm $\mathcal{A}$. Then $\vec{y^*} = \operatorname{argmax}_{\vec{x}} L(\vec{x},\vec{q^*},\vec{y^*})$, and we can define $\nabla\mathbf{f}(\vec{y^*})$ as the subgradient of $\mathbf{f}$ at $\vec{y^*}$ such that

$$\nabla\mathbf{f}(\vec{y^*}) - E^T\vec{q^*} = 0. \tag{22}$$

Applying (21) for $\vec{q}_1$ and $\vec{q}_2$, and taking difference, we have,

$$E^T(\vec{q}_2 - \vec{q}_1) \;=\; [\nabla\mathbf{f}(\vec{x}_2) - \nabla\mathbf{f}(\vec{x}_1)] - V(\vec{x}_2 - \vec{x}_1).$$

The concavity of $\mathbf{f}$ dictates that, for any $\vec{x}_1$, $\vec{x}_2$ and $\nabla\mathbf{f}(\vec{x}_1)$, $\nabla\mathbf{f}(\vec{x}_2)$,

$$[\nabla\mathbf{f}(\vec{x}_2) - \nabla\mathbf{f}(\vec{x}_1)]^T(\vec{x}_2-\vec{x}_1) \le 0. \tag{23}$$

Hence,

$$\begin{aligned} & (\vec{q}_2-\vec{q}_1)^T E V^{-1} E^T (\vec{q}_2-\vec{q}_1) \\ = \;& [\nabla\mathbf{f}(\vec{x}_2) - \nabla\mathbf{f}(\vec{x}_1)]^T V^{-1} [\nabla\mathbf{f}(\vec{x}_2) - \nabla\mathbf{f}(\vec{x}_1)] \\ & -2[\nabla\mathbf{f}(\vec{x}_2) - \nabla\mathbf{f}(\vec{x}_1)]^T (\vec{x}_2-\vec{x}_1) \\ & +(\vec{x}_2-\vec{x}_1)^T V (\vec{x}_2-\vec{x}_1) \\ \ge \;& (\vec{x}_2-\vec{x}_1)^T V (\vec{x}_2-\vec{x}_1). \end{aligned}$$

Part 2 of the Lemma can be shown analogously. ■

We can now prove the following result.

Proposition 2: The continuous-time algorithm $\mathcal{AC}$ will converge to a stationary point $(\vec{y^*}, \vec{q^*})$ of the algorithm $\mathcal{A}$ for any choice of $\hat{\alpha}^l > 0$ and $\hat{\beta}_i > 0$.

Proof: We can prove Proposition 2 using the following Lyapunov function. Let

$$\mathcal{V}(\vec{y},\vec{q}) = ||\vec{q} - \vec{q^*}||_{\hat{A}} + ||\vec{y} - \vec{y^*}||_{\hat{B}V}, \tag{24}$$

where the norms are defined in (19). It is easy to show that (see [17]),

$$\begin{aligned} & \frac{d}{dt}\mathcal{V}(\vec{y}(t),\vec{q}(t)) \\ \le \;& -2\langle\, [\vec{y}(t)-\vec{y^*}, \vec{q}(t)-\vec{q^*}], \mathcal{T}[\vec{y}(t),\vec{q}(t)] - \mathcal{T}[\vec{y^*},\vec{q^*}]\,\rangle. \end{aligned}$$

Hence, by Lemma 1, $\frac{d}{dt}\mathcal{V}(\vec{y}(t),\vec{q}(t)) \le 0$. Therefore, $\mathcal{V}(\vec{y}(t),\vec{q}(t))$ must converge to a limit V_0 as $t \to \infty$. We then use LaSalle's Invariance Principle [21, Theorem 3.4, p115] to show that the limit cycle of $(\vec{y}(t),\vec{q}(t))$ must contain a limit point $(\vec{y}_0,\vec{q}_0)$ that is also a stationary point of algorithm $\mathcal{AC}$ (see [17]). Replace $(\vec{y^*},\vec{q^*})$ in (24) by $(\vec{y}_0,\vec{q}_0)$, we thus have

$$\lim_{t\to\infty} ||\vec{q}(t)-\vec{q}_0||_{\hat{A}} + ||\vec{y}(t)-\vec{y}_0||_{\hat{B}V} = 0, \text{ as } t \to \infty.$$

■

We next study the convergence of the discrete-time algorithm $\mathcal{A}$. Since the continuous-time algorithm $\mathcal{AC}$ can be viewed as an approximation of the discrete-time algorithm $\mathcal{A}$ when the step-sizes are close to zero, we can then expect from Proposition 2 that algorithm $\mathcal{A}$ will converge when the step-sizes α^l and β_i are small. However, when these step-sizes are too small, convergence is unnecessarily slow. Hence, in practice, we would like to choose larger step-sizes, while still preserving the convergence of the algorithm. Such knowledge on the step-size rule can only be obtained by studying the convergence of the discrete-time algorithm directly.

Typically, convergence of the discrete-time algorithms requires stronger conditions on the associated mapping $\mathcal{T}$ defined in (18), i.e., the mapping $\mathcal{T}$ needs to be *strictly monotone* [20]. A mapping $\mathcal{T}$ is *strictly monotone* if and only if

$$\langle\, X_1 - X_2, \mathcal{T}X_1 - \mathcal{T}X_2 \,\rangle \geq d||\mathcal{T}X_1 - \mathcal{T}X_2|| \quad (25)$$

for any vectors X_1 and X_2, where d is a positive constant and $||\cdot||$ is an appropriately chosen norm. Note that strict monotonicity in (25) is stronger than *monotonicity* in (20). Such type of strict monotonicity indeed holds for the case when $K = \infty$, which is why the convergence is much easier to establish under the two-level convergence structure. However, when $K < \infty$, strict monotonicity will not hold for the mapping $\mathcal{T}$ defined in (18) *whenever some users in the network have multiple paths*. To see this, choose $X_2 = [\vec{y^*}, \vec{q^*}]$ to be a stationary point of algorithm $\mathcal{A}$ and assume that $E\vec{y^*} = R$. Let $X_1 = [\vec{y}, \vec{q^*}]$ such that $\sum_{j=1}^{J(i)} y_{ij} = \sum_{j=1}^{J(i)} y^*_{ij}$ for all i. Note that for a user i that has multiple paths, we can still choose $y_{ij} \neq y^*_{ij}$ for some j such that $E\vec{y} \neq R$. By comparing with the complementary slackness conditions (14), we have $\vec{x^0}(\vec{y}, \vec{q^*}) \triangleq \text{argmax}_{\vec{x}}\, L(\vec{x}, \vec{q^*}, \vec{y}) = \vec{y}$. Hence,

$$\vec{u}(\vec{y}, \vec{q^*}) = 0, \text{ and } \vec{v}(\vec{y}, \vec{q^*}) = -(E\vec{y} - R).$$

Further, since $\vec{u}(\vec{y^*}, \vec{q^*}) = 0$ and $\vec{v}(\vec{y^*}, \vec{q^*}) = 0$ by the complementary slackness conditions (14), we have

$$\begin{aligned} & \langle\, X_1 - X_2, \mathcal{T}X_1 - \mathcal{T}X_2 \,\rangle \\ = \;& \langle\, [\vec{y} - \vec{y^*}, \vec{q^*} - \vec{q^*}], \mathcal{T}[\vec{y}, \vec{q^*}] - \mathcal{T}[\vec{y^*}, \vec{q^*}] \,\rangle \\ = \;& [\vec{y} - \vec{y^*}]^T(\vec{u}(\vec{y}, \vec{q^*}) - \vec{u}(\vec{y^*}, \vec{q^*})) \\ & \quad + \vec{0}^{\,T}(\vec{v}(\vec{y}, \vec{q^*}) - \vec{v}(\vec{y^*}, \vec{q^*})) = 0. \end{aligned}$$

However, $\mathcal{T}X_1 - \mathcal{T}X_2 = [0, -(E\vec{y} - R)] \neq 0$. Hence, the inequality (25) will never hold! As we have just seen, it is precisely the multi-path nature of the problem that leads to this lack of strict monotonicity. (One can indeed show that (25) would have held if all users had one single path and the utility functions $f_i(\cdot)$ were strictly concave.)

This lack of strict monotonicity when $K < \infty$ forces us to carry out a more refined convergence analysis than that in the standard convex programming literature. We will need the following key supporting result. Let $(\vec{y^*}, \vec{q^*})$ denote a stationary point of algorithm $\mathcal{A}$. Using (23), we have

$$\left[\nabla\mathbf{f}(\vec{x}_1) - \nabla\mathbf{f}(\vec{y^*})\right]^T(\vec{x}_1 - \vec{y^*}) \leq 0. \quad (26)$$

The following Lemma can be viewed as an extension of the above inequality. The proof is very technical and is given in the Appendix.

Lemma 3: Fix $\vec{y} = \vec{y}(t)$. Let $\vec{q}_1, \vec{q}_2$ be two implicit cost vectors, and let $\vec{x}_1$, $\vec{x}_2$ be the corresponding maximizers of the Lagrangian (17). Then,

$$\begin{aligned} & \left[\nabla\mathbf{f}(\vec{x}_1) - \nabla\mathbf{f}(\vec{y^*})\right]^T(\vec{x}_2 - \vec{y^*}) \\ \leq \;& \frac{1}{2}(\vec{q}_2 - \vec{q}_1)^T E V^{-1} E^T (\vec{q}_2 - \vec{q}_1), \end{aligned}$$

where $\nabla\mathbf{f}(\vec{x}_1)$ and $\nabla\mathbf{f}(\vec{y^*})$ are defined in (21) and (22), respectively.

Remark: If $\vec{q}_2 = \vec{q}_1$, then $\vec{x}_2 = \vec{x}_1$ and we get back to (26). Lemma 3 tells us that as long as $\vec{q}_1$ is not very different from $\vec{q}_2$, the cross-product on the left hand side will not be far above zero either.

We can then prove the following main result, which establishes the sufficient condition on the step-sizes for the convergence of the discrete-time algorithm $\mathcal{A}$.

Proposition 4: Fix $1 \leq K \leq \infty$. As long as the step-size α^l is small enough, algorithm $\mathcal{A}$ will converge to a stationary point $(\vec{y^*}, \vec{q^*})$ of the algorithm, and $\vec{x^*} = \vec{y^*}$ will solve the original problem (1). The sufficient condition for convergence is:

$$\max_l \alpha^l < \begin{cases} \frac{2}{\mathcal{SL}} \min_i c_i & \text{if } K = \infty \\ \frac{1}{2\mathcal{SL}} \min_i c_i & \text{if } K = 1 \\ \frac{4}{5K(K+1)\mathcal{SL}} \min_i c_i & \text{if } K > 1 \end{cases},$$

where $\mathcal{L} = \max\{\sum_{l=1}^{L} E^l_{ij}, i = 1, ..., I, j = 1, ...J(i)\}$, and $\mathcal{S} = \max\{\sum_{i=1}^{I}\sum_{j=1}^{J(i)} E^l_{ij}, l = 1, ..., L\}$.

Proposition 4 establishes the convergence of algorithm $\mathcal{A}$ for any value of K (even $K = 1$ is good enough). Hence, the typical two-level convergence structure is no longer required. Further, we observe that the sufficient condition for convergence when $K = 1$ differs from that of $K = \infty$ by only a factor of 4. Note that for $K = \infty$, the sufficient condition in fact ensures the convergence of the dual updates to the solution of the augmented problem (6) during *one iteration* of step A1. On the other hand, the sufficient condition for $K = 1$ ensures the convergence of the entire algorithm $\mathcal{A}$. By showing that the sufficient conditions for the two cases differ by only a factor of 4, we can infer that the convergence of the entire algorithm when $K = 1$ is not necessarily much slower than the convergence of one iteration of step A1 when $K = \infty$. Hence, the algorithm $\mathcal{A}$ with $K = 1$ in fact converges much faster. For $K > 1$, our result requires that the step-size be inversely proportional to K^2. This is probably not the tightest possible result: we conjecture that the same condition for $K = 1$ would work for any K. However, we leave this for future work. We also note that c_i appears on the right hand side of the sufficient conditions. Hence, by making the objective function more concave, we also relax the requirement on the step-sizes α^l. Finally, Proposition 4 indicates that convergence will hold for any β_i (the step-size in the primal update) that is in $(0, 1]$. In summary, the discrete-time analysis allows much wider choices of the step-sizes than those predicted by the continuous-time analysis.

Proof: [of Proposition 4] Due to space constraints, we will focus on the case when $K = 1$. The other cases can be shown analogously (see [17] for details). Define matrices A and B analogously to matrices $\hat{A}$ and $\hat{B}$, respectively, except that their diagonal elements are now filled with the step-sizes α^l and β_i of the discrete-time algorithm $\mathcal{A}$. Define the norms analogously to (19). When $K = 1$,

$$\vec{q}(t+1) = [\vec{q}(t) + A(E\vec{x}(t) - R)]^+. \quad (27)$$

Let $(\vec{y^*}, \vec{q^*})$ be any stationary point of algorithm $\mathcal{A}$. We will show that the Lyapunov function

$$\mathcal{V}(\vec{y}(t), \vec{q}(t)) = ||\vec{q}(t) - \vec{q^*}||_A + ||\vec{y}(t) - \vec{y^*}||_{BV}$$

is non-increasing in t. Using the property of the projection mapping [16, Proposition 3.2(b), p211], we have

$$(\vec{q}(t+1) - \vec{q^*})^T A^{-1}(\vec{q}(t+1) - [\vec{q}(t) + A(E\vec{x}(t) - R)]) \leq 0.$$

Hence,

$$\begin{aligned} & ||\vec{q}(t+1) - \vec{q^*}||_A \\ = & ||\vec{q}(t) - \vec{q^*}||_A - ||\vec{q}(t+1) - \vec{q}(t)||_A \\ & +2(\vec{q}(t+1) - \vec{q^*})^T A^{-1}(\vec{q}(t+1) - \vec{q}(t)) \\ \leq & ||\vec{q}(t) - \vec{q^*}||_A - ||\vec{q}(t+1) - \vec{q}(t)||_A \\ & +2(\vec{q}(t+1) - \vec{q^*})^T (E\vec{x}(t) - R) \\ \leq & ||\vec{q}(t) - \vec{q^*}||_A - ||\vec{q}(t+1) - \vec{q}(t)||_A \\ & +2(\vec{q}(t+1) - \vec{q^*})^T E(\vec{x}(t) - \vec{y^*}). \end{aligned} \quad (28)$$

where in the last step we have used the fact that $E\vec{y^*} - R \leq 0$ and $\vec{q^*}^T (E\vec{y^*} - R) = 0$. On the other hand, since $y_{ij}(t+1) = (1-\beta_i)y_{ij}(t) + \beta_i z_{ij}(t)$, we have

$$\begin{aligned} & (y_{ij}(t+1) - y^*_{ij})^2 \\ & \quad \leq (1-\beta_i)(y_{ij}(t) - y^*_{ij})^2 + \beta_i(z_{ij}(t) - y^*_{ij})^2, \\ & ||\vec{y}(t+1) - \vec{y^*}||_{BV} - ||\vec{y}(t) - \vec{y^*}||_{BV} \\ & \quad \leq ||\vec{z}(t) - \vec{y^*}||_V - ||\vec{y}(t) - \vec{y^*}||_V. \end{aligned} \quad (29)$$

Hence, combining (28) and (29), we have,

$$\begin{aligned} & ||\vec{q}(t+1) - \vec{q^*}||_A + ||\vec{y}(t+1) - \vec{y^*}||_{BV} \\ & \quad -(||\vec{q}(t) - \vec{q^*}||_A + ||\vec{y}(t) - \vec{y^*}||_{BV}) \\ \leq & -||\vec{q}(t+1) - \vec{q}(t)||_A \\ & \quad +2(\vec{q}(t+1) - \vec{q^*})^T E(\vec{x}(t) - \vec{y^*}) \\ & \quad +||\vec{z}(t) - \vec{y^*})||_V - ||\vec{y}(t) - \vec{y^*}||_V \\ \leq & -||\vec{q}(t+1) - \vec{q}(t)||_A \\ & \quad + \{||\vec{z}(t) - \vec{y^*})||_V - ||\vec{y}(t) - \vec{y^*}||_V \\ & \qquad -2(\vec{z}(t) - \vec{y}(t))^T V(\vec{x}(t) - \vec{y^*})\} \quad (30) \\ & \quad +2\left[\nabla \mathbf{f}(\vec{z}(t)) - \nabla \mathbf{f}(\vec{y^*})\right]^T (\vec{x}(t) - \vec{y^*}), \quad (31) \end{aligned}$$

where in the last step we have used (21) and (22), and consequently

$$E^T(\vec{q}(t+1) - \vec{q^*}) = \nabla \mathbf{f}(\vec{z}(t)) - \nabla \mathbf{f}(\vec{y^*}) - V(\vec{z}(t) - \vec{y}(t)).$$

By simple algebraic manipulation, we can show that the second term (30) is equal to

$$\begin{aligned} & ||\vec{z}(t) - \vec{y^*}||_V - ||\vec{y}(t) - \vec{y^*}||_V \\ & \quad -2(\vec{z}(t) - \vec{y}(t))^T V(\vec{x}(t) - \vec{y^*}) \\ = & ||(\vec{z}(t) - \vec{x}(t)||_V - ||\vec{y}(t) - \vec{x}(t)||_V. \end{aligned} \quad (32)$$

Invoking Lemma 1, part 1,

$$||\vec{z}(t) - \vec{x}(t)||_V \leq (\vec{q}(t+1) - \vec{q}(t))^T E V^{-1} E^T (\vec{q}(t+1) - \vec{q}(t)). \quad (33)$$

For the third term (31), we can invoke Lemma 3,

$$\begin{aligned} & 2\left[\nabla \mathbf{f}(\vec{z}(t)) - \nabla \mathbf{f}(\vec{y^*})\right]^T (\vec{x}(t) - \vec{y^*}) \\ \leq & (\vec{q}(t+1) - \vec{q}(t))^T E V^{-1} E^T (\vec{q}(t+1) - \vec{q}(t)). \end{aligned} \quad (34)$$

Therefore, by substituting (32-33) into (30), and substituting (34) into (31), we have

$$\begin{aligned} & \mathcal{V}(\vec{y}(t+1), \vec{q}(t+1)) - \mathcal{V}(\vec{y}(t), \vec{q}(t)) \\ \leq & -(\vec{q}(t+1) - \vec{q}(t))^T C_1 (\vec{q}(t+1) - \vec{q}(t)) \\ & \quad -||\vec{y}(t) - \vec{x}(t)||_V. \end{aligned}$$

where $C_1 = A^{-1} - 2EV^{-1}E^T$. If C_1 is positive definite, then $\mathcal{V}(\vec{y}(t), \vec{q}(t))$ is non-increasing in t and hence must have a limit, i.e.,

$$\lim_{t \to \infty} ||\vec{q}(t) - \vec{q^*}||_A + ||\vec{y}(t) - \vec{y^*}||_{BV} = V_0 \geq 0. \quad (35)$$

Therefore, the sequence $\{\vec{y}(t), \vec{q}(t), t = 1, ...\}$ is bounded, and there must exist a subsequence $\{\vec{y}(t_h), \vec{q}(t_h), h = 1, ...\}$ that converges to a limit point. Let $(\vec{y_0}, \vec{q_0})$ be this limit. By taking limits of (27) as $h \to \infty$, it is easy to show that $(\vec{y_0}, \vec{q_0})$ is also a stationary point of algorithm $\mathcal{A}$. Replace $(\vec{y^*}, \vec{q^*})$ by $(\vec{y_0}, \vec{q_0})$ in (35) and thus,

$$\begin{aligned} & \lim_{t \to \infty} ||\vec{q}(t) - \vec{q_0}||_A + ||\vec{y}(t) - \vec{y_0}||_{BV} \\ = & \lim_{h \to \infty} ||\vec{q}(t_h) - \vec{q_0}||_A + ||\vec{y}(t_h) - \vec{y_0}||_{BV} = 0. \end{aligned}$$

Hence $(\vec{y}(t), \vec{q}(t)) \to (\vec{y_0}, \vec{q_0})$ as $t \to \infty$. Finally, it is easy to show that a sufficient condition for C_1 to be positive definite is $\max_l \alpha^l < \frac{1}{2S\mathcal{L}} \min_i c_i$ (see [17]). ■

V. Convergence with Measurement Noise

In this section, we will study the convergence of algorithm $\mathcal{A}$ when there is measurement noise, i.e., when the dynamics of the system are governed by (12) and (13). The convergence of algorithm $\mathcal{A}$ will be established in the "stochastic approximation" sense, i.e., when the step-sizes are driven to zero in an appropriate fashion. To be specific, we replace the step-sizes α^l and β_i by

$$\alpha^l(t) = \eta_t \alpha^l_0, \quad \beta_i(t) = \eta_t \beta_{i,0},$$

for some positive sequence $\{\eta_t, t = 1, 2, ...\}$ that goes to zero as $t \to \infty$. For simplicity, we will focus on the case when $K = 1$ and we will drop the index k in (13). Let $N(t) = [n^l(t), l = 1, ..., L]^T$. Use the vector notation from the previous section and define matrices A_0 and B_0 analogously as the matrices A and B, respectively, except that the diagonal elements are now filled with α^l_0 and $\beta_{i,0}$. We can then rewrite algorithm $\mathcal{A}$ as:

Algorithm $\mathcal{AN}$:

A1-N) Let $\vec{x}(t) = \text{argmax}_{\vec{x}}\, L(\vec{x}, \vec{q}(t), \vec{y}(t))$. Update the dual variables by

$$\vec{q}(t+1) = [\vec{q}(t) + \eta_t A_0(E\vec{x}(t) - R + N(t))]^+.$$

A2-N) Let $\vec{z}(t) = \text{argmax}_{\vec{x}}\, L(\vec{x}, \vec{q}(t+1), \vec{y}(t))$. Update the primal variables by

$$\vec{y}(t+1) = \vec{y}(t) + \eta_t B_0(\vec{z}(t) - \vec{y}(t)).$$

Proposition 5: If

$$\sum_{t=1}^{\infty}\eta_t=\infty,\sum_{t=1}^{\infty}{\eta_t}^2<\infty,$$

$$\mathbf{E}[N(t)|\vec{x}(s),\vec{y}(s),\vec{q}(s),s\le t]=0,\ \text{and} \tag{36}$$

$$\sum_{t=1}^{\infty}{\eta_t}^2\mathbf{E}||N(t)||^2<\infty, \tag{37}$$

then algorithm $\mathcal{AN}$ will converge almost surely to a stationary point $(\vec{y^*},\vec{q^*})$ of algorithm $\mathcal{A}$.

Assumption (36) simply states that the noise term $N(t)$ should be un-biased. Assumption (37) is also quite general. For example, it will hold if the variance of the noise, i.e., $\mathbf{E}[(n^l(t))^2]$ is bounded for all l and t. We can prove Proposition 5 by first extending the analysis of Proposition 4 to show that, as $t\to\infty$, $\mathcal{V}(\vec{y}(t),\vec{q}(t))$ converges almost surely to a finite non-negative number. This implies that $(\vec{y}(t),\vec{q}(t))$ is bounded almost surely. We can then use the ODE method of [22] to show that, as $t\to\infty$, the limiting behavior of the stochastic approximation algorithm will converge to that of the ordinary differential equations defined by the continuous-time algorithm $\mathcal{AC}$ in Section IV with $\hat{A}=A_0$ and $\hat{B}=B_0$. Proposition 2 can then be invoked to show that $(\vec{y}(t),\vec{q}(t))$ converges to a stationary point. Due to lack of space, the details of the proof are available online in [17].

We now comment on the step-size rule used in Proposition 5. As is typical for stochastic approximation algorithms, the convergence of algorithm $\mathcal{AN}$ is established when the step-sizes are driven to zero. When this type of stochastic approximation algorithms are employed online, we usually use step-sizes that are away from zero (e.g., constants). In this case, the trajectory $(\vec{y}(t),\vec{q}(t))$ (or $(\vec{x}(t),\vec{q}(t))$) will fluctuate in a neighborhood around the set of stationary points, instead of converging to one stationary point. In practice, we are interested in knowing how to choose the step-sizes so that the trajectory stays in a close neighborhood around the solutions. Since Proposition 5 requires that both α^l and β_i be driven to zero, we would expect that, if we were to choose both α^l and β_i small enough (but away from zero), the trajectory $(\vec{x}(t),\vec{q}(t))$ will be kept in a close neighborhood around the solutions. This choice of the step-sizes might seem overly conservative at first sight. In particular, since the noise terms $n^l(t)$ are only present in the dual update (13), it appears at first quite plausible to conjecture that only α^l needs to be driven to zero in Proposition 5 (in order to average out the noise), while β_i can be kept away from zero. If this conjecture were true, it would imply that, in order to keep the trajectory $(\vec{x}(t),\vec{q}(t))$ in a close neighborhood around the set of stationary points, only α^l needs to be small. However, our simulation results with constant step-sizes seem to suggest the opposite. We observe that, when there is measurement noise, the disturbance in the primal variables $\vec{x}(t)$ cannot be effectively controlled by purely reducing the step-sizes α^l at the links. We will elaborate on this observation in the next section with a numerical example, and we will show that the required step-size rule (i.e., both α^l and β_i needs to small) is again a consequence of the multi-path nature of the problem.

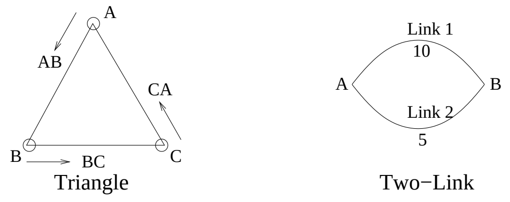

Fig. 1. Network Topologies

VI. Numerical Results

In this section, we present some simulation results for algorithm $\mathcal{A}$. For all simulations, we have chosen $K=1$, i.e., we do not use the two-level convergence structure. We will use the multi-path flow control problem as an example, but the results here apply to other problems as well. We first simulate the case when there is no measurement noise. We use the "Triangle" network in Fig. 1. There are three users (AB,BC,CA). For each user, there are two alternate paths, i.e., a direct one-link path (path 1), and an indirect two-link path (path 2). For example, user AB can take the one-link path $A\to B$ or the two-link path $A\to C\to B$. The utility functions for all three users are of the form:

$$f_i(\sum_{j=1}^{J(i)}x_{ij})=w_i\ln(\sum_{j=1}^{J(i)}x_{ij}),$$

where w_i is the "weight" of user i, and x_{ij} is the data rate of user i on path j. We choose the weights as follows: $w_{AB}=5.5,w_{BC}=2.5,w_{CA}=0.5$. The capacity on each link is 10 units.

Fig. 2 demonstrates the evolution over time of the implicit costs q^l and the users' data rates x_{ij}, respectively, for algorithms $\mathcal{A}$. We choose $c_i=1.0$ for all users. The step-sizes are $\alpha^l=0.1$ for all links, and $\beta_i=1.0$ for all users. We observe that all quantities of interest converge to the optimal solution, which is

$$q^{AB}=0.425,\quad q^{BC}=0.354,\quad q^{CA}=0.071,$$
$$x_{AB,1}=10,\quad x_{AB,2}=2.94,$$
$$x_{BC,1}=x_{CA,1}=7.06,\quad x_{BC,2}=x_{CA,2}=0.$$

Note that at the stationary point, user AB will use both alternative paths while users BC and CA will only use the direct paths. Because the weight of the utility function of user AB is larger than that of the other users, algorithm $\mathcal{A}$ automatically adjusts the resource allocation of users BC and CA to give way to user AB.

Fig. 3 demonstrates the evolution of algorithm $\mathcal{A}$ for the same network when there is measurement noise. We assume that an *i.i.d.* noise term uniformly distributed within $[-2,2]$ is added to each x_{ij} when each link estimates the aggregate load $\sum_{i=1}^{I}\sum_{j=1}^{J(i)}H_{ij}^l x_{ij}$. The step-sizes are $\alpha^l=0.003$ for all links, and $\beta_i=0.1$ for all users. We can observe that all quantities of interest eventually fluctuate around a small neighborhood of the solution.

We now investigate how the choice of the step-sizes α^l and β_i affect the level of fluctuation on the implicit costs and the

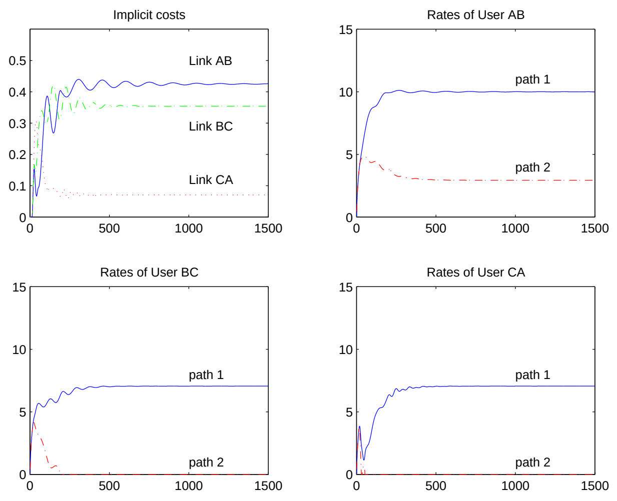

Fig. 2. Evolution of the implicit costs and the data rates when there is no measurement noise. $\alpha^l = 0.1$, $\beta_i = 1.0$.

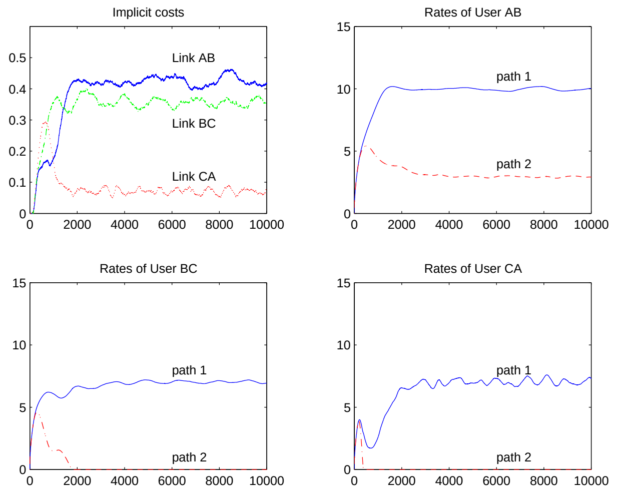

Fig. 3. Evolution of the implicit costs and the data rates when there is measurement noise. $\alpha^l = 0.003$, $\beta_i = 0.1$.

users' data rates when there is measurement noise. We use a simpler "Two-Link" topology in Fig. 1. The capacity of the two links is 10 and 5, respectively. There is only one user, which can use both links. Its utility function is given by

$$f_i(x) = 5.5 \ln x.$$

The noise term $n^l(t)$ is *i.i.d.* and uniformly distributed within $[-2, 2]$.

Fig. 4 shows the evolution over time of the implicit costs (top) and that of the users' data rates (bottom) of algorithm $\mathcal{A}$ for different choices of the step-sizes. In the first three columns, we keep β unchanged and reduce the step-size α from 0.01 to 0.0001. We observe that, although the fluctuation in the implicit costs becomes smaller as the step-size α is reduced, the fluctuation in the data rates decreases only a little. *Note that the unit on the x-axis becomes larger as we move from the first column to the third column.* These figures indicate that, by reducing the step-size α alone, the fluctuation in the data rates becomes slower, but the magnitude of the fluctuation changes little. In the fourth column, we decrease *both* β *and* α. The fluctuation in the data rates is now effectively reduced.

Although somewhat counter-intuitive, these observations are consistent with Proposition 5 where we require both α and β to be driven to zero for the convergence of the stochastic approximation algorithm to hold. As we will show next by studying the linearized version of the system, this step-size rule appears to be necessitated by the multi-path nature of the problem. Assume that algorithm $\mathcal{A}$ has a unique stationary point $(\vec{y^*}, \vec{q^*})$. We can linearize the continuous-time system (15)-(16) around this unique stationary point, and use Laplace transforms to study the frequency response of the system in the presence of noise $N(t)$. Without loss of generality, we can assume that $x_{ij}(t) > 0$ for all i, j and $q^l(t) > 0$ for all l. (Otherwise, we can eliminate the paths with $x_{ij}(t) = 0$ and the links with $q^l(t) = 0$ from the analysis because they do not contribute to the dynamics of the linearized system.) Let $\mathcal{X}(s)$ and $\mathcal{N}(s)$ denote the Laplace transform of the perturbation of $\vec{x}(t)$ and the noise $N(t)$, respectively. We can then compute the transfer function from $N(t)$ to $\vec{x}(t)$ as (see [17] for the detail)

$$\mathcal{X}(s) = H(s)\mathcal{N}(s)$$

with $H(s)$ equal to

$$-\left\{\hat{B}^{-1}s\mathcal{I} + G + V^{-1}(\mathcal{I} - G)\frac{E^T\hat{A}E}{s}\hat{B}^{-1}(s\mathcal{I} + \hat{B})\right\}^{-1} \times \hat{B}^{-1}(s\mathcal{I} + \hat{B})V^{-1}(\mathcal{I} - G)\frac{E^T\hat{A}}{s},$$

where the matrices E, $\hat{A}$, $\hat{B}$ and V are defined as in Section IV, $\mathcal{I}$ is the $\sum_{i=1}^{I} J(i) \times \sum_{i=1}^{I} J(i)$ identity matrix, $G = \mathbf{diag}\{G_i, i = 1, ..., I\}$ and each G_i is a $J(i) \times J(i)$ matrix whose elements are all

$$g_i = \frac{f_i''(\sum_{j=1}^{J(i)} y_{ij}^*)}{J(i)f_i''(\sum_{j=1}^{J(i)} y_{ij}^*) - c_i}.$$

If the utility function $f_i(\cdot)$ is strictly concave, g_i will be positive. However, since each G_i has all identical elements, the matrix G is not invertible whenever some users have multiple paths. Its presence in the denominator of $H(s)$ turns out to be a source of instability. To see this, we compute $H(s)$ for the above Two-Link example. Note that

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad G = g\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \text{ for some } g > 0,$$
$$\hat{A} = \alpha\mathcal{I}, \quad \hat{B} = \beta\mathcal{I}, \text{ and } V = c\mathcal{I}.$$

Let $s = j\omega$. We have,

$$H(j\omega) = -\left\{\frac{j\omega}{\beta}\mathcal{I} + G + \frac{\alpha}{c}\frac{j\omega + \beta}{j\omega\beta}(\mathcal{I} - G)\right\}^{-1} \times \frac{\alpha}{c}\frac{j\omega + \beta}{j\omega\beta}(\mathcal{I} - G). \quad (38)$$

The terms in the denominator can be collected into

$$\left[\frac{\alpha}{\beta c} + \frac{j\omega}{\beta}(1 - \frac{\alpha\beta}{c\omega^2})\right]\mathcal{I} + \left[(1 - \frac{\alpha}{\beta c}) + j\frac{\alpha}{c\omega}\right]G. \quad (39)$$

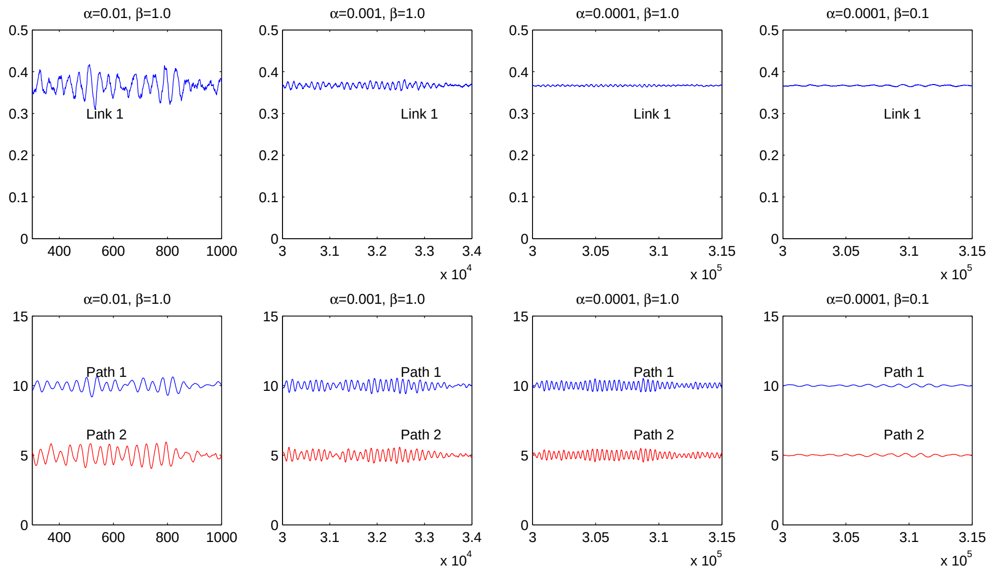

Fig. 4. Simulation of Algorithm $\mathcal{A}$ with measurement noise. Top: the implicit costs. Bottom: the data rates. *Note that the unit on the x-axis becomes larger as we move from the first column to the third column.*

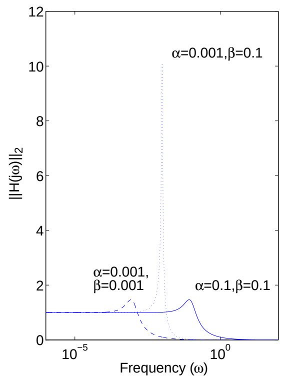

Fig. 5. The frequency response $||H(j\omega)||_2$

Since the matrix G is not invertible, if the terms that are associated with $\mathcal{I}$ is small, a "spike" in the transfer function $H(j\omega)$ will appear. This will happen when $\omega \approx \omega^*$, where ω^* is determined by $1 - \frac{\alpha\beta}{c(\omega^*)^2} = 0$, i.e.,

$$\omega^* = \sqrt{\frac{\alpha\beta}{c}}. \tag{40}$$

Substituting ω^* into (38) and (39) and assuming that $\alpha \ll \beta$, we have

$$\begin{aligned} H(j\omega^*) &\approx -\frac{\frac{\alpha}{c}(\frac{1}{\beta} + \frac{1}{j\omega})}{\frac{\alpha}{\beta c}}(\mathcal{I} - G) \\ &\approx j\frac{\frac{\alpha}{\omega c}}{\frac{\alpha}{\beta c}}(\mathcal{I} - G) \approx j\sqrt{\frac{c\beta}{\alpha}}(\mathcal{I} - G). \end{aligned} \tag{41}$$

We can draw the following conclusions from equations (40) and (41). If we keep β fixed and reduce α alone, the cutoff frequency ω^* will decrease with α. However, the gain $H(j\omega^*)$ at the cutoff frequency will increase! In Fig. 5, we plot $||H(j\omega)||_2$ with respect to ω for different values of α and β. We can easily observe the increased spike when α alone is reduced from 0.1 (the solid curve) to 0.001 (the dotted curve). If we further assume that $n^l(t)$ is white noise with unit energy, then the total energy of the fluctuation of $\vec{x}$ can be estimated by the area under the curve $||H(j\omega)||_2$ in Fig. 5. Due to the increased spike, the total energy of the fluctuation of $\vec{x}(t)$ will not decrease much when α alone is reduced, even though the frequency of the fluctuation becomes smaller. On the other hand, if we reduce β as well as α, the gain at the cutoff frequency ω^* will remain the same as the cutoff frequency itself decreases (shown as the dashed curve in Fig. 5 when β is also reduced to 0.001). Hence, the total energy of the fluctuation in $\vec{x}(t)$ is effectively reduced. These conclusions are thus consistent with our simulation results in Fig. 4. Hence, the step-size rule (i.e., both α^l and β_i needs to be reduced) is necessary to address the potential instability in the system due to the multi-path nature of the problem.

VII. Concluding Remarks

In this work, we have developed a distributed algorithm for the utility maximization problem in communication networks that have the capability to allow multi-path routing. We have studied the convergence of our algorithm in both continuous-time and discrete-time, with and without measurement noise. We have shown how the multi-path nature of the problem can potentially lead to difficulties such as instability and oscillation, and our analyses provide important guidelines on how to choose the parameters of the algorithm to address these difficulties and to ensure efficient network control. When there is no measurement noise, our analysis gives easy-to-verify conditions on how large the step-sizes of the algorithm can be while still ensuring convergence. When there is measurement noise, we find that the step-sizes in the updates of both the user algorithm and the network component algorithm have to decrease at the same time in order to reduce the fluctuation of the resource allocation. Reducing only the step-sizes in the update of the network component algorithm will reduce the frequency of the fluctuation, but not necessarily its magnitude. These guidelines are confirmed by our simulation results.

We briefly discuss possible directions for future work. The analysis in this work has focused on the case when all computation is synchronized. An interesting problem is to study the convergence and stability of the algorithm when the computation is asynchronous and when feedback delays are non-negligible. Simulations suggest that our distributed algorithm may still be used in those situations, however, the step-size rules may need to change. Another direction is to extend our solution to resource allocation problems in wireless networks. In wireline networks, the resource constraints of different network components are orthogonal to each other.

In wireless networks, however, the capacity of a link is a function of the signal to interference ratio, which depends not only on its own transmission power, but also on the power assignments at other links. Hence, the resource constraints in wireless networks are of a more complex form than that of wireline networks. It would be interesting to see whether the results of this paper can be extended to multi-path utility maximization problems in wireless networks.

Appendix: Proof of Lemma 3

We need to use the fact that $\mathbf{f}(\vec{x})$ is of the form $\sum_{i=1}^{I} f_i(\sum_{j=1}^{J(i)} x_{ij})$, and that the Lagrangian $L(\vec{x},\vec{q},\vec{y})$ is given by (7). The maximization of the Lagrangian (in (8)) is taken over $\vec{x}_i \in C_i$ for all i. Since C_i is of the form in (4), we can incorporate the constraint $\sum_{j=1}^{J(i)} x_{ij} \in [m_i, M_i]$ into the definition of the function f_i by setting $f_i(x) = -\infty$ when $x \notin [m_i, M_i]$. Then the function f_i is still concave, and the maximization of the Lagrangian $L(\vec{x},\vec{q},\vec{y})$ can be taken over all $\vec{x} \geq 0$. Given $\vec{y}$ and $\vec{q}$, we associate a Lagrange multiplier L_{ij}^0 for each constraint $x_{ij} \geq 0$ in the maximization of $L(\vec{x},\vec{q},\vec{y})$, and let $\vec{x^0} = \text{argmax}_{\vec{x}\geq 0} L(\vec{x},\vec{q},\vec{y})$. Using the Karush-Kuhn-Tucker condition, we can conclude that, for each i, there must exist a subgradient $\partial f_i(\sum_{j=1}^{J(i)} x_{ij,0})$ of f_i at $\sum_{j=1}^{J(i)} x_{ij,0}$ such that, for all j,

$$\partial f_i(\sum_{j=1}^{J(i)} x_{ij,0}) - \sum_{l=1}^{L} E_{ij}^l q^l - c_i(x_{ij,0} - y_{ij}) + L_{ij}^0 = 0, \quad (42)$$

and $L_{ij}^0 x_{ij,0} = 0$. Similarly, let $(\vec{y^*}, \vec{q^*})$ denote a stationary point of algorithm $\mathcal{A}$. Then $\vec{y^*} = \text{argmax}_{\vec{x}\geq 0} L(\vec{x},\vec{q^*},\vec{y^*})$. Associate a Lagrange multiplier L_{ij}^* for each constraint $x_{ij} \geq 0$ in the maximization of $L(\vec{x},\vec{q^*},\vec{y^*})$. Then, for all i,j,

$$\partial f_i(\sum_{j=1}^{J(i)} y_{ij}^*) - \sum_{l=1}^{L} E_{ij}^l q^{l,*} + L_{ij}^* = 0, \quad (43)$$

and $L_{ij}^* y_{ij}^* = 0$. Comparing (42) and (43) with (21) and (22), we see that

$$[\nabla \mathbf{f}(\vec{x^0})]_{ij} = \partial f_i(\sum_{j=1}^{J(i)} x_{ij,0}) + L_{ij}^0 \text{ for all } i,j,$$

$$[\nabla \mathbf{f}(\vec{y^*})]_{ij} = \partial f_i(\sum_{j=1}^{J(i)} y_{ij}^*) + L_{ij}^* \text{ for all } i,j,$$

where $[\cdot]_{ij}$ is the element in $[\cdot]$ that corresponds to x_{ij}.

We can now proceed with the proof of Lemma 3. Let $\vec{x}_1 = \text{argmax}_{\vec{x}\geq 0} L(\vec{x},\vec{q}_1,\vec{y})$ and $\vec{x}_2 = \text{argmax}_{\vec{x}\geq 0} L(\vec{x},\vec{q}_2,\vec{y})$. Analogously to L_{ij}^0 and $\partial f_i(\sum_{j=1}^{J(i)} x_{ij,0})$, define $L_{ij,1}$, $\partial f_i(\sum_{j=1}^{J(i)} x_{ij,1})$ and $L_{ij,2}$, $\partial f_i(\sum_{j=1}^{J(i)} x_{ij,2})$ for the case when the implicit cost vectors are $\vec{q}_1$ and $\vec{q}_2$, respectively. Then,

$$\begin{aligned}&\left[\nabla \mathbf{f}(\vec{x}_1) - \nabla \mathbf{f}(\vec{y^*})\right]^T (\vec{x}_2 - \vec{y^*}) \\ =&\sum_{i=1}^{I}\left[\partial f_i(\sum_{j=1}^{J(i)} x_{ij,1}) - \partial f_i(\sum_{j=1}^{J(i)} y_{ij}^*)\right](\sum_{j=1}^{J(i)} x_{ij,2} - \sum_{j=1}^{J(i)} y_{ij}^*) \quad (44)\\ &+\sum_{i=1}^{I}\sum_{j=1}^{J(i)}(L_{ij,1} - L_{ij}^*)(x_{ij,2} - y_{ij}^*). \quad (45)\end{aligned}$$

Lemma 3 will follow if we can show that both of the two terms (44) and (45) are bounded by $\frac{1}{4c_i}\sum_{i=1}^{I}\sum_{j=1}^{J(i)}\left[\sum_{l=1}^{L} E_{ij}^l (q_2^l - q_1^l)\right]^2$. We will first bound the term (44). Apply equation (42) for $\vec{q}_1$ and $\vec{q}_2$, respectively, and take difference. We have, for each i,j,

$$\begin{aligned}\sum_{l=1}^{L} E_{ij}^l (q_2^l - q_1^l) = &\left[\partial f_i(\sum_{j=1}^{J(i)} x_{ij,2}) - \partial f_i(\sum_{j=1}^{J(i)} x_{ij,1})\right] \\ &-c_i(x_{ij,2} - x_{ij,1}) + L_{ij,2} - L_{ij,1}. \quad (46)\end{aligned}$$

Now fix i. Let J_i denote the set $\{j : x_{ij,2} > 0 \text{ or } x_{ij,1} > 0\}$. Note that if $x_{ij,2} > 0$ and $x_{ij,1} = 0$, then $L_{ij,2} = 0$ and $L_{ij,1} \geq 0$. Hence, $x_{ij,2} - x_{ij,1} > 0$ and $L_{ij,2} - L_{ij,1} \leq 0$. Let

$$\gamma_{ij} \triangleq -\frac{L_{ij,2} - L_{ij,1}}{c_i(x_{ij,2} - x_{ij,1})} \geq 0,$$

then,

$$\begin{aligned}\sum_{l=1}^{L} E_{ij}^l (q_2^l - q_1^l) \quad = \quad &\left[\partial f_i(\sum_{j=1}^{J(i)} x_{ij,2}) - \partial f_i(\sum_{j=1}^{J(i)} x_{ij,1})\right] \\ &-(1+\gamma_{ij})c_i(x_{ij,2} - x_{ij,1}). \quad (47)\end{aligned}$$

Similarly, we can show that (47) holds for any $j \in J_i$ with some appropriate choice of $\gamma_{ij} \geq 0$. Multiplying (47) by $1/(1+\gamma_{ij})$ and summing over all $j \in J_i$, we have, for all i,

$$\begin{aligned}&\sum_{j\in J_i}\left[\frac{1}{1+\gamma_{ij}}\sum_{l=1}^{L} E_{ij}^l (q_2^l - q_1^l)\right] \\ = \quad &\frac{1}{\gamma_i^0}\left[\partial f_i(\sum_{j=1}^{J(i)} x_{ij,2}) - \partial f_i(\sum_{j=1}^{J(i)} x_{ij,1})\right] \\ &-c_i(\sum_{j=1}^{J(i)} x_{ij,2} - \sum_{j=1}^{J(i)} x_{ij,1}), \quad (48)\end{aligned}$$

where $\gamma_i^0 \triangleq \frac{1}{\sum_{j\in J_i}\frac{1}{1+\gamma_{ij}}}$, and we have used the fact that $x_{ij,2} = x_{ij,1} = 0$ for $j \notin J_i$.

Let

$$a_1 = \partial f_i(\sum_{j=1}^{J(i)} x_{ij,1}) - \partial f_i(\sum_{j=1}^{J(i)} y_{ij}^*),$$

$$a_2 = \partial f_i(\sum_{j=1}^{J(i)} x_{ij,2}) - \partial f_i(\sum_{j=1}^{J(i)} y_{ij}^*),$$

$$b_1 = \sum_{j=1}^{J(i)} x_{ij,1} - \sum_{j=1}^{J(i)} y_{ij}^*, \text{ and } b_2 = \sum_{j=1}^{J(i)} x_{ij,2} - \sum_{j=1}^{J(i)} y_{ij}^*.$$

Since the function f_i is concave, we have $a_1 b_1 \leq 0$ and $a_2 b_2 \leq 0$. Let $\gamma_i \triangleq -\frac{c_i \gamma_i^0 b_1}{a_1} \geq 0$. (The term (44) will be bounded by $\frac{1}{4c_i} \sum_{i=1}^{I} \sum_{j=1}^{J(i)} \left[\sum_{l=1}^{L} E_{ij}^l (q_2^l - q_1^l) \right]^2$ trivially if $a_1 = 0$.) Then

$$\begin{aligned} & (1+\gamma_i) a_1 b_2 = (a_1 - c_i \gamma_i^0 b_1) b_2 \\ = \quad & [(a_1 - a_2) - c_i \gamma_i^0 (b_1 - b_2)] b_2 + (a_2 - c_i \gamma_i^0 b_2)] b_2 \\ \leq \quad & \gamma_i^0 \sum_{j \in J_i} \left[\frac{1}{1+\gamma_{ij}} \sum_{l=1}^{L} E_{ij}^l (q_2^l - q_1^l) \right] b_2 \quad \text{(by (48))} \\ & \quad -c_i \gamma_i^0 b_2^2 \quad \text{(by } a_2 b_2 \leq 0) \\ \leq \quad & \frac{\gamma_i^0}{4c_i} \left\{ \sum_{j \in J_i} \left[\frac{1}{1+\gamma_{ij}} \sum_{l=1}^{L} E_{ij}^l (q_2^l - q_1^l) \right] \right\}^2 \\ & \quad \text{(by completing the square)} \\ \leq \quad & \frac{\gamma_i^0}{4c_i} \left\{ \sum_{j \in J_i} (\frac{1}{1+\gamma_{ij}})^2 \right\} \left\{ \sum_{j \in J_i} \left[\sum_{l=1}^{L} E_{ij}^l (q_2^l - q_1^l) \right]^2 \right\} \\ & \quad \text{(by Cauchy-Schwarz)} \\ \leq \quad & \frac{1}{4c_i} \sum_{j=1}^{J(i)} \left[\sum_{l=1}^{L} E_{ij}^l (q_2^l - q_1^l) \right]^2 , \end{aligned}$$

where in the last inequality we have used

$$\gamma_i^0 \left\{ \sum_{j \in J_i} (\frac{1}{1+\gamma_{ij}})^2 \right\} = \frac{\sum_{j \in J_i} (\frac{1}{1+\gamma_{ij}})^2}{\sum_{j \in J_i} \frac{1}{1+\gamma_{ij}}} \leq 1.$$

Since $1 + \gamma_i \geq 1$, the term (44) (i.e., $a_1 b_2$) is then bounded by $\frac{1}{4c_i} \sum_{i=1}^{I} \sum_{j=1}^{J(i)} \left[\sum_{l=1}^{L} E_{ij}^l (q_2^l - q_1^l) \right]^2$.

To bound the term (45), note that $x_{ij,2} \geq 0, L_{ij,1} \geq 0, L_{ij}^* \geq 0$ and $L_{ij}^* y_{ij}^* = L_{ij,2} x_{ij,2} = L_{ij,1} x_{ij,1} = 0$. Therefore,

$$\begin{aligned} & (L_{ij,1} - L_{ij}^*)(x_{ij,2} - y_{ij}^*) \leq x_{ij,2} L_{ij,1} \\ \leq \quad & x_{ij,1} L_{ij,2} + x_{ij,2} L_{ij,1} \\ = \quad & -(L_{ij,2} - L_{ij,1})(x_{ij,2} - x_{ij,1}). \end{aligned}$$

We thus have,

$$\begin{aligned} & \sum_{j=1}^{J(i)} (L_{ij,1} - L_{ij}^*)(x_{ij,2} - y_{ij}^*) \\ \leq & - \sum_{j=1}^{J(i)} (L_{ij,2} - L_{ij,1})(x_{ij,2} - x_{ij,1}) \\ \leq & - \sum_{j=1}^{J(i)} \left[\partial f_i (\sum_{j=1}^{J(i)} x_{ij,2}) - \partial f_i (\sum_{j=1}^{J(i)} x_{ij,1}) + L_{ij,2} - L_{ij,1} \right] \\ & \quad \times (x_{ij,2} - x_{ij,1}) \quad \text{(by the concavity of } \partial f_i) \\ \leq & - \sum_{j=1}^{J(i)} \left[\sum_{l=1}^{L} E_{ij}^l (q_2^l - q_1^l) + c_i (x_{ij,2} - x_{ij,1}) \right] \\ & \quad \times (x_{ij,2} - x_{ij,1}) \quad \text{(by (46))} \\ \leq & \frac{1}{4c_i} \sum_{j=1}^{J(i)} \left[\sum_{l=1}^{L} E_{ij}^l (q_2^l - q_1^l) \right]^2 \\ & \quad \text{(by completing the square).} \end{aligned}$$

The result of Lemma 3 then follows.

References

[1] X. Lin and N. B. Shroff, "The Multi-path Utility Maximization Problem," in *41st Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, October 2003.

[2] F. P. Kelly, A. Maulloo, and D. Tan, "Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[3] K. Kar, S. Sarkar, and L. Tassiulas, "Optimization Based Rate Control for Multipath Sessions," *Technical Report No. 2001-1, Institute for Systems Research, University of Maryland*, 2001.

[4] W. H. Wang, M. Palaniswami, and S. H. Low, "Optimal Flow Control and Routing in Multi-Path Networks," *Performance Evaluation*, vol. 52, no. 2-3, pp. 119–132, April 2003.

[5] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Overlay TCP for Multi-Path Routing and Congestion Control," *Presented at the ENS-INRIA ARC-TCP Workshop, Paris, France, Nov. 2003 and at the IMA Workshop on Measurements and Modeling of the Internet, January 2004, avilalable at http://tesla.csl.uiuc.edu/∼srikant/pub.html*.

[6] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, "Quality of Service Based Routing: A Performance Perspective," in *Proceedings of ACM SIGCOMM*, Vancouver, Canada, September 1998, pp. 17–28.

[7] Q. Ma and P. Steenkiste, "On Path Selection for Traffi c with Bandwidth Guarantees," in *IEEE ICNP*, 1997.

[8] A. Shaikh, J. Rexford, and K. Shin, "Evaluating the Impact of Stale Link State on Quality-of-Service Routing," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, pp. 162–176, April 2001.

[9] P. B. Key, "Optimal Control and Trunk Reservation in Loss Networks," *Probability in the Engineering and Informational Sciences*, vol. 4, pp. 203–242, 1990.

[10] X. Lin and N. B. Shroff, "An Optimization Based Approach for Quality of Service Routing in High-Bandwidth Networks," in *Proceedings of IEEE INFOCOM*, Hong Kong, China, March 2004.

[11] S. H. Low and D. E. Lapsley, "Optimization Flow Control–I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, December 1999.

[12] S. Kunniyur and R. Srikant, "End-to-End Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks," in *Proceedings of IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.

[13] H. Yaiche, R. Mazumdar, and C. Rosenberg, "A Game Theoretic Framework for Bandwidth Allocation and Pricing in Broadband Networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 667–678, Oct. 2000.

[14] S. H. Low and R. Srikant, "A Mathematical Framework for Designing a Low-Loss Low-Delay Internet," *Network and Spatial Economics*, vol. 4, no. 1, pp. 75–102, March 2004.

[15] S. Kunniyur and R. Srikant, "An Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 286–299, April 2004.

[16] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, New Jersey, 1989.

[17] X. Lin and N. B. Shroff, " Utility Maximization for Communication Networks with Multi-path Routing," *Technical Report, Purdue University, http://min.ecn.purdue.edu/∼linx/papers.html*, 2004.

[18] R. T. Rockafellar, "Monotone Operators and the Proximal Point Algorithm," *SIAM J. Control and Optimization*, vol. 14, pp. 877–898, August 1976.

[19] S. Wright, *Primal-Dual Interior-Point Methods*, Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[20] J. Eckstein, *Splitting Methods for Monotone Operators With Applications to Prallel Optimization*, PhD thesis, Massachusetts Institute of Technology, Department of Civil Engineering, 1989.

[21] H. K. Khalil, *Nonlinear Systems*, Prentice-Hall, Upper Saddle River, New Jersey, second edition, 1996.

[22] H. J. Kushner and G. Yin, *Stochastic Approximation Algorithms and Applications*, Springer-Verlag, New York, 1997.